Problèmes d'homomorphisme à largeur de chemin bornée

Mémoire

Catherine Bédard

Maîtrise en informatique
Maître ès sciences (M.Sc.)

Québec, Canada

Résumé

Un homomorphisme est une fonction entre deux structures, par exemple des graphes, qui respecte certaines contraintes. Dans ce mémoire, on étudie la complexité des problèmes d'homomorphisme, c'est-à-dire des problèmes où l'on doit décider s'il existe une telle fonction entre deux structures. On présentera des propriétés sur ces structures qui permettent de déterminer cette complexité. On s'intéressera particulièrement aux problèmes d'homomorphisme qui appartiennent à la classe de complexité NL, une classe contenant des problèmes dont la résolution par un algorithme non déterministe nécéssite peu d'espace mémoire.

Table des matières

Remerciements

Je veux d'abord remercier mon directeur Pascal Tesson pour sa grande générosité dans son encadrement, sa disponibilité et son temps. Je tiens aussi à le remercier pour le financement et le matériel qu'il a mis à ma disposition.

Je remercie aussi Maxime Dubé d'avoir élaboré avec moi l'algorithme 3.2. Cette aide précieuse a été grandement appréciée. Je remercie François Laviolette pour ses conseils et ses commentaires sur plusieurs parties de ce mémoire.

Je remercie aussi mes correcteurs Danny Dubé et Nadia Tawbi pour leurs commentaires ainsi que Brahim Chaib-draa pour le rôle qu'il a joué.

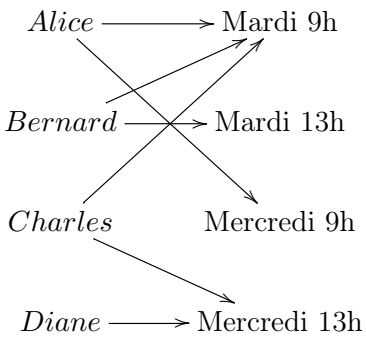
Sur un plan plus personnel, je veux remercier mes parents pour leur aide financière et morale. Je les remercie de m'avoir donné le goût des études. Je remercie finalement mon conjoint et mes proches pour tout leur support.

Introduction

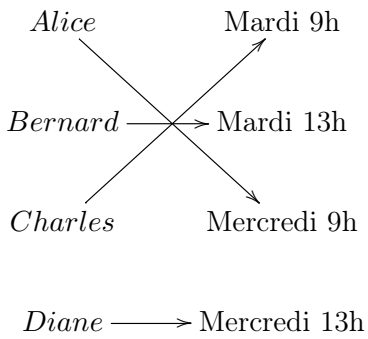
Les problèmes de satisfaction de contraintes sont des problèmes qui s'énoncent comme «Étant données des variables qui peuvent prendre certaines valeurs, est-il possible d'assigner une valeur à chacune des variables qui respecte certaines contraintes ?».

Par exemple, un problème de couplage comme «Étant donnés des conférenciers et un ensemble de plages horaires disponibles, est-il possible d'assigner une heure à chacun des conférenciers en respectant les disponibilités de chacun ?» est un problème de satisfaction de contraintes. Les variables sont les conférenciers, le domaine contenant les valeurs possibles pour chaque conférencier est l'ensemble des plages horaires disponibles et les contraintes sont les suivantes :

- Deux conférenciers ne peuvent pas êtres assignés à la même plage horaire
- La plage horaire assignée à chaque conférencier doit respecter ses disponibilités.



(a) Disponibilités des conférenciers



(b) Une solution possible au problème

FIGURE 0.1 – Exemple de problème de couplage : Les variables sont {Alice, Bernard, Charles, Diane}, les domaines sont {Mardi 9h, Mardi 13h, Mercredi 9h, Mercredi 13h}. Les contraintes à respecter sont de placer Alice à 9h, de placer Bernard le mardi, de placer Charles le mardi à 9h ou le mercredi à 13h et de placer Diane le mercredi à 13h.

Le problème «Est-il possible de donner une couleur parmi bleu, vert ou jaune à chacun des sommets de ce graphe de manière à ce que deux sommets reliés par une arête soient assignés

à des couleurs différentes ?» est lui aussi un problème de satisfaction de contraintes que nous appelerons problème de 3-coloriage de graphe.

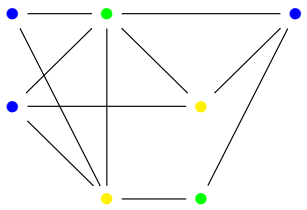


FIGURE 0.2 – Exemple de solution au problème de 3-coloriage de graphe

Nous allons définir rigoureusement les problèmes de satisfaction de contraintes et nous allons remarquer qu'il est possible de les exprimer d'une autre façon, sous forme de problèmes d'homomorphisme. Un homomorphisme est une fonction entre un ensemble de départ et un ensemble d'arrivée qui préserve certaines propriétés.

On peut traduire tout problème de satisfaction de contraintes en un problème d'homomorphisme. Un homomorphisme entre deux graphes $\mathcal{G}$ et $\mathcal{H}$ est une fonction des sommets de $\mathcal{G}$ vers les sommets de $\mathcal{H}$ telle que si (u, v) est une arête dans $\mathcal{G}$, alors $(f(u), f(v))$ est une arête dans $\mathcal{H}$. Cette définition nous amène à considérer une foule de nouveaux problèmes. Nous nous attarderons à deux de ceux-ci. D'abord, on voudra considérer le problème $HOM(-, \mathcal{H})$: «Étant donné un graphe fixé $\mathcal{H}$ et un graphe $\mathcal{G}$ donné en entrée, existe-t-il un homomorphisme de $\mathcal{G}$ vers $\mathcal{H}$?».

Ensuite, on s'attardera au problème $HOM(C, -)$: «Étant donné un ensemble de graphes C fixé, un graphe $\mathcal{H}$ appartenant à cet ensemble C et un graphe $\mathcal{G}$ quelconque donné en entrée, existe-t-il un homomorphisme de $\mathcal{H}$ vers $\mathcal{G}$?».

Dans ce mémoire on cherchera à classifier ces problèmes en fonction de leur complexité. Nous verrons d'abord comment définir cette complexité en nous donnant des outils afin de comparer les problèmes entre eux.

Pour ce faire, nous allons définir des ensembles pour classer ces divers problèmes. Nous appelerons ces ensembles des classes de complexité.

Les problèmes dans P sont des problèmes pour lesquels il existe un algorithme qui trouve la réponse rapidement, c'est-à-dire en temps polynomial par rapport à la taille de l'entrée.

Par exemple, pour notre problème de 3-coloriage de graphe, l'entrée est un graphe. Si n est le nombre de sommets de ce graphe, un algorithme qui trouve la réponse au problème en n^c étapes, où c est un nombre constant, serait un algorithme polynomial. Si un tel algorithme existait, ce problème serait dans la classe P.

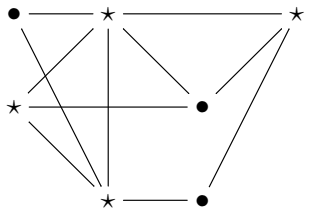


FIGURE 0.3 – Exemple de couverture par sommet avec $k = 4$

Or, aucun algorithme polynomial n'est encore connu pour ce problème. Nous allons donc devoir mettre ce problème dans une autre classe que P. Nous ne connaissons pas d'algorithme polynomial pour trouver une solution mais on remarque que si quelqu'un nous donne le graphe avec les sommets déjà coloriés, on peut vérifier rapidement que ce coloriage satisfait les contraintes du problème.

La classe NP est la classe des problèmes pour lesquels il est rapide de vérifier si une solution donnée est exacte. Le problème de coloriage de graphe est donc dans NP. Tous les problèmes de P sont aussi dans NP.

Au deuxième chapitre, nous allons voir que si le graphe $\mathcal{H}$ a certaines propriétés, alors le problème $HOM(-, \mathcal{H})$ est dans P. Nous verrons de plus que si au contraire, $\mathcal{H}$ ne possède pas ces propriétés, alors $HOM(-, \mathcal{H})$ est NP-complet.

Pour ce faire, nous allons d'abord présenter des outils mathématiques qui ont été développés récemment pour aider à étudier la complexité des problèmes de satisfaction de contraintes. Nous démontrerons ensuite le résultat en faisant appel à ces outils.

Les classes de complexité P et NP sont bien connues et étudiées. Malgré toutes ces recherches, on ne sait toujours pas si tous les problèmes de NP sont les mêmes que les problèmes de P. Entre autres, on ne connaît pas d'algorithme polynomial pour le problème de 3-coloriage de graphe.

Devant cette impasse, on tente d'assouplir nos définitions. Par exemple, le problème de couverture par sommets est le problème qui demande, étant donné un graphe et un nombre k de sentinelles, s'il est possible de placer exactement k sentinelles sur les sommets du graphe de manière à ce que chaque arête soit adjacente à au moins une sentinelle. On connaît un algorithme qui trouve une solution au problème de couverture par sommet qui s'effectue en $2^k \cdot n^c$ étapes, où n est la taille du graphe donné en entrée et c est une constante. Cet algorithme n'est pas polynomial mais le serait si l'on pouvait considérer k comme une constante.

Nous allons considérer les classes FPT et W[1]. La classe FPT est la classe des problèmes qui, tout comme le problème de couverture par sommets, seraient résolubles en temps polynomial si l'on pouvait considérer une partie de l'entrée comme une constante. La classe W[1] contient

tous les problèmes dans FPT, ainsi que les problèmes dont on ignore s'ils ont cette propriété.

Nous allons voir au dernier chapitre que si l'ensemble de graphes C a une certaine propriété de théorie des graphes, alors le problème $HOM(C, -)$ est dans P. Nous verrons que sinon, le problème est dans la classe W[1].

Cette propriété sur l'ensemble C est que tous les graphes de cet ensemble ont une largeur d'arbre bornée. Nous allons définir rigoureusement au premier chapitre ce qu'est une largeur d'arbre bornée, mais l'intuition derrière ce concept est que les graphes dans la classe C sont *presque* des arbres.

Nous nous sommes posés la question à savoir ce qui arrive si, plutôt que d'être presque des arbres, les graphes dans C étaient presque des chemins. Pourrait-on dire que le problème $HOM(C, -)$ est encore plus facile que les problèmes de P ?

On va définir des classes de complexité qui raffinent la classe P. Cette fois, plutôt que de considérer le temps nécéssaire pour résoudre un problème, nous voudrons limiter l'espace mémoire utilisé par un algorithme qui résout le problème. Cet espace devra être logarithmique par rapport à la taille de l'entrée. Ces classes sont appelées L et NL et sont de sorte que

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP}.$$

On verra qu'en restreignant un peu plus la propriété stipulant que C ne contienne que des graphes qui sont presque des chemins, alors on peut dire que le problème $HOM(C, -)$ est dans NL.

Chapitre 1

Définitions préalables

1.1 Les problèmes de décision

Un problème de décision est un problème dont la sortie est toujours 1 ou 0, «oui» ou «non». Par exemple, le problème «Trouver un cycle hamiltonien dans le graphe $\mathcal{G}$» n'est pas un problème de décision mais le problème «Existe-t-il un cycle hamiltonien dans le graphe $\mathcal{G}$?» en est un.

On peut définir plus rigoureusement ces problèmes comme des problèmes d'appartenance à un ensemble. Supposons par exemple que l'ensemble Σ est l'ensemble qui contient tous les graphes. On aura alors que les instances positives du problème «Existe-t-il un cycle hamiltonien dans le graphe $\mathcal{G}$?» sont les éléments de l'ensemble $Q \subset \Sigma$ de tous les graphes qui contiennent un cycle hamiltonien. Si le graphe $\mathcal{G}$ contient un cycle hamiltonien, on pourra écrire $\mathcal{G} \in Q$.

Voici deux autres problèmes de décision qui seront importants pour nous, les problèmes clique et 3-SAT.

Pour un graphe $\mathcal{G}$, nous noterons $V(\mathcal{G})$ l'ensemble qui contient tous les sommets de $\mathcal{G}$ et $E(\mathcal{G})$ l'ensemble qui contient toutes ses arêtes.

Une clique dans un graphe $\mathcal{G}$ est un sous-ensemble de sommets qui sont tous reliés entre eux. Autrement dit, $C \subseteq V(\mathcal{G})$ est une clique de $\mathcal{G}$ si pour tous les sommets x et y qui sont dans C, il y a une arête $(x, y) \in E(\mathcal{G})$. La taille d'une clique est le nombre de sommets qu'elle contient.

Définition 1 (Problème clique) *Le problème suivant est un problème de décision :*

- *Entrée : $(\mathcal{G}, k)$ où $\mathcal{G}$ est un graphe et $k \in \mathbb{N}$*
- *Problème : Déterminer s'il existe dans $\mathcal{G}$ une clique de taille k*

On peut aussi définir le problème clique comme l'ensemble $Q \subseteq \Sigma$ où

- $\Sigma = \{(\mathcal{G}, k) : \mathcal{G}$ *est un graphe et* $k \in \mathbb{N}\}$
- $Q = \{(\mathcal{G}, k) : \mathcal{G}$ *contient une clique de taille* $k\}$.

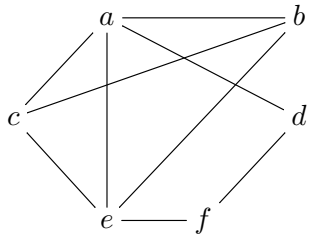


FIGURE 1.1 – Les sommets a, b, c et e forment une clique de taille 4 dans ce graphe.

On dira qu'une formule booléenne φ est en 3-CNF si elle est de la forme

$$\varphi = (l_1^1 \vee l_1^2 \vee l_1^3) \wedge (l_2^1 \vee l_2^2 \vee l_2^3) \wedge \cdots \wedge (l_m^1 \vee l_m^2 \vee l_m^3)$$

où $l_i^j \in \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$, avec $\{x_1, \ldots, x_n\}$ un ensemble de variables booléennes. De plus, on dira qu'une formule φ en 3-CNF est satisfiable s'il est possible de trouver une assignation de valeurs de vérité à ses variables $\{x_1, \ldots, x_n\}$ qui rendent la formule vraie.

Exemple 2 *La formule φ en 3-CNF*

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$$

est satisfiable car les valeurs $x_1 = x_2 = x_3 = 1$ et $x_4 = 0$ lui donnent la valeur 1.

Définition 3 (Problème 3-SAT) *Le problème suivant est un problème de décision :*

- *Entrée : φ, une formule booléenne en 3-CNF*
- *Problème : Déterminer si φ est satisfiable*

On peut aussi définir 3-SAT comme le problème $Q \subseteq \Sigma$ où

- $\Sigma = \{\varphi : \varphi$ *est une formule en 3-CNF*$\}$
- $Q = \{\varphi \in \Sigma : \varphi$ *est satisfiable*$\}$.

1.2 Les classes P et NP

Les classes de complexité les plus connues sont, sans contredit, les classes P et NP. Un problème est dans P s'il existe un algorithme qui le résout en temps polynomial et un problème est dans NP s'il est possible de vérifier qu'une solution à ce problème est exacte en un temps polynomial.

Définition 4 *Soient f, g des fonctions définies de $\mathbb{R}^+$ dans $\mathbb{R}^+$. On dit que $f(n) \in O(g(n))$ s'il existe des constantes c et $n_0 > 0$ telles que $f(n) \leq c \cdot g(n)$ pour tout $n \geq n_0$.*

Par exemple, si pour toute entrée de taille n, un algorithme s'effectue en $3n^4 + 2n + 6$ étapes, on dira que cet algorithme se termine en temps $O(n^4)$ car $3n^4+2n+6 \in O(n^4)$. Cet algortihme est un algorithme polynomial.

Définition 5 *Un problème de décision $Q \subseteq \Sigma$ appartient à la classe de complexité P s'il existe un algorithme A qui prend en entrée des instances $x \in \Sigma$ tel que*

1. *$\forall x \in \Sigma, A(x)$ s'exécute en temps polynomial par rapport à la taille de x*
2. *$\forall x \in \Sigma, A(x) = 1$ si et seulement si $x \in Q$.*

Définition 6 *Un problème de décision $Q \subseteq \Sigma$ appartient à la classe de complexité NP s'il existe un algorithme A qui prend en entrée des paires (x, X) avec $x \in \Sigma$ et X un vecteur de preuve de taille polynomiale par rapport à x tel que*

1. *$\forall x \in \Sigma$, on a que $A(x, X)$ s'exécute en temps polynomial par rapport à la taille de x,*
2. *$\forall x \in Q$, il existe une preuve X telle que $A(x, X) = 1$,*
3. *$\forall x \notin Q$, pour tout vecteur X, on a que $A(x, X) = 0$.*

Par exemple, 3-SAT est dans NP car l'algorithme qui prend en entrée une instance φ du problème 3-SAT et un vecteur $X \in \{0, 1\}^n$ et qui retourne 1 si et seulement si $\varphi(X) = 1$ est tel que

1. $A(\varphi, X)$ s'exécute en temps polynomial par rapport à la taille de φ
2. Si φ est satisfiable, alors il existe une assignation X des variables de φ telle que $\varphi(X) = A(\varphi, X) = 1$
3. Si φ n'est pas satisfiable, alors toutes les assignations de valeurs X sont telles que $\varphi(X) = A(\varphi, X) = 0$

1.2.1 Complétude

On remarque qu'une même classe de complexité peut contenir une très grande variété de problèmes. On a vu que 3-SAT, un problème reconnu pour être très complexe, est dans NP, mais de nombreux problèmes beaucoup plus simples font aussi partie de cette classe.

On aimerait se donner un outil afin de caractériser les problèmes les plus difficiles d'une classe de complexité. La notion de complétude nous permettra d'y arriver.

Un problème est complet pour une classe de complexité s'il appartient à cette classe et si tous les problèmes de la classe peuvent être transformés (en respectant certaines contraintes) en ce problème. Ainsi, si on trouve un bon algorithme pour ce problème complet, alors on aura un bon algorithme pour tous les problèmes de la classe.

Définissons plus formellement cette intuition.

Définition 7 (Réduction polynomiale) *Étant donnés deux problèmes de décision $Q \subseteq \Sigma$ et $Q' \subseteq \Sigma'$, une réduction polynomiale est une fonction $f : \Sigma \to \Sigma'$ calculable en temps polynomial telle que pour tout $x \in \Sigma$,*

$$x \in Q \Leftrightarrow f(x) \in Q'.$$

Si une telle réduction existe, on notera $Q \leq_p Q'$.

Observation 8 *Les énoncés suivants sont vrais :*

1. *Transitivité : Si $Q \leq_p Q'$ et $Q' \leq_p Q''$ alors $Q \leq_p Q''$*
2. *Conserve l'appartenance à P : Si $Q \in P$ et $Q' \leq_p Q$, alors $Q' \in P$*
3. *Conserve l'appartenance à NP : Si $Q \in NP$ et $Q' \leq_p Q$, alors $Q' \in NP$*

Définition 9 (Problème NP-complet) *Un problème de décision Q est un problème NP-complet si*

1. *$Q \in NP$*
2. *pour tout problème $Q' \in NP$, alors $Q' \leq_p Q$.*

Ainsi, si Q est un problème NP-complet et que l'on découvre un algorithme polynomial pour Q, alors on aura un algorithme polynomial pour tous les problèmes de NP. Déterminer si un tel algorithme existe est toujours un problème ouvert.

On dit aussi qu'un problème Q est NP-difficile si pour tout problème $Q' \in$ NP, on a que $Q' \leq_p Q$. Le problème Q ne doit pas nécessairement être dans la classe NP pour être NP-difficile.

Pour démontrer qu'un problème $Q \in$ NP est NP-complet, il suffit de montrer qu'il existe une réduction polynomiale d'un problème NP-complet Q' vers ce problème Q. En effet, le dernier point de l'observation 8 nous permet de conclure que puisque tous les problèmes $Q'' \in NP$ sont tels que $Q'' \leq_p Q'$, si $Q' \leq_p Q$, alors $Q'' \leq_p Q$.

Cette observation est bien simple mais il faut connaître un premier problème NP-complet pour l'utiliser. Cook a été le premier à démontrer en 1971 qu'un problème est NP-complet.

Théorème 10 (Cook) *3-SAT est NP-complet*

À partir de ce résultat, il a été possible de démontrer qu'une foule d'autres problèmes étaient aussi NP-complets. Cette fois, il suffisait d'exhiber une rédution polynomiale de 3-SAT vers ces problèmes. En plus de 3-SAT et du problème clique, les deux problèmes de décision suivants sont aussi NP-complets. Cette liste est loin d'être exhaustive.

Définition 11 (3-coloriage)
- *Entrée : un graphe $\mathcal{G}$*
- *Problème : Déterminer s'il est possible de donner une des trois couleurs disponibles à chaque sommet du graphe $\mathcal{G}$ de manière à ce que deux sommets adjacents dans $\mathcal{G}$ ne recoivent pas la même couleur.*

Définition 12 (couverture de sommet)
- *Entrée : une paire $(\mathcal{G}, k)$, où $\mathcal{G}$ est un graphe et $k \in \mathbb{N}$*
- *Problème : Déterminer s'il est possible de choisir k sommets de $\mathcal{G}$ tels que chaque arête de $\mathcal{G}$ est adjacente à au moins un des sommets choisis.*

1.3 Les problèmes paramétrés

On a vu que malgré beaucoup de travaux de recherche, il n'existe présentement pas d'algorithme polynomial qui résout un problème NP-complet. Par contre, la difficulté de certains problèmes semble dépendre d'un paramètre.

Par exemple, il existe un algorithme en $O(2^k \cdot n^c)$ pour le problème de couverture de sommet, où c est une constante et n est la taille du graphe. Cet algorithme n'est pas polynomial par rapport à la taille de l'entrée $(k, \mathcal{G})$. Par contre, si on sait que k est relativement petit, on pourrait être tentés de ne pas tenir compte de k comme un élément de l'entrée et de considérer que l'on a un algorithme polynomial.

Les classes de complexité paramétrées que nous allons présenter sont justement des classes pour différencier ces problèmes.

Définition 13 (Paramétrisation) *Une paramétrisation d'un problème de décision $Q \subseteq \Sigma$ est une fonction $\kappa : \Sigma \to \mathbb{N}$ calculable en temps polynomial.*

Exemple 14 *La fonction $\kappa(\mathcal{G}, k) = k$ est une paramétrisation du problème clique.*

Définition 15 (Problème paramétré) *Un problème paramétré est une paire (Q, κ) où Q est un problème de décision et κ est une paramétrisation de Q.*

Exemple 16 (p-clique) *Le problème suivant est un problème paramétré :*
- *Entrée : $(\mathcal{G}, k)$ où $\mathcal{G}$ est un graphe et où $k \in \mathbb{N}$*
- *Paramètre : $\kappa(\mathcal{G}, k) = k$*
- *Problème : Déterminer si $\mathcal{G}$ contient une clique de taille au moins k*

Exemple 17 (p-k-3-SAT) *Le problème suivant est un problème paramétré :*

- *Entrée : φ une formule booléenne en 3-CNF*
- *Paramètre : $\kappa(\varphi) = k$ où $k \in \mathbb{N}$*
- *Problème : Déterminer s'il existe une assignation des variables de φ avec exactement k variables mises à vrai qui satisfait φ.*

1.3.1 La classe FPT

La classe FPT est la classe des problèmes qui peuvent être résolus en temps polynomial si l'on ne considère pas le paramètre comme un élément de l'entrée.

Définition 18 (Classe FPT) *La classe FPT est la classe des problèmes paramétrés (Q, κ) pour lesquels il existe un algorithme qui prend en entrée une instance x de Q et qui détermine si $x \in Q$ en temps*

$$O(f(\kappa(x)) \cdot |x|^c)$$

où $f : \mathbb{N} \to \mathbb{N}$ est une fonction calculable et c est une constante.

1.3.2 La classe W[1]

Il existe aussi pour les problèmes paramétrés des notions de complétude et de difficulté. Celles-ci sont définies de manière analogue à ces notions pour les classes P, NP et NP-complet en utilisant les FPT-réductions plutôt que les réductions polynomiales.

Définition 19 (FPT-réduction) *Soient (Q, κ) et (Q', κ') deux problèmes paramétrés où $Q \subseteq \Sigma$ et $Q' \subseteq \Sigma'$ sont des problèmes de décision. Une FPT-réduction de (Q, κ) vers (Q', κ') est une fonction $R : \Sigma \to \Sigma'$ telle que*

- *pour tout $x \in \Sigma$,*
$$x \in Q \Leftrightarrow R(x) \in Q',$$
- *il existe un algorithme qui, pour toute entrée $x \in \Sigma$, calcule $R(x)$ en temps*
$$g(\kappa(x)) \cdot p(|x|)$$
où $g : \mathbb{N} \to \mathbb{N}$ est une fonction calculable et où p est un polynôme,
- *il existe une fonction calculable $h : \mathbb{N} \to \mathbb{N}$ telle que pour tout $x \in \Sigma$, on a que*
$$\kappa'(R(x)) \leq h(\kappa(x)).$$

Si une telle réduction existe, on notera $(Q, \kappa) \leq_{FPT} (Q', \kappa')$.

Remarque 20 *Si $(Q, \kappa) \leq_{FPT} (Q', \kappa')$ et $(Q', \kappa') \in FPT$, alors $(Q, \kappa) \in FPT$.*

Nous aurons besoin pour la suite de la classe de complexité W[1]. Cette classe est à FPT ce que NP-complet est à P.

Définition 21 (La classe W[1]) *Le problème paramétré (Q, κ) appartient à la classe W[1] si $(Q, \kappa) \leq_{FPT}$ p-clique .*

Cette définition n'est pas une définition standard pour W[1] mais elle convient à nos besoins.

S'il existe une réduction FPT de p-clique vers un problème paramétré (Q, κ), alors (Q, κ) est W[1]-difficile. De plus, si l'on suppose que FPT$\neq$ W[1], on aura alors que $(Q, \kappa) \notin$ FPT.

1.4 Les classes L et NL

Jusqu'à maintenant, nous ne nous sommes intéressés qu'à des classes de complexité évaluant le *temps* de calcul nécéssaire à la résolution de divers problèmes. Nous voulons maintenant nous attarder à un autre type de classes, les classes L et NL.

La classe L est la classe des problèmes de décision résolubles en *espace* logarithmique, c'est-à-dire pour lesquels il existe un algorithme qui n'utilise que $O(\log(n))$ bits de mémoire, en plus de l'entrée, où n est la taille de l'entrée.

Les problèmes suivants sont des problèmes dans L.

Définition 22 (Existence de cycle impair)
- *Entrée : Un graphe non dirigé $\mathcal{G}$*
- *Problème : Déterminer s'il existe un cycle impair dans $\mathcal{G}$.*

Définition 23 (Connexité non dirigée)
- *Entrée : Un graphe non dirigé $\mathcal{G}$*
- *Problème : Déterminer si le graphe $\mathcal{G}$ est un graphe connexe.*

D'autre part, la classe NL est la classe qui contient l'ensemble des problèmes résolubles par un algorithme *non déterministe* en espace logarithmique.

On dit qu'un algorithme est non déterministe s'il fait des choix aléatoires dans son calcul. Un problème Q sur Σ est résoluble par un algorithme non déterministe A si pour tout $x \in Q$, il existe une séquence de choix aléatoires tels que A accepte x en faisant ces choix. Aussi, pour tout $x \notin Q$, l'algorithme A doit rejeter l'entrée x sur toutes les séquences de choix possibles.

Notons que les problèmes dans NP sont les problèmes qui peuvent être résolus en temps polynomial par un algorithme non déterministe. En effet, un tel algorithme pourrait choisir non déterministement un vecteur de preuve X pour l'entrée x du problème Q et ensuite simuler les étapes de calcul de $A(x, X)$, où A est l'algorithme qui résout Q.

Les problèmes dans NL sont les problèmes de décision qui peuvent être résolus par un algorithme non déterministe qui, tout comme dans L, n'utilise que $O(\log(n))$ bits de mémoire, en plus de l'entrée, où n est la taille de l'entrée.

La classe de complexité NL contient tous les problèmes de L.

À titre d'exemple, l'algorithme 1 qui suit est un algorithme non déterministe qui résout le problème existence de cycle impair. Il serait possible de résoudre ce problème avec un algorithme déteministe, mais nous croyons que cet exemple est parlant. Le principe de l'algorithme est de suivre non déterministement un cycle dans le graphe.

Algorithme 1 : Existence de cycle impair

Entrées : Graphe $G = (V, E)$
Sorties : «ACCEPTER» si l'entrée contient un cycle impair, «REJETER» sinon

```
1  choisir non déterministement une arête e = (a, b) ∈ E;
2  y ← b ; cpt ← 1;
3  tant que cpt ≤ |E| faire
4      si a = y alors
5          si cpt mod 2 = 1 alors
6              retourner ACCEPTER ;
7          sinon
8              retourner REJETER ;
9      Choisir non déterministement une arête e' = (y, c) ∈ E;
10     y ← c ; cpt ← cpt + 1;
11 retourner REJETER;
```

Cet algorithme ne garde en mémoire que les variables *cpt* et y ainsi que le sommet de départ a. Comme celles-ci valent au plus $|E|$, les encoder en mémoire nécessite au plus $3\lceil\log(|E|)\rceil$ bits.

Il est clair que si le graphe en entrée a un cycle impair, il existe une séquence de choix non déterministes qui mène à l'acceptation de l'instance. Il suffit de choisir à chaque étape de la boucle une arête de ce cycle.

Pour voir que tout graphe accepté contient nécessairement un cycle impair, il faut remarquer que si le chemin suivi retourne sur ses pas, alors le nombre d'arêtes visitées reste pair. Aussi, le passage par un cycle pair n'ajouterait lui aussi qu'un nombre pair d'arêtes, d'où l'exactitude de l'algorithme.

Le problème suivant est aussi un problème dans NL.

Définition 24 (s, t-connexité) *Soit un graphe dirigé $\mathcal{G} = (V, E)$ et soient deux sommets s et t de $\mathcal{G}$.*

– *Entrées :* $(\mathcal{G}, s, t)$

– *Problème : Déterminer s'il existe un chemin dans $\mathcal{G}$ allant du sommet s vers le sommet t.*

L'algorithme 2 est un algorithme NL pour s,t-connexité. Ici, le principe est de choisir non déterministement un chemin à partir de s vers la cible t.

Algorithme 2 : s,t-connexité

Entrées : $(\mathcal{G}, s, t)$ où $\mathcal{G} = (V, E)$ est un graphe dirigé et où $s, t \in V$
Sorties : ACCEPTER s'il existe dans $\mathcal{G}$ un chemin de s vers t, REJETER sinon

```
1 cpt ← 1;
2 x ← s ;
3 tant que cpt ≤ |V| faire
4     si x = t alors
5         retourner ACCEPTER;
6     Choisir non déterministement un sommet y tel que (x, y) ∈ E;
7     cpt ← cpt + 1;
8     x ← y;
9 retourner REJETER;
```

Cet algorithme ne garde en mémoire que les variables x et *cpt*. Comme *cpt* est toujours plus petit ou égal à $|V|$, l'encoder en mémoire se fait en au plus $\log(|V|)$ bits. Il en est de même pour la variable x et l'algorithme ne nécéssite que $2\log(|V|)$ bits de mémoire.

L'algorithme s'arrête toujours grâce à la variable *cpt* puisque la taille de V est finie. S'il existe un chemin dans $\mathcal{G}$ entre le sommet s et le sommet t, alors l'algorithme accepte l'entrée lorsque ses choix non déterministes sont les sommets de ce chemin.

Aussi, s'il n'existe pas de tel chemin, alors toutes les séquences de choix non déterministes mènent au rejet de l'instance.

Notons que l'algorithme bien connu de recherche en profondeur pourrait aussi résoudre ce problème. Par contre, l'espace mémoire nécéssaire à cet algorithme est d'au moins $|\mathcal{G}|$. Comparer ces deux algorithmes nous permet de mettre en évidence les différences entre la conception d'un algorithme en espace mémoire réduit par rapport à celle d'un algorithme plus conventionnel.

1.5 NL-complétude

Des notions de complétude existent aussi pour NL.

Définition 25 (Fonction calculable en espace logarithmique) *Une fonction calculable en espace logarithmique est une fonction $f : \Sigma \to \Sigma'$ calculable par un algorithme qui n'utilise*

que $O(\log(n))$ bits de mémoire, où n est la taille de l'entrée. Comme la taille de $f(x)$ peut être plus grande que $O(\log(n))$ bits, l'espace nécéssaire pour écrire les bits en sortie n'est pas comptabilisé.

Le problème Q sur Σ se réduit au problème R sur Σ' en espace logarithmique, et on note $Q \leq_L R$, s'il existe une fonction calculable en espace logarithmique $f : \Sigma \to \Sigma'$ telle que pour tout $x \in \Sigma$, on a que $x \in Q$ si et seulement si $f(x) \in R$.

Exemple 26 *Le problème d'existence de cycle impair se réduit au problème s,t-connexité en espace logarithmique.*

À partir du graphe $\mathcal{G} = (V, E)$, on construit une instance de s,t-connexité de la manière suivante. Les noeuds du graphe dirigé $f(\mathcal{G}) = (N, \overrightarrow{E})$ obtenu en transformant le graphe $\mathcal{G}$ représentent les valeurs possibles pour les variables a, y et cpt lors de l'exécution de l'algorithme 1.

Un arc entre deux noeuds $n_1 = (u, v_1, i)$ et $n_2 = (u, v_2, j)$ indique qu'il est possible que la mémoire de l'algorithme 1 passe de l'état $a = u, y = v_1, cpt = i$ à l'état $a = u, y = v_2, cpt = j$.

Plus formellement, on a que

$$N = \{s, t\} \cup \{(u, v, i) : u, v \in V, i \in \{1, \ldots, |E|\}$$

$$\overrightarrow{E} = \{(s, (u, v, 1)) : (u, v) \in E\} \cup \{((u, v, i), (u, w, i + 1)) : (v, w) \in E\} \cup$$
$$\{((u, u, i), t) : i \bmod 2 = 1\}$$

L'algorithme 3 énumère en espace logarithmique tous les arcs de ce graphe.

Si le graphe $\mathcal{G}$ contient un cycle impair, disons $(v_1, v_2, \ldots, v_{2i+1}, v_1)$ pour un certain $i \in \mathbb{N}$, alors le graphe $f(\mathcal{G})$ en sortie contient le chemin

$$(s, (v_1, v_2, 1), (v_1, v_3, 2), \ldots, (v_1, v_{2i+1}, 2i), (v_1, v_1, 2i + 1), t)$$

et est donc accepté par l'algorithme 2 pour s,t-connexité.

Inversement, s'il existe dans $f(\mathcal{G})$ un chemin entre les sommets s et t, disons

$$(s, (u, v_1, 1), (u, v_2, 2), \ldots, (u, v_i, i), t),$$

c'est que $(u, v_1) \in E$. De plus, s'il y a un arc $((u, v_1, 1), (u, v_2, 2))$ dans $f(\mathcal{G})$, c'est que $(v_1, v_2) \in E$. Il en est de même pour toutes les paires de sommets v_k, v_{k+1} pour $k < i$. Finalement l'arc $((u, v_i, i), t)$ implique que $u = v_i$ et que i est impair. Il y a donc un chemin $(u, v_1, v_2, \ldots, v_{i-1}, u)$ de longueur impaire dans $\mathcal{G}$.

Algorithme 3 : Transforme «existence de cycle impair» en «s,t-connexité»

Entrées : Un graphe non-dirigé $\mathcal{G} = (V, E)$

Sorties : Un graphe dirigé $f(\mathcal{G}) = (N, \overrightarrow{E})$ et des sommets $s, t \in N$ identifiés

```
1  pour chaque i ∈ {1, ..., |E|} faire
2      pour chaque u ∈ V faire
3          pour chaque v ∈ V faire
4              si i=1 et (u, v) ∈ E alors
5                  écrire (s, (u, v, 1))
6              si i est impair et u=v alors
7                  écrire ((u, v, i), t)
8              si i < |E| alors
9                  pour chaque w ∈ V tel que (v, w) ∈ E faire
10                     écrire ((u, v, i), (u, w, i + 1))
```

Cette réduction est donnée comme un exemple seulement puisque le problème d'existence de cycle impair est un problème qui appartient à la classe de complexité L, une classe plus simple que NL.

Définition 27 *Un problème A sur Σ est NL-complet sous les réductions en espace logarithmique si*

- $A \in NL$
- *pour tout $B \in NL$, $B \leq_L A$.*

Pour simplifier la suite du texte, nous dirons simplement NL-complet.

Théorème 28 *Le problème s,t-connexité est NL-complet.*

Pour montrer qu'un problème est NL-complet, il suffira donc de trouver une réduction en espace logarithmique de s,t-connexité vers ce problème et il faudra démontrer que le problème est dans NL.

Théorème 29 $NL \subseteq P$

En effet, il suffit de remarquer que le problème NL-complet s,t-connexité est résoluble en temps polynomial par un algorithme de recherche en profondeur.

1.6 Problèmes de satisfaction de contraintes

Les problèmes de satisfaction de contraintes (CSP) sont des problèmes très fréquents en informatique. Nous allons les définir formellement.

Définition 30 (Relation) *Une relation d'arité k sur un domaine D est un ensemble $R \subseteq D^k$, c'est-à-dire un ensemble de k-tuples d'éléments de D.*

Définition 31 (Problème de satisfaction de contraintes (CSP(Γ)) *Soit Γ un ensemble de relations sur un domaine D. Une instance de CSP(Γ) est un triplet $P = (V, D, C)$ avec*

- *$V = \{v_1, \ldots, v_n\}$ un ensemble de variables*
- *$D = \{d_1, \ldots, d_m\}$ le domaine des valeurs que peuvent prendre les variables*
- *$C = \{C_1, \ldots, C_q\}$ un ensemble de contraintes avec $C_i = (\bar{t}, R)$ où $\bar{t}$ est un k-tuple d'éléments de V et $R \in \Gamma$ est une relation d'arité k.*

Une solution de l'instance P est une fonction $f : V \to D$ telle que $\forall (\bar{t}, R) \in C$ avec $\bar{t} = (t_1, \ldots, t_k)$,

$$(f(t_1), \ldots, f(t_k)) \in R.$$

Étant donnée une instance P de CSP(Γ), nous nous intéressons au problème de décider s'il existe une telle solution.

Exemple 32 (3-SAT est un CSP) *L'instance de 3-SAT*

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$$

est le CSP $P = (V, D, C)$ avec

- *$V = \{x_1, x_2, x_3, x_4\}$*
- *$D = \{0, 1\}$*
- *$C = \{C_1, C_2, C_3, C_4\}$ où*
  - *$C_1 = ((x_1, x_2, x_3), R_0)$ $\quad R_0 = \{0,1\}^3 \backslash \{0,0,0\}$*
  - *$C_2 = ((x_3, x_2, x_4), R_1)$ $\quad R_1 = \{0,1\}^3 \backslash \{1,0,0\}$*
  - *$C_3 = ((x_1, x_4, x_3), R_2)$ $\quad R_2 = \{0,1\}^3 \backslash \{1,1,0\}$*
  - *$C_4 = ((x_1, x_2, x_4), R_3)$ $\quad R_3 = \{0,1\}^3 \backslash \{1,1,1\}$*

La fonction $f : V \to D$ telle que $f(x_1) = 1, f(x_2) = 1, f(x_3) = 1$ et $f(x_4) = 0$ est une solution.

L'exemple 32 nous montre que les CSP sont en général NP-difficiles. En effet, pour $\Gamma = \{R_0, R_1, R_2, R_3\}$, l'ensemble des instances de CSP(Γ) est l'ensemble des instances du problème 3-SAT, un problème NP-complet.

Définition 33 *On dira que Γ est*

- *tractable si pour tout $\Gamma' \subseteq \Gamma$, $CSP(\Gamma')$ est dans P*
- *NP-complet s'il existe $\Gamma' \subseteq \Gamma$ tel que $CSP(\Gamma')$ est NP-complet.*

1.7 Problèmes d'homomorphisme

La définition 31 ne sera pas la seule à être utilisée dans ce texte pour définir les CSP. En effet, on peut voir le problème sous un autre angle. Ce nouveau point de vue nous permettra de généraliser le problème des CSP et nous donnera de nouveaux outils pour attaquer l'étude de sa complexité.

Définition 34 (Vocabulaire) *Un vocabulaire τ est un ensemble de symboles de relations R_i d'arités $\rho(R_i)$ spécifiées.*

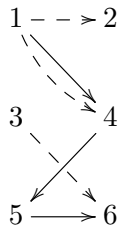
Définition 35 (τ-structure) *Étant donné un vocabulaire $\tau = \{R_1, ..., R_n\}$ et une fonction d'arité ρ, une τ-structure $\mathcal{A} = (\boldsymbol{A}, R_1^A, ..., R_n^A)$ contient un ensemble fini $\boldsymbol{A}$ appelé* **univers** *de $\mathcal{A}$ et pour chaque symbole de relation R_i de τ, un sous-ensemble $R_i^A \subseteq \boldsymbol{A}^{\rho(R_i)}$.*

Pour un vocabulaire τ, nous noterons l'ensemble des τ-structures $STR[\tau]$.

Exemple 36 *Soit le vocabulaire $\tau = \{E\}$ avec $\rho(E) = 2$. L'ensemble des τ-structures est l'ensemble des graphes dirigés.*

En effet, la τ-structure $\mathcal{G}$ a comme univers l'ensemble $\mathbf{G} = V(\mathcal{G})$ et l'interprétation de E dans $\mathcal{G}$ est $E^G = E(\mathcal{G})$, l'ensemble des arêtes de $\mathcal{G}$.

Exemple 37 *Soit $\tau = \{R_1, R_2\}$ avec $\rho(R_1) = 2$ et $\rho(R_2) = 2$. Le multigraphe suivant est la τ-structure $\mathcal{M} = \{\boldsymbol{M}, R_1^M, R_2^M\}$ avec $\boldsymbol{M} = \{1, 2, 3, 4, 5, 6\}$ et $R_1^M = \{(1, 4), (4, 5), (5, 6)\}$, $R_2^M = \{(1, 2), (1, 4), (3, 6)\}$.*



Nous pouvons maintenant définir plus formellement la notion d'homomorphisme présentée plus tôt.

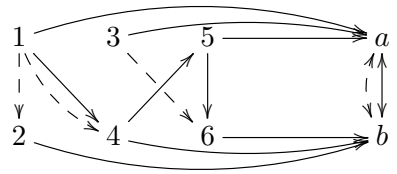
Définition 38 (Homomorphisme) *Soit τ un vocabulaire et soit $\mathcal{A}, \mathcal{B}$ des τ-structures. Un homomorphisme entre $\mathcal{A}$ et $\mathcal{B}$ est une fonction $h : \boldsymbol{A} \to \boldsymbol{B}$ telle que $\forall R \in \tau$, $\forall (a_1, a_2, ..., a_{\rho(R)}) \in R^A$, $(h(a_1), h(a_2), ..., h(a_{\rho(R)})) \in R^B$.*

Si $\overline{a} = (a_1, ..., a_{\rho(R)}) \in R^A$, nous noterons $h(\overline{a}) = (h(a_1), ..., h(a_{\rho(R)}))$. Aussi, s'il existe un homomorphisme entre les τ-structures $\mathcal{A}$ et $\mathcal{B}$, nous écrirons $\mathcal{A} \to \mathcal{B}$ et nous dirons que $\mathcal{A}$ est *homomorphe* à $\mathcal{B}$.

Exemple 39 *La τ-structure $\mathcal{M}$ de l'exemple précédent est homomorphe à la τ-structure $\mathcal{N} = (\boldsymbol{N}, R_1^N, R_2^N)$, $\boldsymbol{N} = \{a, b\}, R_1^N = R_2^N = \{(a, b), (b, a)\}$. En effet, la fonction $h : \boldsymbol{M} \to \boldsymbol{N}$ telle que*

$$h(n) = \begin{cases} a & \text{si } n \text{ est impair} \\ b & \text{si } n \text{ est pair} \end{cases}$$

est un homomorphisme.



Remarque 40 *Soit $\mathcal{A}, \mathcal{B}, \mathcal{C}$ des τ-structures. Si $\mathcal{A} \to \mathcal{B}$ et $\mathcal{B} \to \mathcal{C}$, alors $\mathcal{A} \to \mathcal{C}$.*

En effet, soit $h_1 : \mathbf{A} \to \mathbf{B}$ et $h_2 : \mathbf{B} \to \mathbf{C}$ des homomorphismes. La fonction $h_2 \circ h_1 : \mathbf{A} \to \mathbf{C}$ est elle aussi un homomorphisme. Pour toute relation $R \in \tau$, pour tout tuple $\bar{t} \in R^A$, alors $h_1(\bar{t}) \in R^B$ car h_1 est un homomorphisme. De plus, comme h_2 est aussi un homomorphisme et que $h_1(\bar{t}) \in R^B$, on a $h_2(h_1(\bar{t})) \in R^C$.

Définition 41 (Isomorphisme) *Les structures $\mathcal{A}$ et $\mathcal{B}$ sont isomorphes s'il existe un homomorphisme bijectif $h : \mathcal{A} \to \mathcal{B}$ dont l'inverse $h^{-1} : \mathcal{B} \to \mathcal{A}$ est aussi un homomorphisme.*

Définition 42 (Core) *On dit que la τ-structure $\mathcal{A}$ est un core si tout homomorphisme $h : \mathcal{A} \to \mathcal{A}$ est un isomorphisme.*

Exemple 43 *Pour tout nombre naturel i, le graphe K_i est le graphe composé de i sommets tous reliés entre eux par une arête. Ces graphes sont des cores.*

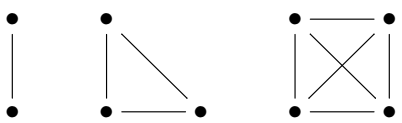


FIGURE 1.2 – Les graphes K_2, K_3 et K_4.

On dira qu'un core $\mathcal{A}'$ est ***le*** core de la τ-structure $\mathcal{A}$ si $\mathcal{A} \to \mathcal{A}'$ et que $\mathcal{A}' \to \mathcal{A}$. Remarquez qu'une τ-structure n'a, à isomorphisme près, qu'un seul core.

En effet, si $\mathcal{B}$ et $\mathcal{C}$ sont des cores et que $\mathcal{B} \to \mathcal{A}$, $\mathcal{A} \to \mathcal{B}$, $\mathcal{C} \to \mathcal{A}$ et $\mathcal{A} \to \mathcal{C}$, alors il existe deux homomorphismes $\mathcal{B} \to \mathcal{C}$ et $\mathcal{C} \to \mathcal{B}$.

De plus, si $f : \mathcal{B} \to \mathcal{C}$ est un homomorphisme qui n'est pas surjectif, alors l'homomorphisme $f' : \mathcal{C} \to \mathcal{C}$ tel que $f' = f \circ h \circ g$ pour des homomorphismes $g : \mathcal{C} \to \mathcal{A}$, $h : \mathcal{A} \to \mathcal{B}$ n'est pas surjectif, une contradiction.

Aussi, supposons que cette fonction n'est pas injective et qu'il existe $x, y \in \mathbf{B}, x \neq y$ tels que $f(x) = f(y)$. Soit un homomorphisme $g : \mathcal{C} \to \mathcal{B}$ tel que $g(a) = x$ et $g(b) = y$ pour $x, y \in \mathbf{C}$. L'homomorphisme $f' : \mathcal{C} \to \mathcal{C}$ avec $f' = f \circ g$ est tel que $f'(a) = f'(b)$ et n'est pas injectif, une contradiction.

On a donc que tout homomorphisme $f : \mathcal{B} \to \mathcal{C}$ est bijectif. On peut utiliser le même raisonnement pour conclure que tout homomorphisme $g : \mathcal{C} \to \mathcal{B}$ l'est aussi.

Comme $\mathcal{B}$ est un core, la fonction $h : \mathcal{B} \to \mathcal{B}$ telle que $h(x) = g(f(x))$ est un isomorphisme et donc h^{-1} est un homomorphisme.

Pour tout $R \in \tau$, pour tout $\bar{a} = (a_1, \ldots, a_k) \in R^{\mathbf{C}}$, comme f est bijective, il existe $x_1, \ldots, x_k \in \mathbf{B}$ tels que $f(x_1) = a_1, \ldots, f(x_k) = a_k$. Aussi, comme g est un homomorphisme, $g(\bar{a}) \in R^{\mathbf{B}}$. Or,

$$g(\bar{a}) = (g(a_1), \ldots, g(a_k)) = (g(f(x_1)), \ldots, g(f(x_k))) = (h(x_1), \ldots, h(x_k)).$$

Comme $(h(x_1), \ldots, h(x_k)) \in R^{\mathbf{B}}$ et comme h^{-1} est un homomorphisme,

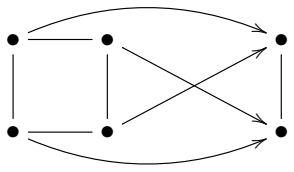
$$(h^{-1}(h(x_1)), \ldots, h^{-1}(h(x_k))) = (x_1, \ldots, x_k) \in R^{\mathbf{B}}.$$

On a bien que f^{-1} est un homomorphisme. $\mathcal{B}$ et $\mathcal{C}$ sont donc isomorphes.

Remarque 44 *Le core de tout graphe biparti est K_2.*

En effet, soit $\mathcal{G}$ un graphe biparti, c'est-à-dire un graphe pour lequel il existe une partition $V_1, V_2 \subseteq V(\mathcal{G})$ telle que $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V(\mathcal{G})$ et telle que $\forall u, v \in V(\mathcal{G})$, si u et v appartiennent au même ensemble V_i, alors $(u, v) \notin E(\mathcal{G})$. La fonction $h : \mathcal{G} \to K_2$ telle que $h(u) = 1$ si $u \in V_1$ et $h(u) = 2$ si $u \in V_2$ est un homomorphisme.

Exemple 45 *Le core du cycle à 4 sommets est K_2*



Définition 46 (Équivalence homomorphique) *Soit $\mathcal{A}, \mathcal{B}$ des τ-structures. On dira que $\mathcal{A}$ et $\mathcal{B}$ sont homomorphiquement équivalentes si $\mathcal{A} \to \mathcal{B}$ et $\mathcal{B} \to \mathcal{A}$.*

Définition 47 (Problème d'homomorphisme $HOM(-, \mathcal{B})$) *Soit τ un vocabulaire et $\mathcal{B} \in STR[\tau]$. Le problème d'homomorphisme $HOM(-, \mathcal{B})$ est le problème de décider, pour l'instance $\mathcal{A} \in STR[\tau]$ du problème, si $\mathcal{A} \to \mathcal{B}$.*

Si $\mathcal{H}$ est un graphe, le problème $HOM(-, \mathcal{H})$ est aussi appelé $\mathcal{H}$-coloriage.

Remarquons que les instances du problème $HOM(-, \mathcal{B})$ sont des instances du problème $CSP(\Gamma)$, pour $\Gamma = \{R_i^{\mathbf{B}} : R_i \in \tau\}$, un ensemble de relations sur le domaine $\mathbf{B}$.

En effet, soit $\mathcal{A} \in STR[\tau]$ une instance de $HOM(-, \mathcal{B})$. On a que $\mathcal{A} \to \mathcal{B}$ si et seulement si l'instance $P = (V, D, C)$ de $CSP(\Gamma)$ avec $V = \mathbf{A}$, $D = \mathbf{B}$ et $C = \{(\bar{a}, R_i^{\mathbf{B}}) : \bar{a} \in R_i^{\mathbf{A}}, R_i \in \tau\}$ a une solution.

Exemple 48 *Soit $\tau = \{R_1, R_2\}$ un vocabulaire comprenant deux relations d'arité deux et $\mathcal{N}$ et $\mathcal{M}$ les τ-structures de l'exemple 39. On a que $\mathcal{M} \to \mathcal{N}$ si et seulement si le CSP $P = (V, D, C)$ avec*

- $V = \{1, 2, 3, 4, 5, 6\}$
- $D = \{a, b\}$
- $C = \{C_1, C_2, C_3, C_4, C_5, C_6\}$ *où*
  - $C_1 = \{(1, 4), R_1\}$ $\quad R_1 = \{(a, b), (b, a)\}$
  - $C_2 = \{(4, 5), R_1\}$
  - $C_3 = \{(5, 6), R_1\}$
  - $C_4 = \{(1, 2), R_2\}$ $\quad R_2 = \{(a, b), (b, a)\}$
  - $C_5 = \{(1, 4), R_2\}$
  - $C_6 = \{(3, 6), R_2\}$

a une solution.

Aussi, pour tout ensemble de relations Γ sur un domaine D, le problème de satisfaction de contraintes CSP(Γ) peut être exprimé sous la forme du problème d'homomorphisme $HOM(-, \mathcal{B})$ pour une certaine τ-structure $\mathcal{B}$.

En effet, on pose le vocabulaire τ comme l'ensemble $\{R_1, \ldots, R_u\}$ qui contient un symbole de relation pour chaque élément de Γ avec la même arité que la relation de Γ correspondante.

On définit $\mathcal{B}$ comme la τ-structure avec l'univers $\mathbf{B} = D$ et $\forall i \in \{1, \ldots, u\}$, les tuples dans R_i^B sont les mêmes que ceux dans la relation R_i de Γ.

Pour toute instance $P = (V, D, C)$ de CSP(Γ), on considère la τ-structure $\mathcal{A}_\mathcal{P}$, où $\mathbf{A}_\mathcal{P} = V$. Aussi, pour tout $(t, R_i) \in C$, on ajoute le tuple t à $R_i^{A_\mathcal{P}}$.

Il s'ensuit que $\mathcal{A}_\mathcal{P} \to \mathcal{B}$ si et seulement s'il existe une solution à l'instance $P = (V, D, C)$ de CSP(Γ).

Exemple 49 *Soit $\tau = \{R_0, R_1, R_2, R_3\}$ un vocabulaire contenant 4 relations d'arité 3. L'exemple 32 peut s'écrire sous la forme du problème d'homomorphisme $HOM(\mathcal{A}_\varphi, \mathcal{B})$ où $\mathcal{A}_\varphi = \{\boldsymbol{A}, R_0^A, R_1^A, R_2^A, R_3^A\}$ et $\mathcal{B} = \{\boldsymbol{B}, R_0^B, R_1^B, R_2^B, R_3^B\}$ sont des τ-structures avec*

- $\boldsymbol{A} = \{x_1, x_2, x_3, x_4\}$ $\quad R_0^A = \{(x_1, x_2, x_3)\}, R_1^A = \{(x_3, x_2, x_4)\}$
  $\quad R_2^A = \{(x_1, x_4, x_3)\}, R_3^A = \{(x_1, x_2, x_4)\}$

- $\boldsymbol{B} = \{0, 1\}$ $\quad R_0^B = \{0,1\}^3 \backslash \{0,0,0\}, R_1^B = \{0,1\}^3 \backslash \{1,0,0\}$
  $\quad R_2^B = \{0,1\}^3 \backslash \{1,1,0\}, R_3^B = \{0,1\}^3 \backslash \{1,1,1\}$

La fonction $f : \boldsymbol{A} \to \boldsymbol{B}$ telle que $f(x_1) = 1, f(x_2) = 1, f(x_3) = 1$ et $f(x_4) = 0$ est un homomorphisme car pour toute relation $R \in \tau$, pour tout $\bar{t} \in R^A$, $f(\bar{t}) \in R^B$.

On peut dire que le problème 3-SAT est équivalent à décider s'il existe un homomorphisme entre un certain $\mathcal{A}_\varphi$ et la structure $\mathcal{B}$ de l'exemple précédent. 3-SAT peut donc être exprimé par $HOM(-, \mathcal{B})$, tout aussi bien que par CSP(Γ).

Voir les CSP sous forme de problème d'homomorphisme nous amène aussi à considérer les problèmes suivants.

Définition 50 (Problème d'homomorphisme $HOM(-,-)$)

- *Entrée : $(\mathcal{A}, \mathcal{B})$ où $\mathcal{A}, \mathcal{B} \in STR[\tau]$.*
- *Problème : Déterminer si $\mathcal{A} \to \mathcal{B}$.*

Définition 51 (Problème d'homomorphisme $HOM(C,-)$) *Soit C un ensemble de τ-structures.*

- *Entrée : $(\mathcal{A}, \mathcal{B})$ où $\mathcal{A} \in C \subseteq STR[\tau]$ et $\mathcal{B} \in STR[\tau]$.*
- *Problème : Déterminer si $\mathcal{A} \to \mathcal{B}$.*

1.8 Définitions sur les graphes

Nous aurons besoin pour établir nos résultats de certaines notions de la théorie des graphes.

1.8.1 Mineurs

Définition 52 (Contraction d'arête) *Soit un graphe $\mathcal{G}$ et $e = (a, b)$ une arête de $\mathcal{G}$. Le graphe $\mathcal{G}'$ obtenu par la contraction de l'arête e est le graphe qui a les sommets $V(\mathcal{G}') = V(\mathcal{G})\backslash\{a, b\} \cup \{v_e\}$ et les arêtes*

$$E(\mathcal{G}') = E(\mathcal{G}) \cap (V(\mathcal{G}') \times V(\mathcal{G}')) \cup \{(u, v_e) : (u, a) \in E(\mathcal{G}), (u, b) \in E(\mathcal{G})\}.$$

La contraction d'une arête $e = (a, b)$ d'un graphe consiste donc à remplacer cette arête par un sommet et à connecter ce sommet aux voisins de a et de b. La figure 1.3 est un exemple.

(a) Le graphe $\mathcal{G}$ (b) Le graphe $\mathcal{G}'$

FIGURE 1.3 – Le graphe $\mathcal{G}'$ a été obtenu du graphe $\mathcal{G}$ par la contraction de l'arête (b, d)

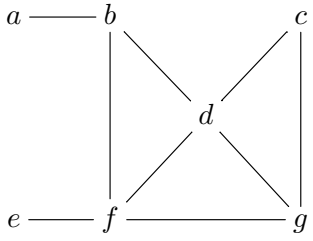
Définition 53 (Mineur) *On dira que le graphe $\mathcal{H}$ est un mineur du graphe $\mathcal{G}$ s'il existe un sous graphe $\mathcal{G}'$ de $\mathcal{G}$ tel qu'il est possible d'obtenir le graphe $\mathcal{H}$ en contractant des arêtes de $\mathcal{G}'$.*

Définition 54 (Fonction mineur) *Une fonction mineur entre deux graphes $\mathcal{H}$ et $\mathcal{G}$ est une fonction $\mu : V(\mathcal{H}) \to 2^{|V(\mathcal{G})|}$ telle que*

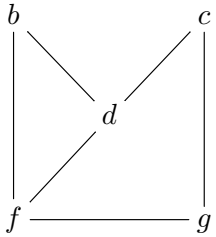
- *pour tout $v \in V(\mathcal{H})$, l'ensemble $\mu(v)$ n'est pas vide et est connexe dans $\mathcal{G}$*
- *pour tout $u, v \in V(\mathcal{H})$ où $v \neq w$, on a que $\mu(v) \cap \mu(w) = \emptyset$*
- *pour tout $(u, v) \in E(\mathcal{H})$, il existe $u' \in \mu(u)$ et $v' \in \mu(v)$ tels que $(u', v') \in E(\mathcal{G})$.*

Remarque 55 *Il existe une fonction mineur $\mu : V(\mathcal{H}) \to 2^{|V(\mathcal{G})|} \Leftrightarrow \mathcal{H}$ est un mineur de $\mathcal{G}$.*

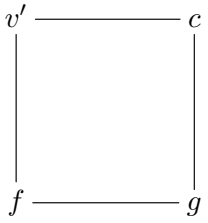
La figure 1.4 est un exemple de toutes ces notions. Le principe de mineur est très important. En effet, on peut déduire plusieurs propriétés d'un graphe grâce à ses mineurs. Nous verrons plus loin en quoi ceux-ci sont liés à une caractérisque qui nous intéresse, la largeur d'arbre.



(a) Le graphe $\mathcal{G}$



(b) Un sous-graphe de $\mathcal{G}$



(c) Le graphe obtenu en contractant l'arête (b, d)

$$\begin{aligned} \mu : V(C_4) &\to 2^{|V(\mathcal{G})|} \\ v' &\mapsto \{b, d, a\} \\ f &\mapsto \{e, f\} \\ g &\mapsto \{g\} \\ c &\mapsto \{c\} \end{aligned}$$

(d) Une fonction mineur du cycle à 4 sommets vers $\mathcal{G}$

FIGURE 1.4 – Le cycle à 4 sommets est un mineur du graphe $\mathcal{G}$

1.8.2 Largeur d'arbre

La largeur d'arbre d'un graphe est un nombre naturel positif qui est une mesure de la distance entre ce graphe et un arbre. Par exemple, tous les arbres ont une largeur d'arbre 1 et un graphe complet à n sommets a une largeur d'arbre $n - 1$.

Bien qu'elle soit d'abord conçue pour s'appliquer aux graphes, la largeur d'arbre peut aussi être définie sur des τ-structures générales. Pour pouvoir la calculer, il faut comprendre ce qu'est une décomposition en arbre.

Définition 56 (Décomposition en arbre) *Soit $\mathcal{A}$ une τ-structure. Une décomposition en arbre de $\mathcal{A}$ est une paire $(\mathcal{T}, \beta)$ où $\mathcal{T}$ est un arbre et $\beta : \boldsymbol{T} \to 2^{|\boldsymbol{A}|}$ est une fonction des sommets de $\mathcal{T}$ vers des sous-ensembles de $\boldsymbol{A}$ telle que*

1. *pour tout $a \in \boldsymbol{A}$, l'ensemble $\{t \in \boldsymbol{T} : a \in \beta(t)\} \neq \emptyset$ et est connexe dans $\mathcal{T}$*
2. *pour tout $R \in \tau$, pour tout $(a_1, \ldots, a_k) \in R^{\boldsymbol{A}}$, il existe un $t \in \boldsymbol{T}$ tel que*

$$\{a_1, \ldots, a_k\} \in \beta(t).$$

La largeur d'une décomposition en arbre est $\max\{|\beta(t)| : t \in \boldsymbol{T}\} - 1$.

On définit la largeur d'arbre d'une τ-structure $\mathcal{A}$ comme le plus petit k tel que $\mathcal{A}$ a une décomposition en arbre de largeur k et on note $tw(\mathcal{A}) = k$. Les figures 1.5 et 1.6 sont des exemples.

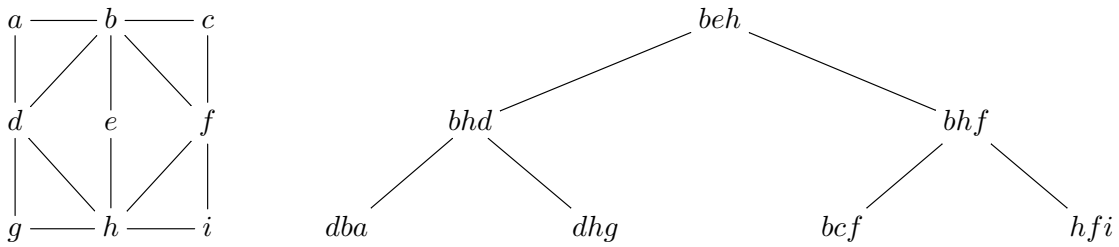


FIGURE 1.5 – Un graphe et une décomposition en arbre de largeur 2

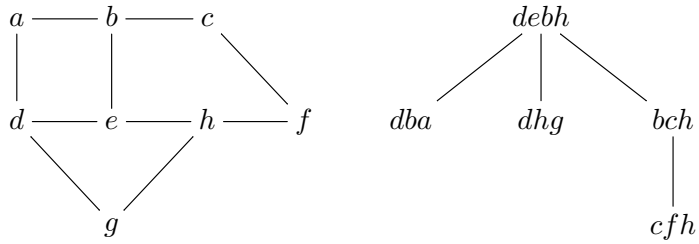


FIGURE 1.6 – Un graphe et une décomposition en arbre de largeur 3

Remarque 57 *Soit une τ-structure $\mathcal{A}$ et soit $\mathcal{A}'$ le core de $\mathcal{A}$. Alors $tw(\mathcal{A}) \geq tw(\mathcal{A}')$.*

Démonstration : Il suffit de remarquer que toute décomposition en arbre $(\mathcal{T}, \beta)$ de $\mathcal{A}$ est aussi une décomposition en arbre de $\mathcal{A}'$.

En effet, pour tout $a \in \mathbf{A}'$, comme $\mathcal{A}'$ est le core (et donc une sous structure) de $\mathcal{A}$, on a que $a \in \mathbf{A}$ et que l'ensemble $\{t \in \mathbf{T} : a \in \beta(t)\}$ n'est pas vide et est connexe dans $\mathcal{T}$.

Aussi, pour tout $R \in \tau$, pour tout $(a_1, \ldots, a_n) \in R^{\mathbf{A'}}$, on a que $(a_1, \ldots, a_n) \in R^{\mathbf{A}}$ et donc que $\{a_1, \ldots, a_n\} \in \beta(t)$ pour un certain $t \in V(\mathcal{T})$.

On peut donc conclure que si $tw(\mathcal{A}) = k$, c'est qu'il existe une décomposition en arbre de largeur k de $\mathcal{A}$. Il existe alors une décomposition en arbre de $\mathcal{A}'$ de largeur k et $tw(\mathcal{A}') \leq k = tw(\mathcal{A})$. □

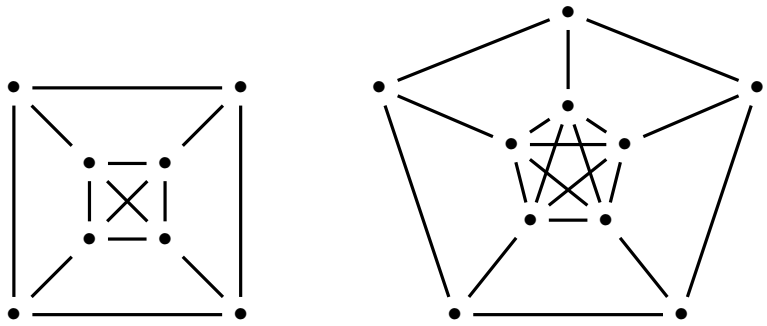
Définition 58 (Largeur d'arbre bornée modulo équivalence homomorphique) *Un ensemble C de τ-structures est de largeur d'arbre bornée modulo équivalence homomorphique s'il*

existe un nombre k tel que pour toute τ-structure $\mathcal{A} \in C$, il existe une τ-structure $\mathcal{A}'$ de largeur d'arbre inférieure ou égale à k telle que $\mathcal{A} \to \mathcal{A}'$ et $\mathcal{A}' \to \mathcal{A}$.

Par exemple, l'ensemble des cycles $C = \{C_2, C_3, \ldots\}$ a une largeur d'arbre bornée modulo équivalence homomorphique car tous les cycles ont une largeur d'arbre de 2.

Aussi, l'ensemble de graphes $D = \{\mathcal{G}_3, \mathcal{G}_4, \mathcal{G}_5, \ldots\}$ où $V(\mathcal{G}_i) = V(C_i) \cup V(K_i)$ et $E(\mathcal{G}_i) = E(C_i) \cup E(K_i) \cup \{(v_j, u_j) : 1 \leq j \leq i\}$ dessiné à la figure 1.7 n'a pas une largeur d'arbre bornée modulo équivalence homomorphique.

En effet, le core du graphe $\mathcal{G}_i$ est K_i. Le graphe K_i est aussi le core de tout graphe $\mathcal{G}$ homomorphiquement équivalent à $\mathcal{G}_i$. Par la remarque 57, on a que $tw(\mathcal{G}) \geq tw(K_i) = i - 1$. Ainsi, il est impossible de trouver un nombre k tel que tous les graphes de D sont homomorphiquement équivalent à un graphe de largeur d'arbre au plus k.



(a) $\mathcal{G}_4$ (b) $\mathcal{G}_5$

FIGURE 1.7 – Exemple d'un ensemble de graphes qui n'a pas une largeur d'arbre bornée modulo équivalnce homomorphique

1.8.3 Largeur de chemin

La largeur de chemin est définie de manière analogue à la largeur d'arbre. Nous la définirons ici sur les graphes mais cette définition peut être étendue aux τ-structures.

Définition 59 (Décomposition en chemin) *Soit G un graphe. Une décomposition en chemin de G est une suite d'ensembles $P_1, \ldots, P_m$, où $P_i \subseteq V(G)$ pour tout $i \in \{1, \ldots, m\}$ et telle que*

1. *pour tout $v \in V(G)$, il existe $i \in \{1, \ldots, m\}$ tel que $v \in P_i$*
2. *pour tout $v \in V(G)$, s'il existe $i, j \in \{1, \ldots, m\}$ tels que $v \in P_i$ et $v \in P_j$, alors pour tout l tel que $i \leq l \leq j$, on a que $v \in P_l$*
3. *pour tout $(u, v) \in E(G)$, il existe un $i \in \{1, \ldots, m\}$ tel que $u \in P_i$ et $v \in P_i$.*

La largeur d'une décomposition en chemin est $\max\{|P_i| : i \in \{1, \ldots, m\}\} - 1$.

On définit la largeur de chemin d'un graphe G comme le plus petit k tel que G a une décomposition en chemin de largeur k et on note $pw(\mathcal{A}) = k$.

On remarque facilement qu'une décomposition en chemin est une décomposition en arbre mais que l'inverse n'est pas toujours vrai. Pour cette raison, on a que pour tout graphe G, $pw(G) \geq tw(G)$.

1.8.4 Largeur de bande

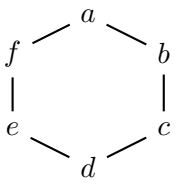
Définition 60 (Largeur de bande (*Bandwidth*)) *Un* ordre (layout) *des sommets d'un graphe G est une fonction bijective $\lambda : V(G) \to \{1, \ldots, |V(G)|\}$. La taille d'un ordre λ est la distance maximale qui sépare deux sommets adjacents, c'est-à-dire*

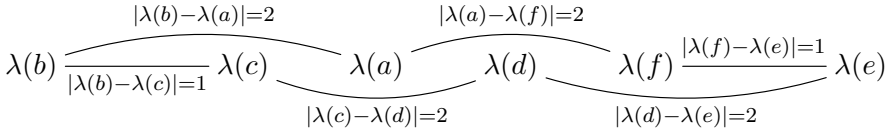
$$max_{(u,v)\in E(G)}|\lambda(u) - \lambda(v)|.$$

La largeur de bande du graphe G, notée $bw(G)$, est la taille minimale de tous les ordres de ses sommets.

Il faut comprendre que la fonction d'ordre λ est comme une position qu'on donne à chaque sommet du graphe. Avec cette façon de voir, la distance entre deux sommets adjacents $|\lambda(u) - \lambda(v)|$ est le nombre de sommets qui sépare u et v dans cet ordre.

Par exemple, le cycle à 6 sommets a une largeur de bande 2. En effet, l'ordre b, c, a, d, f, e est de taille 2.





Sur ce graphe, l'ordre a, b, c, d, e, f n'est pas minimal car il est de taille 5. En effet, on aurait que $|\lambda(a) - \lambda(f)| = |1 - 6| = 5$.

Remarque 61 *La largeur de chemin d'un graphe est plus petite ou égale à sa largeur de bande.*

En effet, étant donné un graphe G et un ordre $\lambda = v_1, v_2, \ldots, v_n$ sur ses sommets, on peut construire une décomposition en chemin $P = P_1, P_2, \ldots, P_n$ en posant

$$P_i = v_i \cup \{v_j : \exists k \leq i \text{ t.q } (v_k, v_j) \in E(G) \text{ et } j > i\}$$

pour tout i allant de 1 jusqu'à n.

Autrement dit, chaque ensemble P_i contient le sommet à la i-ième position de l'ordre ainsi que tous les sommets ayant une position plus grande que i qui sont adjacents au sommet en position i ou à un autre sommet de position inférieure à i.

Par exemple, si le sommet v_5 est adjacent à v_7 et que v_3 est adjacent à v_6, alors l'ensemble P_5 contiendra v_5, v_6 et v_7. Pour l'exemple précédent avec le cycle de taille 6, on obtiendrait la décomposition en chemin suivante :

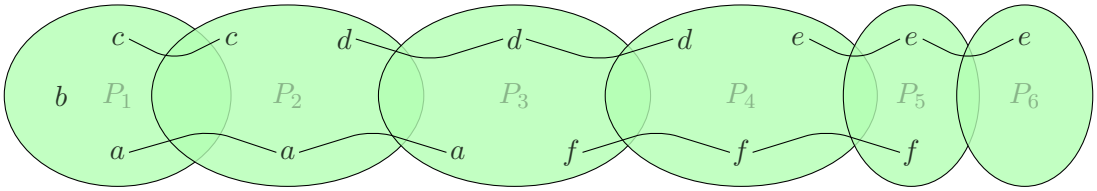


FIGURE 1.9 – Décomposition en chemin du cycle à six sommets à partir de l'ordre b, c, a, d, f, e. Les ensembles P_5 et P_6 ne sont pas nécessaires, mais ils n'empêchent pas la décomposition en chemin obtenue d'être valide.

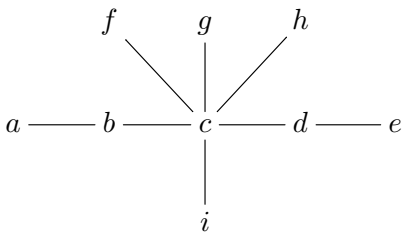
On obtient bien une décomposition en chemin de G :

- Chaque sommet de G appartient à au moins un ensemble P_i.
- Pour chaque arête (u, v) de G, il y a au moins un ensemble P_i qui contient à la fois u et v. En effet, supposons sans perte de généralité que $i = \lambda(u) < \lambda(v)$. Par construction, u et v appartiennent tous deux à l'ensemble P_i.
- Si un sommet u est dans P_i et dans P_j, alors pour tout l avec $i \leq l \leq j$, on a que $u \in P_l$. Soit un sommet u tel que $\lambda(u) = j$. Par construction, $u \in P_j$ et u ne peut appartenir à aucun ensemble P_k avec $k > j$.

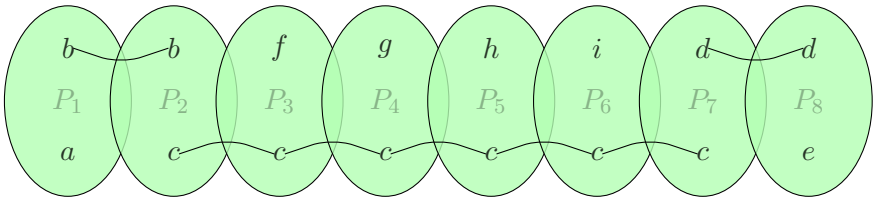
  Supposons qu'il existe $i < j$ tel que $u \in P_i$. C'est donc qu'il existe $k \leq i$ tel que $(u, v_k) \in E(G)$. Pour tout ensemble P_l tel que $k \leq i < l < j$, on aura toujours que $u \in P_l$.

Aussi, si l'ordre λ utilisé pour la construction est minimal, la taille de cette décomposition en chemin est au plus $bw(G)$. Pour toute arête (x, y), on a que $|\lambda(x) - \lambda(y)| \leq bw(G)$. Ainsi, pour tout ensemble P_i, il y a au plus $bw(G)$ sommets qui ont pu être ajoutés à l'ensemble, soient les sommets $v_{i+1}, v_{i+2}, \ldots, v_{i+bw(G)}$.

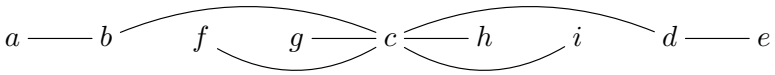
Exemple 62 *Le graphe de la figure 1.10 a une largeur de chemin égale à 1 mais une largeur de bande égale à 3.*



(a) Graphe G



(b) Une décomposition en chemin minimale de taille 1 de G



(c) Une décomposition en bande minimale de taille 3 de G

FIGURE 1.10 – Exemple de graphe avec $bw(G) > pw(G)$

Une autre manière de relier les concepts de largeur de bande et de largeur de chemin est de voir la largeur de bande comme le nombre maximal d'ensembles P_i qui contiennent un même sommet dans une décomposition en chemin.

Un ordre sur les sommets d'un graphe est un cas particulier d'une décomposition en chemin et on a vu qu'une décomposition en chemin est une décomposition en arbre. Nous avons que

$$bw(G) \geq pw(G) \geq tw(G).$$

Ces inégalités donnent l'intuition qu'un problème sur des graphes ayant une largeur de bande restreinte est plus simple qu'un problème sur des graphes ayant une largeur d'arbre restreinte. Au chapitre 3, nous allons relier la largeur d'arbre à la classe P et la largeur de bande à la classe NL.

1.8.5 Les grilles

Comme leur nom l'indique, les grilles sont des graphes qui ont une forme quadrillée. La (n, m)-grille est le graphe qui a les sommets $\{(i, j) : 1 \leq i \leq n, 1 \leq j \leq m\}$ et les arêtes $\{((i, j), (i + 1, j)) : i \leq n - 1\} \cup \{((i, j), (i, j + 1)) : j \leq m - 1\}$. La figure 1.11 est un exemple.

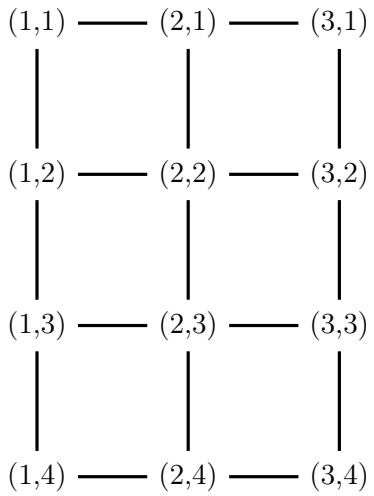


FIGURE 1.11 – La $(3,4)$-grille

Ces graphes ont de belles propriétés que nous allons exploiter. Remarquons d'abord que la largeur d'arbre de la (n,m)-grille est au plus $\min(n,m)$.

En effet, supposons que $n \leq m$. Soit $\mathcal{P}_{n,m-1}$ le chemin à $n \cdot (m-1)$ sommets

$$\{p_{1,1}, p_{2,1}, \ldots, p_{n,1}, p_{1,2}, \ldots, p_{n,m-1}\}.$$

La fonction

$$\begin{array}{rcl} \beta : V(\mathcal{P}_{n,m-1}) & \to & \{0,1\}^{|\{(i,j):1\leq i\leq n, 1\leq j\leq m\}|} \\ p_{a,b} & \mapsto & \{(u,b+1) : 1 \leq u \leq a\} \cup \{(u,b) : a \leq u \leq n\} \end{array}$$

est une décomposition en arbre de la (n,m)-grille de largeur n.

Nous aurons besoin du théorème suivant, un résultat très profond dû à Robertson et Seymour.

Théorème 63 *Pour tout k, il existe un nombre $w(k)$ tel que la (k,k)-grille est un mineur de tout graphe de largeur d'arbre plus grande ou égale à $w(k)$.*

Chapitre 2

Les problèmes d'homomorphisme avec une structure d'arrivée fixée

Un premier résultat intéressant et profond sur les problèmes de satisfaction de contraintes a été publié par Schaefer (1978). Le but de l'article était de généraliser les problèmes SAT et de les classifier. Le résultat est le suivant.

Théorème 64 (Dichotomie de Schaefer) *Soit Γ un ensemble de relations sur le domaine $\{0,1\}$. Le problème $CSP(\Gamma)$ est résoluble en temps polynomial si l'une de ces contiditions est respectée. Si pour toute relation $R \in \Gamma$ d'arité k,*

1. *le tuple $(0,\ldots,0)$ contenant k zéros est un élément de R,*
2. *le tuple $(1,\ldots,1)$ contenant k uns est un élément de R,*
3. *le tuple $(b_1,\ldots b_k) \in \{0,1\}^k$ est dans R si et seulement si*

$$b_1 \wedge \cdots \wedge b_{k-1} \Rightarrow b_k \text{ ou } b_1 \wedge \cdots \wedge b_{k-1} \Rightarrow \neg b_k,$$

4. *le tuple $(b_1,\ldots b_k) \in \{0,1\}^k$ est dans R si et seulement si*

$$b_1 \Rightarrow b_2 \vee \cdots \vee b_k \text{ ou } \neg b_1 \Rightarrow b_2 \vee \cdots \vee b_k,$$

5. *on a que $k \leq 2$, c'est-à-dire que toutes les relations de Γ sont d'arité au plus 2,*
6. *il existe un nombre $n \in \mathbb{N}$, une matrice A de dimention $n \times k$ à coefficients dans $\{0,1\}$ et un vecteur $b \in \{0,1\}^n$ tels que pour tout $\bar{x} \in R$,*

$$Ax \equiv b \ mod\ 2.$$

Sinon, $CSP(\Gamma)$ est NP-complet.

Il faut cependant faire attention de bien comprendre l'énoncé du théorème. On peut conclure que $CSP(\Gamma)$ est dans P si la même propriété est vraie pour toutes les relations de Γ. Par

exemple, si Γ contient trois relations, il faut que toutes ces relations contiennent le tuple $(0, \ldots, 0)$.

Ce théorème nous dit que si Γ est sur un domaine binaire, alors $CSP(\Gamma)$ est ou bien dans P, ou bien NP-complet. Il est légitime de se demander si une telle dichotomie existe aussi sur d'autres domaines.

Feder et Vardi (1993) ont tenté de répondre à cette question. Ils décident de regarder les problèmes de CSP comme des problèmes d'homomorphisme, font des liens entre les CSP et la théorie des bases de données et utilisent des outils de logique pour découvrir de nouveaux CSP qui sont dans P.

À l'époque, l'équivalence entre les CSP et les problèmes d'homomorphisme n'avait pas encore été mise en évidence et cet article est en quelque sorte un article précurseur. Leur définition des CSP est celle qui est encore utilisée aujourd'hui.

Dans cet article, ils conjecturent que la dichotomie de Schaefer peut s'étendre aux ensembles de relations Γ définis sur n'importe quel domaine, c'est-à-dire que pour tout Γ, on aurait que $CSP(\Gamma)$ est dans P ou $CSP(\Gamma)$ est NP-complet. Cette conjecture est toujours ouverte.

Par la suite, Jeavons et collab. (1995, 1997) publient une série d'article dans lesquels ils reprennent des résultats d'algèbre universelle qu'ils appliquent aux problèmes de satisfaction de contraintes.

L'algèbre la plus connue est une discipline qui étudie des opération binaires définies sur des ensembles et qui permet de conclure divers résultats selon les propriétés qu'ont ces opérations.

Par exemple, pour un ensemble D, s'il existe une opération $\bullet : D^2 \to D$ telle que pour tout $x, y, z \in D$, on a que

- $x \bullet y = y \bullet x$ (commutativité),
- $(x \bullet y) \bullet z = x \bullet (y \bullet z)$ (associativité),
- $x \bullet x = x$ (idempotence),

on dira que D est un semi-treillis et que $\bullet$ est une opération de semi-treillis.

L'algèbre universelle est une généralisation de l'algèbre «régulière» et étudie des opérations d'arité quelconque. Nous verrons plus loin dans ce chapitre comment l'algèbre universelle peut être appliquée à l'étude de la complexité des CSP.

Nous donnons ici un des résultats bien connu publié par Jeavons et collab. (1997) afin de donner une idée au lecteur du type de théorème pouvant être obtenu avec l'approche algébrique.

Théorème 65 *Soit Γ un ensemble de relations définies sur un domaine D. Alors $CSP(\Gamma)$*

est dans P s'il existe une opération de semi-treillis $\bullet$ sur D telle que

$$\forall R \in \Gamma, \text{ pour tout choix de tuples } (x_1, \ldots, x_k), (y_1, \ldots, y_k) \in R,$$

on a que le tuple $(x_1 \bullet y_1, \ldots, x_k \bullet y_k)$ est aussi un élément de R.

La publication de ces articles mettant en évidence l'utilité de l'algèbre universelle dans le contexte de l'étude de la complexité des CSP a incité des mathématiciens spécialisés en algèbre universelle à s'intéresser à ce domaine.

Leur apport a été considérable. En collaboration avec Jeavons, les chercheurs Bulatov et Krokhin (Bulatov et collab., 2000), ont été les premiers à trouver un critère suffisant sur un ensemble de relations Γ pour conclure que $CSP(\Gamma)$ est NP-complet. Ce critère est un critère algébrique.

Aussi, en plus d'adhérer à la conjecture citée plus haut, ils la rendent plus spécifique. Ils conjecturent que la propriété algébrique sur Γ suffisante pour conclure que $CSP(\Gamma)$ est NP-complet évoquée au paragraphe précédent est aussi une propriété nécéssaire à la NP-complétude.

Conjecture 66 *Soit Γ un ensemble de relations. Γ répond au critère C si et seulement si $CSP(\Gamma)$ est NP-complet. Sinon, $CSP(\Gamma)$ est dans P.*

Deux ans plus tard, Bulatov (2002) réussit à généraliser la dichotomie de Schaefer aux domaines à trois éléments. Il démontre que si Γ est un ensemble de relations définies sur un domaine d'arité au plus trois, alors la conjecture 66 est vraie. Il exhibe aussi un algorithme polynomial qui prend en entrée un ensemble de relations Γ et qui détermine si $CSP(\Gamma)$ est un problème dans P.

Ce même chercheur s'attaque ensuite à une autre dichotomie connue dans le domaine des problèmes de satisfactions de contraintes (Bulatov, 2005). Auparavant, Hell et Nešetřil (1990) avaient démontré le résultat suivant.

Théorème 67 *Soit $\mathcal{H}$ un graphe non dirigé sans boucle. Alors $HOM(-, \mathcal{H}) \in P$ si et seulement si $\mathcal{H}$ est bipartie. Sinon, $HOM(-, \mathcal{H})$ est NP-complet.*

Ce résultat avait été obtenu à l'aide de méthodes purement combinatoires. La preuve de ce théorème est une série de constructions et de réductions polynomiales qui s'étend sur plusieurs pages.

Bulatov refait cette démonstration en utilisant ses outils algébriques. Cette nouvelle démonstration est un excellent indicateur de la puissance de ces outils, puisqu'on peut la comparer avec celle qui n'utilisait que des méthodes combinatoires.

Non seulement Bulatov redémontre le résultat de Hell et Nestrill, mais il remarque aussi qu'encore une fois, la conjecture 66 est vraie, cette fois quand Γ contient une seule relation binaire irréflexive et symétrique, c'est-à-dire quand Γ représente un graphe sans boucle et non dirigé.

Bulatov n'a pas été le seul à travailler pour préciser la dichotomie de Schaefer. Allender et collab. (2009) ont, de leur côté, remarqué que la partie du résultat qui énonce les conditions pour lequelles $CSP(\Gamma)$ est dans P peut être raffinée.

En effet, le résultat de Schaefer donne six conditions sur Γ pour lesquelles $CSP(\Gamma)$ est dans P. Or, il existe des problèmes de satisfaction de contraintes qui ont des complexités inférieures à P, comme par exemple la classe NL. Cet article propose de classifier complètement les problèmes de satisfaction de contraintes sur des domaines binaires.

En opposition avec le résultat de Schaefer, qui infère la complexité de Γ en fonction des relations qui s'y trouvent, les critères utilisés dans cet article sont complètement algébriques.

Finalement, Larose et Tesson (2009) ont donné des critères algébriques qui garantissent la complétude de $CSP(\Gamma)$, entre autres pour les classes de compléxité L, NL et P. Ce résultat tient pour les domaines de toutes les tailles.

Les méthodes algébriques sont un outil trop important pour se permettre de les passer sous silence dans un document qui a comme objet les problèmes de satisfaction de contraintes.

Pour cette raison, dans les sections suivantes de ce chapitre, on passera en revue certains de ces résultats et on présentera une démonstration du théorème 67 qui fait appel à ces notions.

2.1 Résultats algébriques utiles

Nous avons vu plus tôt dans la section 1.6 la définition d'un problème de satisfation de contraintes $CSP(\Gamma)$ où Γ est un ensemble de relations. Dans cette section, nous ferons un survol des méthodes algébriques qui peuvent être utilisées pour étudier la complexité de $CSP(\Gamma)$.

2.1.1 Clones relationnels

Nous verrons ici qu'étant donné un ensemble de relations Γ, certaines relations peuvent être ajoutées à Γ sans modifier la complexité de $CSP(\Gamma)$.

Définition 68 (Relation pp-définissable) *Une relation R est pp-définissable par Γ si elle peut être définie de la manière suivante :*

$$R = \{(v_1, \ldots, v_n) | \exists u_1, \ldots, u_i R_1(w_1^1, \ldots, w_{l_1}^1) \wedge \cdots \wedge R_k(w_1^k, \ldots, w_{l_k}^k)\}$$

avec $w_1^1, \ldots, w_{l_1}^1, \ldots, w_1^k, \ldots, w_{l_k}^k \in \{v_1 \ldots, v_n, u_1, \ldots, u_n\}$ *et* $R_1, \ldots R_k$ *sont des relations de* Γ *ou la relation d'égalité* $=_D$.

Par exemple, si $\Gamma = \{R_1, R_2\}$ avec $R_1 = \{(0,0),(0,1),(1,1)\}$ et $R_2 = \{(0,1),(1,0)\}$, on a que

$$R = \{(v_1, v_2) | \exists u R_1(u, v_1) \wedge R_2(u, v_2)\} = \{(1,1),(0,1),(1,0)\}.$$

La relation R_1 est la relation $x \Rightarrow y$ et la relation R_2 est la relation $x \neq y$. On remarque que la nouvelle relation R est la relation $x \vee y$. On peut donc pp-définir $\vee$ avec $\Rightarrow$ et $\neq$.

De plus, retrouver les tuples de la relation R est équivalent à chercher toutes les solutions du $CSP(\Gamma)$ ayant les variables $\{v_1, v_2, u\}$ sur le domaine $\{0,1\}$ et les contraintes

$$C = \{((u, v_1), R_1), ((u, v_2), R_2)\}.$$

En effet, les solutions de cette instance sont les fonctions

$\varphi_1 : \{v_1, v_2, u\} \to \{0,1\}$	$\varphi_2 : \{v_1, v_2, u\} \to \{0,1\}$	$\varphi_3 : \{v_1, v_2, u\} \to \{0,1\}$
$v_1 \mapsto 0$	$v_1 \mapsto 1$	$v_1 \mapsto 1$
$v_2 \mapsto 1$	$v_2 \mapsto 1$	$v_2 \mapsto 0$
$u \mapsto 0$	$u \mapsto 0$	$u \mapsto 1$

et $R = \{(\varphi_1(v_1), \varphi_1(v_2)), (\varphi_2(v_1), \varphi_2(v_2)), (\varphi_3(v_1), \varphi_3(v_2))\}$. On peut généraliser cette remarque.

Remarque 69 *Une relation* R *est pp-définissable par* Γ *si et seulement s'il existe une instance* $P = (V, D, C) \in CSP(\Gamma)$ *et des variables* $v_1, \ldots, v_n \in V$ *telles que*

$$R = \{(f(v_1), \ldots, f(v_n)) | f \text{ est une solution de } P\}.$$

Définition 70 (Clone relationnel de Γ) *Le clone relationnel* $\langle\Gamma\rangle$ *d'un ensemble de relations* Γ *est l'ensemble des relations pp-définissables par* Γ.

Théorème 71 *Pour tout ensemble de relations* Γ, *pour tout* $\Gamma_0 \subseteq \langle\Gamma\rangle$, *on a que* $CSP(\Gamma_0) \leq_p CSP(\Gamma)$.

DÉMONSTRATION : On va démontrer le théorème 71 en construisant une réduction polynomiale de $CSP(\Gamma_0)$ vers $CSP(\Gamma)$.

Soient Γ et $\Gamma_0 \subseteq \langle\Gamma\rangle$. Soit P_0 une instance de $CSP(\Gamma_0)$ avec $P_0 = (V_0, D, C_0)$. On va transformer P_0 en une instance $P = (V, D, C)$ de $CSP(\Gamma)$ de la manière suivante.

Pour chaque $(s, R) \in C_0$, avec $s = (s_1, \ldots, s_n)$ et $R \in \Gamma_0$, on peut exprimer R comme

$$R = \{(v_1, \ldots, v_n) | \exists u_1, \ldots, u_i R_1(w_1^1, \ldots, w_{l_1}^1) \wedge \cdots \wedge R_k(w_1^k, \ldots, w_{l_k}^k)\}$$

avec $w_1^1, \ldots, w_{l_1}^1, \ldots, w_1^k, \ldots, w_{l_k}^k \in \{v_1 \ldots, v_n, u_1, \ldots, u_i\}$ et $R_1, \ldots, R_k \in \Gamma \cup \{=_D\}$. On fait :

1. ajouter les variables $s_1, \ldots, s_n, u_1, \ldots, u_n$ à V
2. ajouter les contraintes $((w_1^1, \ldots, w_{l_1}^1), R_1), \ldots, ((w_1^k, \ldots, w_{l_k}^k), R_k)$ à C.
3. si on a $=_D (v_i, v_j)$ dans la pp-définition de R, remplacer la variable s_j par s_i.

Montrons que P_0 a une solution si et seulement si P a une solution.

($\Rightarrow$) Soit φ une solution de P_0. Alors $\forall (s, R) \in C_0$, $(\varphi(s_1), \ldots, \varphi(s_n)) \in R$. Comme

$$R = \{(v_1, \ldots, v_n) | \exists u_1, \ldots, u_i R_1(w_1^1, \ldots, w_{l_1}^1) \wedge \cdots \wedge R_k(w_1^k, \ldots, w_{l_k}^k)\},$$

il existe $x_1, \ldots, x_i \in D$ tels que $R_1(w_1^1, \ldots, w_{l_1}^1) \wedge \cdots \wedge R_k(w_1^k, \ldots, w_{l_k}^k)$ avec

$$w_1^1, \ldots, w_{l_1}^1, \ldots, w_1^k, \ldots, w_{l_k}^k \in \{\varphi(s_1) \ldots, \varphi(s_n), x_1, \ldots, x_i\}.$$

La fonction φ' définie comme

$$\begin{array}{rcl} \varphi' : V & \rightarrow & D \\ \varphi'(v_j) & \mapsto & \varphi(v_j) \\ \varphi'(u_j) & \mapsto & x_j \end{array}$$

est une solution de P.

($\Leftarrow$) Soit $\varphi : V \rightarrow D$ une solution de P. Alors la restriction de φ dans V_0 est une solution de P_0.

□

Exemple 72 *Voici un exemple de cette réduction avec* $\Gamma = \{R_1, R_2\}$ *où* $R_1 = \{(0,1), (1,0), (1,1)\}$, $R_2 = \{(0,0), (0,1), (1,0)\}$. *On a que*

$$R_3 = \{(0,0,0), (1,0,0), (1,1,1)\} = \{(v_1, v_2, v_3) | \exists u R_1(v_1, u) \wedge R_2(v_2, u) \wedge =_D (v_2, v_3)\} \in \langle \Gamma \rangle$$

et $P_0 = (V_0, D, C_0)$ *avec* $V_0 = \{x_1, x_2, x_3\}, D = \{0, 1\}$ *et où*
$C_0 = \{((x_1, x_2, x_3), R_3), ((x_1, x_3, x_4), R_3), ((x_2, x_3, x_4), R_3)\})$ *est une instance de* $CSP(R_3)$.

On trouve $f(P_0) = P = (V, D, C)$ *de la manière suivante :*

- *Pour* $((x_1, x_2, x_3), R_3)$ *: on remplace* x_3 *par* x_2*, donc on a* $((x_1, x_2, x_2), R_3)$
  1. $V = \{x_1, x_2, u\}$

2. $C = \{((x_1, u), R_1), ((x_2, u), R_2)\}$

– *Pour* $((x_1, x_3, x_4), R_3)$ *: on remplace* x_4 *par* x_3*, et comme on avait remplacé* x_3 *par* x_2*, on a* $((x_1, x_2, x_2), R_3)$

1. $V = \{x_1, x_2, u, v\}$
2. $C = \{((x_1, u), R_1), ((x_2, u), R_2), ((x_1, v), R_1), ((x_2, v), R_2)\}$

– *Pour* $((x_2, x_3, x_4), R_3)$ *: on remplace* x_3 *et* x_4 *par* x_2*, on a* $((x_2, x_2, x_2), R_3)$

1. $V = \{x_1, x_2, u, v, w\}$
2. $C = \{((x_1, u), R_1), ((x_2, u), R_2), ((x_1, v), R_1), ((x_2, v), R_2), ((x_2, w), R_1), ((x_2, w), R_2)\}$

La solution $\varphi(x_1) = 1, \varphi(x_2) = 0, \varphi(x_3) = 0, \varphi(x_4) = 0$ *du problème* P_0 *peut être étendue à la solution* $\varphi'(x_1) = \varphi(x_1), \varphi'(x_2) = \varphi(x_2), \varphi'(u) = 0, \varphi'(v) = 0, \varphi'(w) = 1$ *du problème* P*.*

Corollaire 73 *Pour tout ensemble de relations* Γ*,*

– $CSP(\Gamma)$ *est tractable* $\Leftrightarrow CSP(\langle\Gamma\rangle)$ *est tractable*

– $CSP(\Gamma)$ *est NP-complet* $\Leftrightarrow CSP(\langle\Gamma\rangle)$ *est NP-complet*

Comme $\Gamma \subseteq \langle\Gamma\rangle$, on a que $CSP(\Gamma) \leq_p CSP(\langle\Gamma\rangle)$ et on a montré au théorème précédent que $CSP(\langle\Gamma\rangle) \leq_p CSP(\Gamma)$.

Ce corollaire nous dit que la complexité de $CSP(\Gamma)$ est la même que celle de $CSP(\langle\Gamma\rangle)$. L'ajout des relations pp-définies par Γ ne complique donc pas le problème. On pourra donc à l'avenir se concentrer sur $\langle\Gamma\rangle$.

2.1.2 Polymorphismes et Invariants

On va voir que certaines fonctions définies sur $\langle\Gamma\rangle$ permettent de déduire des informations sur la complexité des CSP sur cet ensemble.

Définition 74 (Polymorphisme) *Soit* R *une relation d'arité* k *définie sur un domaine* D *et soit* $f : D^n \to D$*. La fonction* f *est un polymorphisme de* R *si pour tout choix de* n *éléments*

$$\begin{array}{lll} t_1 & = (t_1^1, t_2^1, \ldots, t_k^1) & \in R \\ t_2 & = (t_1^2, t_2^2, \ldots, t_k^2) & \in R \\ \vdots & \vdots & \vdots \\ t_n & = (t_1^n, t_2^n, \ldots, t_k^n) & \in R \end{array}$$

on a que $(f(t_1^1, t_1^2, \ldots, t_1^n), f(t_2^1, t_2^2, \ldots, t_2^n), \ldots, f(t_k^1, t_k^2, \ldots, t_k^n)) \in R$*.*

On dit aussi que la relation R *est* ***préservée*** *par* f*.*

$$
\begin{array}{c}
\\
f \\
\downarrow \\
\\
\\
\\
\\
\end{array}
\begin{array}{c}
\begin{array}{cccc} f\downarrow & f\downarrow & & f\downarrow \end{array} \\
\left(\begin{array}{cccc}
t_1^1 & t_2^1 & \dots & t_k^1 \\
t_1^2 & t_2^2 & \dots & t_k^2 \\
\vdots & \vdots & \ddots & \vdots \\
t_1^n & t_2^n & \dots & t_k^n \\
\downarrow & \downarrow & & \downarrow \\
f(\bar{t}_1) & f(\bar{t}_2) & \dots & f(\bar{t}_k)
\end{array}\right)
\begin{array}{c} \in R \\ \in R \\ \vdots \\ \in R \\ \\ \in R \end{array}
\end{array}
$$

Exemple 75 *Prenons* $R = \{(0,0,0),(0,1,0),(1,1,0),(2,1,0),(2,0,1),(2,1,1)\}$ *une relation d'arité* $k = 3$ *sur le domaine* $D = \{0,1,2\}$. *Alors la fonction*

$$
\begin{array}{rcl}
f : D^4 & \to & D \\
(x,y,z,w) & \mapsto & max(x,y,z,w)
\end{array}
$$

est un polymorphisme de R. *Par exemple, si le choix des tuples est* $t_1 = (1,1,0), t_2 = (2,1,0), t_3 = (0,0,0)$ *et* $t_4 = (0,1,0)$, *on trouve que*

$$
(f(1,2,0,0), f(1,1,0,1), f(0,0,0,0)) = (2,1,0) \in R.
$$

On obtient un tel résultat avec tout choix de tuples t_1, t_2, t_3, t_4.

Par contre, la fonction

$$
\begin{array}{rcl}
h : D^3 & \to & D \\
(x,y,z) & \mapsto & min(x,y,z)
\end{array}
$$

n'est pas un polymorphisme de R *car le choix des tuples* $t_1 = (2,1,0), t_2 = (2,0,1)$ *et* $t_3 = (2,1,1)$ *nous donne*

$$
(f(2,2,2), f(1,0,1), f(0,1,1)) = (2,0,0) \notin R.
$$

$$
\begin{array}{c}
\\
f \\
\downarrow \\
\\
\\
\end{array}
\begin{array}{c}
\begin{array}{ccc} f\downarrow & f\downarrow & f\downarrow \end{array} \\
\left(\begin{array}{ccc}
2 & 1 & 0 \\
2 & 0 & 1 \\
2 & 1 & 1 \\
\downarrow & \downarrow & \downarrow \\
2 & 0 & 0
\end{array}\right)
\begin{array}{c} \in R \\ \in R \\ \in R \\ \\ \notin R \end{array}
\end{array}
$$

Voici deux définitions qui nous seront utiles pour la suite.

Définition 76 ($Pol(\Gamma)$) *Soit Γ un ensemble de relations sur D.*

$$Pol(\Gamma) = \{f | \forall R \in \Gamma, f \text{ est un polymorphisme de } R\}.$$

Définition 77 ($Inv(F)$) *Soit F un ensemble de fonctions $\{f_i | D^{k_i} \to D\}$.*

$$Inv(F) = \{R | \forall f \in F, R \text{ est préservée par } f\}.$$

Remarque 78 *Pour tout ensemble de relations Γ défini sur un domaine D,*

$$\Gamma \subseteq Inv(Pol(\Gamma)).$$

Soit un ensemble de relations Γ défini sur le domaine D. Montrons que pour tout $R \in \Gamma$, $R \in Inv(Pol(\Gamma))$. Il faut donc montrer que pour tout $f \in Pol(\Gamma)$, R est préservée par f, ce qui est bien vrai par définition car f est un polymorphisme de R.

Nous verrons plus loin qu'en fait $\langle\Gamma\rangle = Pol(Inv(\Gamma))$, mais avant, faisons quelques remarques sur les opérateurs Pol et Inv.

Correspondance de Galois

Définition 79 (Correspondance de Galois) *Soient deux ensembles A, B, $\mathcal{P}(A), \mathcal{P}(B)$ leurs ensembles puissance et deux opérateurs σ et τ, avec*

$$\sigma : \mathcal{P}(A) \to \mathcal{P}(B) \text{ et } \tau : \mathcal{P}(B) \to \mathcal{P}(A).$$

Les opérateurs σ et τ forment une correspondance de Galois entre les ensembles A et B si $\forall X, X' \in \mathcal{P}(A)$ et $\forall Y, Y' \in \mathcal{P}(B)$,

- $X \subseteq X' \Rightarrow \sigma(X') \subseteq \sigma(X)$
- $Y \subseteq Y' \Rightarrow \tau(Y') \subseteq \tau(Y)$
- $X \subseteq \tau(\sigma(X))$
- $Y \subseteq \sigma(\tau(Y))$.

Si D est un domaine, posons R_D comme étant l'ensemble de toutes les relations définies sur D et posons O_D l'ensemble de toutes les fonctions $f : D^n \to D$ pour tout n naturel.

Nous allons montrer que pour tout domaine D, les opérateurs Pol et Inv forment une correspondance de Galois entre l'ensemble R_D et l'ensemble O_D.

Théorème 80 *Soit D un domaine fini. Les opérateurs $Pol : \mathcal{P}(R_D) \to \mathcal{P}(O_D)$ et $Inv : \mathcal{P}(O_D) \to \mathcal{P}(R_D)$ forment une correspondance de Galois.*

DÉMONSTRATION : Soient $\Gamma, \Gamma' \subseteq R_D$ et $F, F' \subseteq O_D$.

- $\Gamma \subseteq \Gamma' \Rightarrow Pol(\Gamma') \subseteq Pol(\Gamma)$:

  Suposons que $\Gamma \subseteq \Gamma'$, et soit $f \in Pol(\Gamma')$. Alors pour tout $R \in \Gamma$, $R \in \Gamma'$ et donc f est un polymorphisme de R car f préserve toutes les relations de Γ'. On a donc que $f \in Pol(\Gamma)$.

- $\Gamma \subseteq Inv(Pol(\Gamma))$: Il s'agit de la remarque 78.

- $F \subseteq F' \Rightarrow Inv(F') \subseteq Inv(F)$:

  Suposons que $F \subseteq F'$, et soit $R \in Inv(F')$. Montrons que R est préservée par toutes les fonctions dans F. Soit donc $f \in F$. On a que $f \in F'$. Or, R est préservée par toutes les opérations de F', et donc R est préservée par f. On a bien que $R \in Inv(F)$.

- $F \subseteq Pol(Inv(F))$:

  Soit $f \in F$. Alors pour tout $R \in Inv(F)$, R est préservée par f. On a donc que f est un polymorphisme de R et finalement que $f \in Pol(Inv(F))$.

□

Lemme 81 *Si (σ, τ) forment une correspondance de Galois entre les ensembles A et B, alors $\sigma\tau\sigma = \sigma$ et $\tau\sigma\tau = \tau$.*

DÉMONSTRATION : Soit $X \in \mathcal{P}(A)$. Par définition, $X \subseteq \tau(\sigma(X))$. Aussi, $X \subseteq X' \Rightarrow \sigma(X') \subseteq \sigma(X)$. En posant $X' = \tau(\sigma(X))$, on trouve que $\sigma(\tau(\sigma(X))) \subseteq \sigma(X)$.

De plus, en utilisant la propriété que $Y \subseteq \sigma(\tau(Y))$ pour tout $Y \in \mathcal{P}(B)$ et en posant $Y = \sigma(X)$, on trouve que $\sigma(X) \subseteq \sigma(\tau(\sigma(X)))$, et donc $\sigma(\tau(\sigma(X))) = \sigma(X)$. La deuxième égalité peut être démontrée de la même manière. □

On a donc que $Inv(F) = Inv(Pol(Inv(F)))$ et que $Pol(\Gamma) = Pol(Inv(Pol(\Gamma)))$. C'est cette propriété qui rend la correspondance de Galois intéressante.

Théorème 82 *Pour tout ensemble de relations Γ, on a que $\langle\Gamma\rangle = Inv(Pol(\Gamma))$.*

DÉMONSTRATION : Nous allons d'abord montrer l'inclusion $\langle\Gamma\rangle \subseteq Inv(Pol(\Gamma))$, et ensuite nous démontrerons l'inclusion inverse $Inv(Pol(\Gamma)) \subseteq \langle\Gamma\rangle$.

1. Montrons que $\langle\Gamma\rangle \subseteq Inv(Pol(\Gamma))$.

   Pour montrer ce résultat, nous devrons prouver que toute relation pp-définissable par Γ est invariante sous les polymorphismes de Γ. Nous allons prouver que si f est un polymorphisme de Γ et que $R_1, R_2 \in \Gamma$, alors les relations

   $$\{(v_1, \ldots, v_k) | R_1(v_1, \ldots, v_i) \wedge R_2(v_{i+1}, \ldots, v_k)\},$$

   $$\{(v_1, \ldots, v_k) | \exists u_1, \ldots, u_i R_1(w_1, \ldots, w_{i+k})\}$$

   sont préservées par f. De plus, nous verrons que f préserve la relation d'égalité sur le domaine D. Nous verrons pourquoi cela nous permet de terminer la démonstration.

   – Si R_1 et R_2 sont des relations préservées par $f : D^n \to D$, alors la relation

   $$R = \{(v_1, \ldots, v_k) | R_1(v_1, \ldots, v_i) \wedge R_2(v_{i+1}, \ldots, v_k)\}$$

   est préservée par f.

   En effet, pour tout choix de n éléments $t^1, \ldots, t^n \in R$, on a que

   $$(f(t_1^1, \ldots, t_1^n), \ldots, f(t_i^1, \ldots, t_i^n)) \in R_1$$

   car f est un polymorphisme de R_1. Pour la même raison, on a aussi que

   $$(f(t_{i+1}^1, \ldots, t_{i+1}^n), \ldots, f(t_k^1, \ldots, t_k^n)) \in R_2$$

   et donc $(f(t_1^1, \ldots, t_1^n), \ldots, f(t_i^1, \ldots, t_i^n), f(t_{i+1}^1, \ldots, t_{i+1}^n), \ldots, f(t_k^1, \ldots, t_k^n)) \in R$.

   – Si R_1 est préservée par f, alors la relation

   $$R = \{(v_1, \ldots, v_k) | \exists u_1, \ldots, u_i R_1(w_1, \ldots, w_{i+k})\}$$

   avec $w_1, \ldots, w_{i+k} \in \{v_1, \ldots, v_k, u_1, \ldots, u_i\}$ est préservée par f.

   Soient

   $$\begin{array}{lll} t_1 & = (t_1^1, t_2^1, \ldots, t_k^1) & \in R \\ t_2 & = (t_1^2, t_2^2, \ldots, t_k^2) & \in R \\ \vdots & \vdots & \vdots \\ t_n & = (t_1^n, t_2^n, \ldots, t_k^n) & \in R \end{array}$$

   un choix de n éléments de R. Alors par définition,

   $$\begin{array}{c} \exists u_1^1, \ldots, u_i^1 \text{ t.q. } (w_1^1, \ldots, w_{i+k}^1) \in R_1, \text{ où } w_1^1, \ldots, w_{i+k}^1 \in \{t_1^1, \ldots, t_k^1, u_1^1, \ldots, u_i^1\}, \\ \exists u_1^2, \ldots, u_i^2 \text{ t.q. } (w_1^2, \ldots, w_{i+k}^2) \in R_1, \text{ où } w_1^2, \ldots, w_{i+k}^2 \in \{t_1^2, \ldots, t_k^2, u_1^2, \ldots, u_i^2\}, \\ \vdots \\ \exists u_1^n, \ldots, u_i^n \text{ t.q. } (w_1^n, \ldots, w_{i+k}^n) \in R_1, \text{ où } w_1^n, \ldots, w_{i+k}^n \in \{t_1^n, \ldots, t_k^n, u_1^n, \ldots, u_i^n\}. \end{array}$$

Aussi, on a que $(f(w_1^1,\ldots,w_1^n),\ldots,f(w_{i+k}^1,\ldots,w_{i+k}^n)) \in R_1$ car f est un polymorphisme de R_1. Ainsi,

$$(f(t_1^1,\ldots,t_1^n),\ldots,f(t_k^1,\ldots,t_k^n)) \in R.$$

– Toute fonction $f : D^n \to D$ est un polymorphisme de la relation $=_D$ définie comme $\{(u,u)|u \in D\}$.
En effet, pour tout choix de tuples $t_1,\ldots,t_n \in =_D$, on a que $t_1^i = t_2^i$ pour tout $i \in \{1,\ldots,n\}$ et donc $f(t_1^1,\ldots,t_1^n) = f(t_2^1,\ldots,t_2^n)$. Ainsi,

$$(f(t_1^1,\ldots,t_1^n), f(t_2^1,\ldots,t_2^n)) \in =_D .$$

Soit une relation $R \in \langle\Gamma\rangle$ où

$$R = \{(v_1,\ldots,v_j)|\exists u_1,\ldots,u_i R_1(w_1^1,\ldots,w_{l_1}^1) \wedge \cdots \wedge R_k(w_1^k,\ldots,w_{l_k}^k)\}$$

avec $w_1^1,\ldots,w_{l_1}^1,\ldots,w_1^k,\ldots,w_{l_k}^k \in \{v_1\ldots,v_j,u_1,\ldots,u_i\}$ et $R_1,\ldots R_k \in \Gamma \cup \{=_D\}$.
Pour tout $f \in Pol(\Gamma)$, les relations $R_1,\ldots R_k$ sont préservées par f. On a donc que la relation

$$R_1(w_1^1,\ldots,w_{l_1}^1) \wedge \cdots \wedge R_k(w_1^k,\ldots,w_{l_k}^k)$$

est aussi préservée par f et finalement que

$$\exists u_1,\ldots,u_i R_1(w_1^1,\ldots,w_{l_1}^1) \wedge \cdots \wedge R_k(w_1^k,\ldots,w_{l_k}^k)\}$$

est aussi préservée par f et donc $R \in Inv(Pol(\Gamma))$.

2. Montrons que $Inv(Pol(\Gamma)) \subseteq \langle\Gamma\rangle$. Soit $R \in Inv(Pol(\Gamma))$. Par la remarque 69 et la définition 70, pour montrer que $R \in \langle\Gamma\rangle$, nous allons construire un problème $P = (V,D,C) \in CSP(\Gamma)$ et choisir des variables $v^1,\ldots,v^n \in V$ telles que

$$R = \{(f(v^1),\ldots,f(v^n))|f \text{ est une solution de } P\}.$$

Supposons que R contient k relations d'arité n, c'est-à-dire que

$$R = \{t^1 = (t_1^1,t_2^1,\ldots,t_n^1), t^2 = (t_1^2,t_2^2,\ldots,t_n^2),\ldots,t^k = (t_1^k,t_2^k,\ldots,t_n^k)\}.$$

L'idée de la preuve est de construire un problème $P \in CSP(\Gamma)$ tel que l'ensemble des solutions de P soit exactement l'ensemble des polymorphismes d'arité k de Γ.

Ainsi, pour nous assurer que toutes les solutions de P soient des fonctions d'arité k sur le domaine D de Γ, on posera l'ensemble des variables de P comme étant l'ensemble $V = D^k$.

Ensuite, on veut avoir la certitude qu'une fonction $f : D^k \to D$ solution du problème P est un polymorphisme de Γ. En se reportant à la définition de polymorphisme, il faut donc que pour toute relation $R_i \in \Gamma$, pour tout choix de k j-tuples de R_i (on suppose ici que R_i est d'arité j)

$$s^1 = (s_1^1, s_2^1, \ldots, s_j^1), s^2 = (s_1^2, s_2^2, \ldots, s_j^2), \ldots, s^k = (s_1^k, s_2^k, \ldots, s_j^k) \in R_i,$$

le résultat $(f(s_1^1, s_1^2, \ldots, s_1^k), (s_2^1, s_2^2, \ldots, s_2^k), \ldots, f(s_j^1, s_j^2, \ldots, s_j^k)) \in R_i$.

Pour ce faire, nous allons mettre cette condition dans les contraintes du problème P. Ainsi, pour tout $R_i \in \Gamma$, pour tout choix possible de k j-tuples $s^1, s^2, \ldots, s^k$ de R_i, on aura la contrainte

$$(((s_1^1, s_1^2, \ldots, s_1^k), (s_2^1, s_2^2, \ldots, s_2^k), \ldots, (s_j^1, s_j^2, \ldots, s_j^k)), R_i)$$

dans l'ensemble des contraintes C.

Par exemple, si $k = 3$ et que $\Gamma = \{R_1 = \{(0,1),(1,0)\}\}$, on aura que $V = \{0,1\}^3$ et $D = \{0,1\}$. L'ensemble C va contenir les 8 contraintes suivantes :

$$(((0,0,0),(1,1,1)),R_1),(((0,0,1),(1,1,0)),R_1),$$

$$(((0,1,0),(1,0,1)),R_1),(((1,0,0),(0,1,1)),R_1),$$

$$(((1,1,0),(0,0,1)),R_1),(((1,0,1),(0,1,0)),R_1),$$

$$(((0,1,1),(1,0,0)),R_1),(((1,1,1),(0,0,0)),R_1).$$

Toujours dans cet exemple, si $f : \{0,1\}^3 \to \{0,1\}$ est un polymorphisme de Γ, alors f est une solution du problème $P = (V, D, C)$.

Nous devons maintenant choisir les variables $v^1, \ldots, v^n \in V = D^k$ utilisées pour construire l'ensemble $\{(f(v^1), \ldots, f(v^n)) | f$ est une solution de $P\}$.

Plus tôt, nous avons supposé que

$$R = \{t^1 = (t_1^1, t_2^1, \ldots, t_n^1), t^2 = (t_1^2, t_2^2, \ldots, t_n^2), \ldots, t^k = (t_1^k, t_2^k, \ldots, t_n^k)\}.$$

Les variables choisies sont

$$v^1 = (t_1^1, t_1^2, \ldots, t_1^k)$$

$$v^2 = (t_2^1, t_2^2, \ldots, t_2^k)$$

$$\vdots$$

$$v^n = (t_n^1, t_n^2, \ldots, t_n^k).$$

Nous voulons donc montrer que

$$\begin{aligned} R &= \{(t_1^1, t_2^1, \ldots, t_n^1), (t_1^2, t_2^2, \ldots, t_n^2), \ldots, (t_1^k, t_2^k, \ldots, t_n^k)\} \\ &= \{(f(t_1^1, t_1^2, \ldots, t_1^k), \ldots, f(t_n^1, t_n^2, \ldots, t_n^k)) | f \text{ est une solution de } P\}. \end{aligned}$$

Montrons d'abord que $R \subseteq \{(f(v^1), \ldots, f(v^n)) | f \text{ est une solution de } P\}$. Il suffit de remarquer que la fonction $\pi_1 : D^k \to D$ telle que $\pi_1(x_1, x_2, \ldots, x_k) = x_1$ pour tout $x_1, x_2, \ldots, x_k \in D$ est une solution de P.
En effet, chacune des contraintes

$$(((s_1^1, s_1^2, \ldots, s_1^k), (s_2^1, s_2^2, \ldots, s_2^k), \ldots, (s_j^1, s_j^2, \ldots, s_j^k)), R_i)$$

de l'ensemble C a été construite de sorte que

$$(\pi_1(s_1^1, s_1^2, \ldots, s_1^k), \pi_1(s_2^1, s_2^2, \ldots, s_2^k), \ldots, \pi_1(s_j^1, s_j^2, \ldots, s_j^k)) = (s_1^1, s_2^1, \ldots, s_j^1) \in R_i.$$

Comme π_1 est une solution de P, on a que

$$\begin{aligned} (\pi_1(v^1), \ldots, \pi_1(v^n)) &= (\pi_1(t_1^1, t_1^2, \ldots, t_1^k), \ldots, \pi_1(t_n^1, t_n^2, \ldots, t_n^k)) \\ &= (t_1^1, \ldots, t_n^1) = t^1 \in R \end{aligned}$$

On peut appliquer le même raisonement aux fonctions $\pi_2, \ldots, \pi_k$ et on obtient que tous les tuples de R sont dans l'ensemble

$$\{(f(v^1), \ldots, f(v^n)) | f \text{ est une solution de } P\}.$$

Montrons maintenant que $\{(f(v^1), \ldots, f(v^n)) | f \text{ est une solution de } P\} \subseteq R$. Le problème P a été construit de sorte que ses solutions doivent être des polymorphismes d'arité k de Γ.
Or, la relation R appartient à l'ensemble $Inv(Pol(\Gamma))$. Ainsi, si f est une solution de P, f est un polymorphisme de Γ et R est invariante sous f. Ainsi, pour tout $f : D^k \to D$ solution de P, puisque

$$\begin{aligned} t^1 &= (t_1^1, t_2^1, \ldots, t_n^1) \in R \\ t^2 &= (t_1^2, t_2^2, \ldots, t_n^2) \in R \\ &\vdots \\ t^k &= (t_1^k, t_2^k, \ldots, t_n^k) \in R, \end{aligned}$$

et que f préserve R, on a

$$(f(t_1^1, t_1^2, \ldots, t_1^k), \ldots, f(t_n^1, t_n^2, \ldots, t_n^k)) = (f(v^1), \ldots, f(v^n)) \in R.$$

□

Voici un corollaire important du théorème 82. L'idée derrière ce corollaire est que plus il y a de fonctions qui préservent les relations de Γ, plus les instances de $CSP(\Gamma)$ doivent être simples. À l'inverse, plus les relations dans Γ sont complexes, moins il y aura de polymorphismes pour cet ensemble.

Corollaire 83 *Soient Γ, Γ_0 des ensembles de relations. Si $Pol(\Gamma) \subseteq Pol(\Gamma_0)$, alors $CSP(\Gamma_0) \leq_p CSP(\Gamma)$.*

Soient Γ, Γ_0 des ensembles de relations tels que $Pol(\Gamma) \subseteq Pol(\Gamma_0)$.

$$
\begin{array}{rcll}
Pol(\Gamma) \subseteq Pol(\Gamma_0) & \Rightarrow & Inv(Pol(\Gamma_0)) \subseteq Inv(Pol(\Gamma)) & \text{(Théorème 80, correspondance de Galois)} \\
& \Rightarrow & Inv(Pol(\Gamma_0)) \subseteq \langle\Gamma\rangle & \text{(Théorème 82, } \langle\Gamma\rangle = Inv(Pol(\Gamma))\text{)} \\
\text{Or,} & & \Gamma_0 \subseteq Inv(Pol(\Gamma_0)) \subseteq \langle\Gamma\rangle & \text{(Théorème 80, correspondance de Galois)} \\
& \Rightarrow & \Gamma_0 \subseteq \langle\Gamma\rangle & \\
& \Rightarrow & CSP(\Gamma_0) \leq_p CSP(\Gamma) & \text{(Théorème 71)}
\end{array}
$$

□

Ce corollaire montre que la complexité de $CSP(\Gamma)$ est déterminée par les polymorphismes de Γ.

2.1.3 Autres résultats

Nous appliquerons les définitions et les résultats précédents aux graphes. Pour un graphe $\mathcal{G}$, l'ensemble des arêtes de $\mathcal{G}$ est une relation sur le domaine $V(\mathcal{G})$. Pour simplifier la notation, nous noterons par $\mathcal{G}$ la relation $E(\mathcal{G})$. Par exemple, on écrira $CSP(\mathcal{G})$ même si la notation correcte devrait être $CSP(E(\mathcal{G}))$.

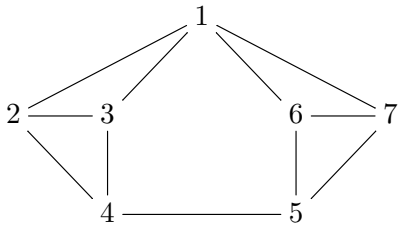
Remarquons aussi que pour un graphe $\mathcal{G}$, le problème $CSP(\mathcal{G})$ est équivalent au problème $HOM(-, \mathcal{G})$. Les résultat qui suivent sont présentés pour le cas particulier des graphes, mais ils peuvent être étendus aux ensembles de relations Γ quelconques.

Définition 84 (Sous algèbre) *Soit $\mathcal{G}$ un graphe. Une sous algèbre de $\mathcal{G}$ est une relation unaire $B \subseteq V(\mathcal{G})$ pp-définissable dans $\mathcal{G}$. Aussi, $\mathcal{G}|_B$ est le graphe avec les sommets B et l'ensemble d'arêtes $E(\mathcal{G}) \cap B^2$.*

Exemple 85 *Soit le graphe $\mathcal{G}$ suivant. La relation unaire $B = \{2, 3, 6, 7\}$ définie par*

$$
\begin{array}{ll}
B = \{x | \exists u, v, w, i, j, k & E(x,u) \wedge E(x,v) \wedge E(u,v) \wedge \\
& E(x,w) \wedge E(v,w) \wedge E(w,i) \wedge E(i,j) \wedge \\
& E(i,k) \wedge E(j,k) \wedge E(j,u) \wedge E(k,u)\}
\end{array}
$$

est une sous algèbre de $\mathcal{G}$.



On pourrait aussi définir les sous algèbres de $\mathcal{G}$ comme des ensembles $B \subseteq V(\mathcal{G})$ tels que pour tout $f \in Pol(E(\mathcal{G}))$, pour tout $b_1, \ldots, b_n \in B$, on a que

$$f(b_1, \ldots, b_n) \in B.$$

En effet,

$$\begin{aligned} B \text{ est pp-définissable dans } \mathcal{G} &\Leftrightarrow B \in \langle \mathcal{G} \rangle = Inv(Pol(\mathcal{G})) \\ &\Leftrightarrow B \text{ est préservée par tous les polymorphismes de } \mathcal{G}. \end{aligned}$$

C'est d'ailleurs parce qu'ils sont fermés sous les polymorphismes de $\mathcal{G}$ que ces ensembles portent le nom de sous algèbre.

Théorème 86 *Soit $\mathcal{G}$ un graphe.*

- *Si $CSP(\mathcal{G}) \in P$, alors pour toute sous algèbre B de $\mathcal{G}$, $CSP(\mathcal{G}|_B) \in P$.*
- *Si $\mathcal{G}$ a une sous algèbre B telle que $CSP(\mathcal{G}|_B)$ est NP-complet, alors $CSP(\mathcal{G})$ est NP-complet.*

DÉMONSTRATION : Pour tout graphe $\mathcal{G}$, $Pol(\mathcal{G}) \subseteq Pol(\mathcal{G}|_B)$. En effet, $\forall f \in Pol(\mathcal{G})$, $\forall b^1, \ldots, b^n \in \mathcal{G}|_B$,

$$(f(b_1^1, \ldots, b_1^n), f(b_2^1, \ldots, b_2^n)) \in \mathcal{G}$$

car $b^1, \ldots, b^n \in \mathcal{G}$. De plus,

$$f(b_1^1, \ldots, b_1^n) \in B \text{ et } f(b_2^1, \ldots, b_2^n) \in B$$

car B est une sous algèbre de $\mathcal{G}$. On a bien que

$$(f(b_1^1, \ldots, b_1^n), f(b_2^1, \ldots, b_2^n)) \in \mathcal{G} \cap B^2 = \mathcal{G}|_B.$$

Finalement, par le corollaire 83, comme $Pol(\mathcal{G}) \subseteq Pol(\mathcal{G}|_B)$, $CSP(\mathcal{G}|_B) \leq_p CSP(\mathcal{G})$. □

Nous aurons finalement besoin d'un dernier résultat.

Définition 87 *Soit un graphe $\mathcal{G}$. Pour $x \in V(\mathcal{G})$, la relation C_x est la relation unaire $\{x\}$.*

Les instances du problème $CSP(\mathcal{G}^c)$ sont de la forme $P = (V, D, C)$ où $D = V(\mathcal{G})$ et où les contraintes permises dans C sont des contraintes sur la relation $\mathcal{G}$ et les relations C_x pour tout $x \in V(\mathcal{G})$.

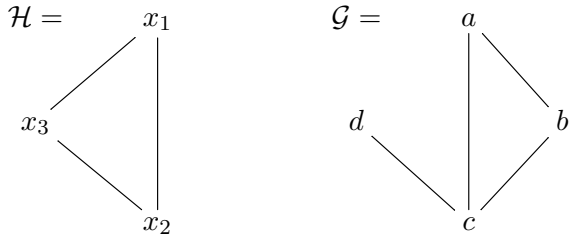
Par exemple, pour le graphe $\mathcal{G}$ avec les sommets $\{a, b, c, d\}$ et les arêtes $\{(a, b), (b, c), (c, a), (c, d)\}$,

$$P = (\{x_1, x_2, x_3\}, \{a, b, c, d\}, C)$$

où

$$C = \{((x_1, x_2), \mathcal{G}), ((x_1, x_3), \mathcal{G}), ((x_2, x_3), \mathcal{G}), C_a(x_1), C_d(x_2)\}$$

est une instance de $CSP(\mathcal{G}^c)$. Elle a une solution si et seulement s'il existe un homomorphisme $f : \mathcal{H} \to \mathcal{G}$ tel que $f(x_1) = a$ et $f(x_2) = d$. Cette instance n'a pas de solution.



Théorème 88 *Si $\mathcal{H}$ est un core, $CSP(\mathcal{H}^c) \leq_p CSP(\mathcal{H})$.*

Soit $\mathcal{G}$ une instance de $CSP(\mathcal{H}^c)$, c'est-à-dire un graphe avec une liste de contraintes $C \subseteq \{C_x(u) | u \in V(\mathcal{G}), x \in V(\mathcal{H})\}$ qui force certains sommets de $\mathcal{G}$ à être associés à un sommet particulier de $\mathcal{H}$. L'instance $\mathcal{G}$ est acceptée si et seulement s'il existe un homomorphisme $f : \mathcal{G} \to \mathcal{H}$ tel que pour toute contrainte $C_x(u) \in C$, $f(u) = x$.

Construisons un graphe $\mathcal{G}'$ tel que $\mathcal{G}' \to \mathcal{H}$ si et seulement si $\mathcal{G}$ est acceptée.

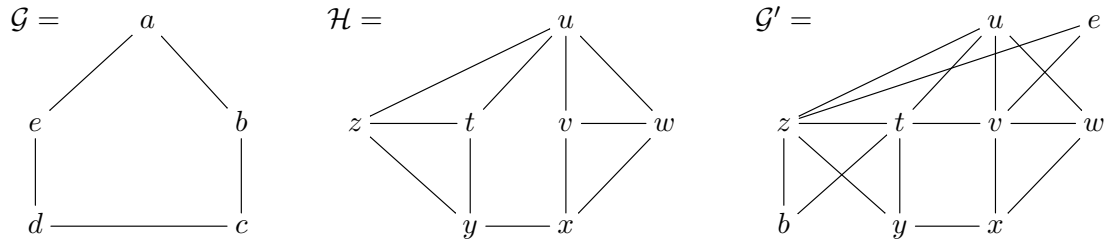
Si C contient des contraintes contradictoires, c'est-à-dire s'il existe u tel que $C_x(u)$ et $C_y(u)$ sont dans C et que $x \neq y$, le problème n'a pas de solution. On peut alors construire $\mathcal{G}'$ en prenant le graphe $\mathcal{H}$ et en y ajoutant un sommet relié à tous les autres sommets. Dans ce cas, $\mathcal{G}' \not\to \mathcal{H}$.

Supposons maintenant que C ne contient pas de contraintes contradictoires, donc que chaque sommet de $\mathcal{G}$ apparaît au plus une seule fois dans les contraintes C.

Le graphe $\mathcal{G}'$ que nous allons construire est le graphe obtenu en collant les deux graphes $\mathcal{G}$ et $\mathcal{H}$ par les sommets qui sont en relation dans C. Ainsi, les sommets de $\mathcal{G}'$ sont les sommets de

$\mathcal{H}$ ainsi que les sommets de $\mathcal{G}$ auxquels aucune contrainte de C n'est associée. Les arêtes de $\mathcal{G}'$ sont les arêtes de $\mathcal{H}$ et les arêtes de $\mathcal{G}$.

Par exemple, si l'on a les deux graphes $\mathcal{G}$ et $\mathcal{H}$ suivants et la liste des contraintes $C = \{C_z(a), C_t(c), C_v(d)\}$, le graphe $\mathcal{G}'$ est le suivant.



Soit le graphe $\mathcal{G}'$ tel que

$$V(\mathcal{G}') = V(\mathcal{H}) \cup \{u \in V(\mathcal{G}) | \forall x, C_x(u) \notin C\}$$

$$E(\mathcal{G}') = E(\mathcal{H}) \cup \{(u,v) \in E(\mathcal{G}) | u, v \in V(\mathcal{G}')\}$$

$$\cup \{(u,x) | \exists w \in V(\mathcal{G}) \text{ t.q. } C_x(w) \text{ et } (u,w) \in E(\mathcal{G})\}$$

$$\cup \{(x,y) | \exists v, w \in V(\mathcal{G}) \text{ t.q. } C_x(v) \text{ et } C_y(w) \text{ et } (v,w) \in E(\mathcal{G})\}.$$

Montrons que $\mathcal{G}$ est acceptée si et seulement si $\mathcal{G}' \to \mathcal{H}$.

($\Rightarrow$) Soit un homomorphisme $f : \mathcal{G} \to \mathcal{H}$ tel que pour tout $C_x(v) \in C$, $f(v) = x$. Alors la fonction

$$\begin{array}{rcl} g : \mathcal{G}' & \to & \mathcal{H} \\ u & \mapsto & f(u) \quad \text{si } u \in V(\mathcal{G}) \\ x & \mapsto & x \quad \text{si } x \in V(\mathcal{H}) \end{array}$$

est un homomorphisme car pour toute arête (u,v) de $\mathcal{G}'$,

- si $u, v \in V(\mathcal{G})$, alors $(g(u), g(v)) = (f(u), f(v)) \in E(\mathcal{H})$ car f est un homomorphisme
- si $u \in V(\mathcal{G})$ et $v \in V(\mathcal{H})$, alors il existe $w \in V(\mathcal{G})$ tel que $(u,w) \in E(\mathcal{G})$ et $C_v(w) \in C$. Ainsi, $f(w) = v$ et $(g(u), g(v)) = (f(u), v) = (f(u), f(w)) \in E(\mathcal{H})$.
- si $u, v \in V(\mathcal{H})$, alors
  - si $(u,v) \in E(\mathcal{H})$, $(g(u), g(v)) = (u,v) \in E(\mathcal{H})$
  - si $(u,v) \notin E(\mathcal{H})$, c'est qu'il existe $w, z \in V(\mathcal{G})$ tels que $C_u(w), C_v(z) \in C$ et que $(w,z) \in E(\mathcal{G})$. On a que $(g(u), g(v)) = (u,v) = (f(w), f(z)) \in E(\mathcal{H})$.

($\Leftarrow$) Soit un homomorphisme $f : \mathcal{G}' \to \mathcal{H}$. Posons

$$\begin{array}{rcll} \varphi : \mathcal{H} & \to & \mathcal{H} & \\ f(x) & \mapsto & x & \text{pour } x \in V(\mathcal{H}) \subset V(\mathcal{G}'). \end{array}$$

La fonction φ est un homomorphisme bijectif car $\mathcal{H}$ est un core. Montrons que

$$\begin{array}{rcll} g : \mathcal{G} & \to & \mathcal{H} & \\ u & \mapsto & \varphi(f(u)) & \text{si } u \in V(\mathcal{G}') \\ u & \mapsto & \varphi(f(x)) & \text{si } C_x(u) \in C \end{array}$$

est un homomorphisme. Pour tout $(u, v) \in E(\mathcal{G})$,

- si $u, v \in V(\mathcal{G}')$, alors $(u, v) \in E(\mathcal{G}')$ et $(f(u), f(v)) \in E(\mathcal{H})$ donc $(\varphi(f(u)), \varphi(f(v))) = (g(u), g(v)) \in E(\mathcal{H})$.
- si $u \in V(\mathcal{G}')$ et qu'il existe x tel que $C_x(v) \in C$, alors $(u, x) \in E(\mathcal{G}')$ donc $(f(u), f(x)) \in E(\mathcal{H})$ et il en est de même pour $(\varphi(f(u)), \varphi(f(x))) = (g(u), g(v))$.
- s'il existe x, y tels que $C_x(u), C_y(v)$, alors par construction, $(x, y) \in E(\mathcal{G}')$ et $(f(x), f(y)) \in E(\mathcal{H})$ donc $(g(u), g(v)) = (\varphi(f(x)), \varphi(f(y))) = (x, y) \in E(\mathcal{H})$.

Finalement, on remarque que pour tout $v \in V(\mathcal{G})$ tel que $C_x(v) \in C$, on a que $g(v) = \varphi(f(x)) = x$. □

2.2 Le problème $HOM(-, \mathcal{H})$

La complexité du problème $HOM(-, \mathcal{H})$, qui prend en entrée un graphe $\mathcal{G}$ quelconque et qui a pour but de déterminer si $\mathcal{G} \to \mathcal{H}$ est très bien documentée. En fonction des caractéristiques du graphe $\mathcal{H}$, le problème est soit trivial, soit L-complet ou soit NP-complet.

Si $\mathcal{H}$ contient une boucle, alors pour tout graphe $\mathcal{G}$ donné en entrée, $\mathcal{G} \to \mathcal{H}$ et le problème a toujours la solution «oui». Si v est le sommet de $\mathcal{H}$ tel que $(v, v) \in E(\mathcal{H})$, alors pour tout graphe $\mathcal{G}$, la fonction $h : V(\mathcal{G}) \to V(\mathcal{H})$ telle que $h(u) = v$ pour tout sommet u de $\mathcal{G}$ est un homomorphisme.

Avant de continuer, remarquons que le problème $HOM(-, \mathcal{H})$ est une généralisation du problème bien connu de k-coloriage de graphe.

Définition 89 *Problème du k-coloriage*

- *Entrée : Un graphe $\mathcal{G}$*

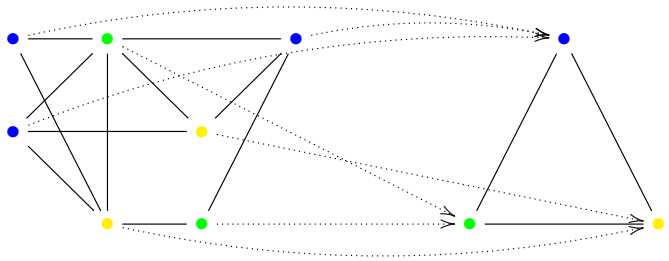


FIGURE 2.1 – Exemple de solution au problème de 3-coloriage de graphe vu sous forme de problème d'homomorphisme.

– *Problème : Déterminer s'il est possible d'assigner une des k couleurs disponibles à chaque sommet de $\mathcal{G}$ de manière à ce que deux sommets reliés par une arête ne reçoivent pas la même couleur.*

Il a été démontré que si $k \geq 3$, alors le problème du k-coloriage est NP-complet. Aussi, le 2-coloriage est L-complet.

Remarque 90 *Pour tout $k \geq 2$, le graphe $\mathcal{G}$ est k-coloriable si et seulement si $\mathcal{G} \to K_k$, où K_k est le graphe complet à k sommets.*

Soit $k \geq 2$ et $\mathcal{G}$ un graphe k-coloriable. Supposons que $\mathcal{G}$ est k-colorié. Alors la fonction qui associe à chaque sommet u de $\mathcal{G}$ le i-ième sommet de K_k si u est colorié avec la i-ème couleur disponible est un homomorphisme. Il suffit de remarquer que pour toute arête (u, v) de $\mathcal{G}$, on a par hypothèse que la couleur attribuée à u n'est pas la même couleur que celle attribuée à v. Ceci implique que $h(u) \neq h(v)$ et donc $(h(u), h(v)) \in E(K_k)$.

Inversement, s'il existe un homomorphisme $h : V(\mathcal{G}) \to V(K_k)$, alors on obtiendra un k-coloriage de $\mathcal{G}$ si l'on colorie chaque sommet u de $\mathcal{G}$ de la i-ième couleur disponible si et seulement si $h(u)$ est envoyé sur le i-ème sommet de K_k. □

Puisque le problème $HOM(-, \mathcal{H})$ est une généralisation du problème de k-coloriage, plusieurs auteurs appellent le problème $HOM(-, \mathcal{H})$ problème de $\mathcal{H}$-coloriage.

Remarquons aussi que si $\mathcal{H}$ est biparti, alors le core de $\mathcal{H}$ est le graphe K_2. On a alors que pour tout graphe $\mathcal{G}$ en entrée, $\mathcal{G} \to \mathcal{H}$ si et seulement si $\mathcal{G} \to K_2$, c'est-à-dire si et seulement si $\mathcal{G}$ est 2-coloriable. Or, le problème de 2-coloriage est L-complet. Nous pouvons donc énoncer le résultat suivant.

Théorème 91 *Si $\mathcal{H}$ est un graphe biparti, alors $HOM(-, \mathcal{H})$ est L-complet.*

Finalement, les auteurs Hell et Nešetřil ont démontré ont démontré le théorème suivant en 1990. La suite de cette section sera consacrée à sa démonstration. Bulatov a par la suite publié une nouvelle preuve qui utilise les méthodes algébriques vues à la section précédente. La preuve qui est présentée ici est un mélange de ces deux articles.

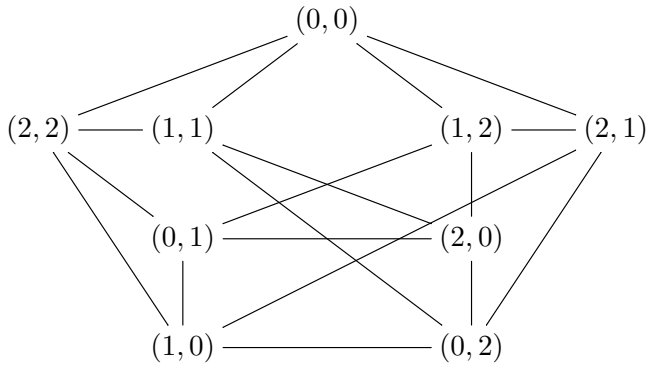
Théorème 92 *Si $\mathcal{H}$ est un graphe sans boucle et non biparti, alors $HOM(-, \mathcal{H})$ est NP-complet.*

Le concept suivant nous sera bien utile par la suite.

Définition 93 (Puissances de triangle) *Le graphe $\mathcal{T}^k$ pour $k \in \mathbb{N}$ est le graphe avec les sommets $V(\mathcal{T}) = \{0, 1, 2\}^k$ et les arêtes*

$$\mathcal{T} = \{((x_1, \ldots, x_k), (y_1, \ldots, y_k)) | x_i \neq y_i \forall i \leq k\}.$$

Voici le graphe $\mathcal{T}^2$.



Nous pouvons maintenant commencer la démonstration du théorème.

DÉMONSTRATION : Supposons le contraire, c'est-à-dire qu'il existe un graphe $\mathcal{H}$ non biparti et sans boucle tel que le $\mathcal{H}$-coloriage n'est pas NP-complet. Supposons de plus que, parmi tous les graphes ayant ces propriétés,

- $\mathcal{H}$ est celui qui a le plus petit nombre de sommets,
- parmi ceux ayant ce nombre de sommets, $\mathcal{H}$ est celui qui a le plus d'arêtes.

On a donc que $\mathcal{H}$ est un graphe connexe et un core.

Nous allons prouver qu'un tel graphe $\mathcal{H}$ n'existe pas. Pour ce faire, on va montrer au théorème 100 que $\mathcal{H}^c$ a une sous algèbre B telle que $\mathcal{H}^c|_B = \mathcal{T}^k$ pour un certain $k \in \mathbb{N}$. En combinant les théorèmes 86 et 88, on aura que

$$CSP(\mathcal{T}^k) = CSP(\mathcal{H}^c|_B) \leq_p CSP(\mathcal{H}^c) \leq_p CSP(\mathcal{H}).$$

Au théorème 101, on démontrera que pour tout $k \in \mathbb{N}$, le problème $CSP(\mathcal{T}^k)$ est NP-complet. Ceci nous permettra de conclure que $CSP(\mathcal{H})$ est aussi NP-complet. Ceci implique que l'ensemble des graphes $\mathcal{G}$ non bipartis et sans boucle tels que le $\mathcal{G}$-coloriage est NP-complet est vide car il ne contient pas de plus petit élément.

Pour démontrer le théorème 100 (qui stipule que $\mathcal{H}^c$ a une sous algèbre B telle que $\mathcal{H}^c|_B = \mathcal{T}^k$), nous aurons besoin d'une série de résultats intermédiaires. Nous allons d'abord démontrer que $\mathcal{H}$ a les propriétés suivantes :

1. $\mathcal{H}$ contient un triangle (observation 94),
2. $\mathcal{H}$ ne contient pas le graphe K_4 (observation 95),
3. chaque sommet de $\mathcal{H}$ appartient à un triangle (observation 96),
4. chaque arête de $\mathcal{H}$ appartient à au moins un triangle (observation 97),
5. chaque arête de $\mathcal{H}$ appartient à au plus un triangle (observation 98).

Nous aurons finalement besoin du fait que chaque arête de $\mathcal{H}$ est dans exactement un triangle pour démontrer un dernier résultat, le théorème 99, avant de s'attaquer à la preuve du théorème 100.

Observation 94 *$\mathcal{H}$ contient un triangle.*

DÉMONSTRATION : Pour montrer que $\mathcal{H}$ contient un triangle, on va supposer le contraire. Supposons donc que le plus petit cycle de longueur impaire dans $\mathcal{H}$ est de taille $k > 3$. Il y a au moins un cycle impair dans $\mathcal{H}$ car il est un graphe non biparti.

Soit la relation

$$\mathcal{H}^* = \{(x, y) \in V(\mathcal{H})^2 | \exists v_1, v_2 \mathcal{H}(x, v_1) \wedge \mathcal{H}(v_1, v_2) \wedge \mathcal{H}(v_2, y)\}.$$

$\mathcal{H}^*$ est pp-définissable par $\mathcal{H}$ donc $\mathcal{H}^* \in \langle \mathcal{H} \rangle$ et $CSP(\mathcal{H}^*) \leq_p CSP(\mathcal{H})$. C'est équivalent à dire que $\mathcal{H}^*$-coloriage $\leq_p$ $\mathcal{H}$-coloriage.

On remarque que

- $\mathcal{H} \subseteq \mathcal{H}^*$: en effet, si $(x, y) \in \mathcal{H}$, alors en posant $v_1 = y$ et $v_2 = x$, on a que $(x, y) \in \mathcal{H}^*$. $\mathcal{H}^*$ a donc les mêmes sommets et a au moins autant d'arêtes que $\mathcal{H}$.
- $\mathcal{H}^*$ ne contient pas de boucle : s'il y avait une boucle $(x, x) \in \mathcal{H}^*$, ce serait parce qu'il existe $v_1, v_2 \in V(\mathcal{H})$ tels que $(x, v_1), (v_1, v_2), (v_2, x) \in \mathcal{H}$. C'est impossible car on a supposé que $\mathcal{H}$ ne contient pas de triangle.
- $\mathcal{H}^*$ a au moins une arête de plus de $\mathcal{H}$: soient $c_1, c_2, \ldots, c_k$ les sommets de $\mathcal{H}$ formant un cycle impair de longeur k. Alors $(c_1, c_k), (c_k, c_{k-1}), (c_{k-1}, c_{k-2})$ sont des arêtes de $\mathcal{H}$ et donc $(c_1, c_{k-2}) \in \mathcal{H}^*$. De plus, $(c_1, c_{k-2}) \notin \mathcal{H}$ car sinon il y aurait un cycle de taille $k - 2$ dans $\mathcal{H}$.

Nous avons que $\mathcal{H}^*$ est un graphe sans boucle, non biparti, ayant autant de sommets que $\mathcal{H}$ et contenant plus d'arêtes. Le $\mathcal{H}^*$-coloriage doit donc être un problème NP-complet pour ne pas violer les hypothèses sur le choix de $\mathcal{H}$. Or, on a vu que $\mathcal{H}^*$-coloriage $\leq_P$ $\mathcal{H}$-coloriage. Le $\mathcal{H}^*$-coloriage ne peut pas être NP-complet car le $\mathcal{H}$-coloriage n'est pas NP-complet. $\mathcal{H}$ contient nécéssairement un triangle. □

Observation 95 *K_4 n'est pas un sous graphe de $\mathcal{H}$.*

DÉMONSTRATION : Encore une fois, nous allons supposer le contraire, c'est-à-dire que $\mathcal{H}$ contient K_4. Posons y comme étant l'un des sommets de $\mathcal{H}$ faisant partie du K_4.

Soit

$$B = \{(x) \in V(\mathcal{H}) | \exists v \mathcal{H}(x, v) \wedge C_y(v)\},$$

une sous algèbre de $\mathcal{H}^c$. On sait que

$$CSP(\mathcal{H}^c|_B) \leq_p CSP(\mathcal{H}^c) \leq_p CSP(\mathcal{H}).$$

Posons $\mathcal{H}^+ = \mathcal{H}^c|_B$. $\mathcal{H}^+$ est le graphe qui contient tous les voisins de y. On a que le $\mathcal{H}^+$-coloriage $\leq_p$ $\mathcal{H}$-coloriage.

$\mathcal{H}^+$ est un graphe non biparti, car il contient un triangle formé par les trois autres sommets du K_4. De plus, comme les arêtes de $\mathcal{H}^+$ sont un sous ensemble des arêtes de $\mathcal{H}$, le graphe $\mathcal{H}^+$ ne contient pas de boucle. Il a moins de sommets que $\mathcal{H}$ car y n'est pas dans $\mathcal{H}^+$.

Pour ne pas contredire nos hypothèses sur le choix de $\mathcal{H}$, le $\mathcal{H}^+$-coloriage doit être NP-complet. Or, comme $\mathcal{H}^+$-coloriage $\leq_P$ $\mathcal{H}$-coloriage, on a que que le problème du $\mathcal{H}$-coloriage est NP-complet, une contradiction. K_4 n'est pas un sous graphe de $\mathcal{H}$. □

Observation 96 *Chaque sommet de $\mathcal{H}$ appartient à un triangle*

DÉMONSTRATION : La relation unaire

$$B = \{(x) | \exists y, z \mathcal{H}(x, y) \wedge \mathcal{H}(y, z) \wedge \mathcal{H}(z, x)\}$$

est une sous algèbre de $\mathcal{H}$ qui contient tout les sommets de $\mathcal{H}$ contenus dans un triangle.

Comme $\mathcal{H}$ ne contient pas de boucle, $\mathcal{H}|_B$ n'en contient pas non plus. $\mathcal{H}|_B$ n'est pas biparti car comme il y a au moins un triangle dans $\mathcal{H}$, ces trois sommets sont nécessairement dans B et $\mathcal{H}|_B$ contient un triangle.

Si $\mathcal{H}|_B$ a moins de sommets que $\mathcal{H}$ (c'est-à-dire que $\mathcal{H}$ a des sommets qui ne sont pas dans un triangle), alors le $\mathcal{H}|_B$-coloriage doit être NP-complet. Or, par le théorème 86, ceci implique que le $\mathcal{H}$-coloriage est NP-complet, une contradiction. On conclut donc que $B = V(\mathcal{H})$ et que tous les sommets de $\mathcal{H}$ sont dans un triangle. □

Observation 97 *Chaque arête de $\mathcal{H}$ appartient à un triangle.*

DÉMONSTRATION : Supposons le contraire, c'est-à-dire qu'il existe une arête (x, y) de $\mathcal{H}$ qui n'est pas dans un triangle. Soit la sous algèbre de $\mathcal{H}^c$

$$B = \{(z)|\exists v_1, v_2 \mathcal{H}(z, v_1) \wedge \mathcal{H}(v_1, v_2) \wedge C_x(v_2)\}.$$

Le graphe $\mathcal{H}|_B$ est non biparti, car il contient tous les triangles auquels x appartient. Il ne contient pas de boucle car $\mathcal{H}$ n'en contient pas.

Par contre, y n'est pas un sommet de $\mathcal{H}|_B$ car si c'était le cas, il y aurait dans $\mathcal{H}$ un sommet v tel que (y, v) et (v, x) sont des arêtes de $\mathcal{H}$. Ce sommet v formerait donc un triangle avec les sommets x et y. Or, on a supposé qu'un tel triangle n'existait pas.

Le $\mathcal{H}|_B$-coloriage doit être un problème NP-complet pour satisfaire les hypothèses sur le choix de $\mathcal{H}$. Or, si le $\mathcal{H}|_B$-coloriage est NP-complet, le $\mathcal{H}$-coloriage l'est aussi. Cette contradiction prouve que chaque arête de $\mathcal{H}$ est dans un triangle. □

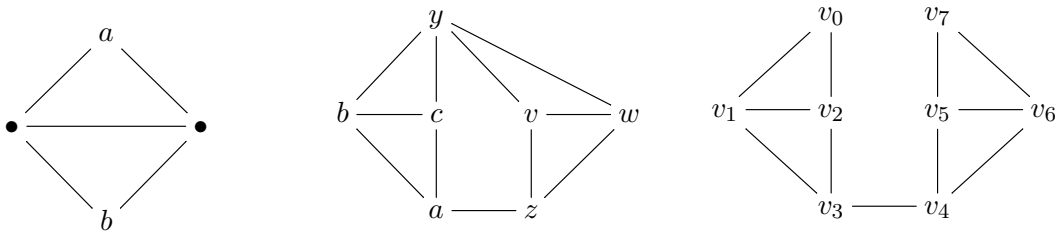
Observation 98 *Chaque arête de $\mathcal{H}$ appartient à au plus un triangle.*

DÉMONSTRATION : Ici, on veut prouver que le graphe de la figure 2.2a n'est pas un sous graphe de $\mathcal{H}$. Nous ferons cette démonstration en deux temps. Nous aurons d'abord besoin de montrer que le graphe $\mathcal{S}$ de la figure 2.2b est tel que $\mathcal{S} \not\to \mathcal{H}$. Ensuite, on prouvera que chaque arête de $\mathcal{H}$ appartient à au plus un triangle.

Montrons que $\mathcal{S} \not\to \mathcal{H}$ par contradiction. Supposons donc qu'il existe un homomorphisme $f : \mathcal{S} \to \mathcal{H}$. Cet homomorphisme associe le sommet $v \in \mathcal{S}$ à un sommet $f(v) \in \mathcal{H}$ et le sommet $w \in \mathcal{S}$ à un sommet $f(w) \in \mathcal{H}$.

Soit B la sous algèbre de $\mathcal{H}^c$ telle que

$$B = \{(x)|\exists\ i, j, k\ \mathcal{H}(i, k) \wedge \mathcal{H}(j, k) \wedge \mathcal{H}(x, k) \wedge C_{f(v)}(i) \wedge C_{f(w)}(j)\}.$$



(a) graphe interdit (b) graphe $\mathcal{S}$ (c) graphe $\mathcal{I}$

FIGURE 2.2

Ainsi, un sommet x de $\mathcal{H}$ est dans B s'il existe un sommet $k \in \mathcal{H}$ tel que $(f(v), k), (f(w), k)$ et $(x, k) \in E(\mathcal{H})$.

Par exemple, le sommet $f(b)$ fait partie de B. En effet, comme f est un homomorphisme, $(f(v), f(y)), (f(w), f(y))$ et $(f(b), f(y)) \in E(\mathcal{H})$. Pour la même raison, $f(c)$ fait aussi partie de B. Il en est de même pour $f(a)$ puisque $(f(v), f(z)), (f(w), f(z))$ et $(f(a), f(z)) \in E(\mathcal{H})$.

On pose $\mathcal{H}^+ = \mathcal{H}^c|_B$, c'est-à-dire que $\mathcal{H}^+$ est le sous graphe de $\mathcal{H}$ formé par les sommets de $\mathcal{H}$ qui sont dans B.

On remaque que H^+ n'est par biparti car il contient le triangle formé par les sommets $f(a), f(b)$ et $f(c)$. De plus, $\mathcal{H}^+$ ne contient pas de boucle car $\mathcal{H}$ n'en contient pas. Finalement, $\mathcal{H}^+$ a moins de sommets que $\mathcal{H}$ car $f(y)$ n'appartient pas à $\mathcal{H}^+$.

En effet, si $f(y)$ appartenait à $\mathcal{H}^+$, il existerait un sommet $k \in \mathcal{H}$ tel que $(f(v), k), (f(w), k)$ et $(f(y), k) \in E(\mathcal{H})$. Or, comme f est un homomorphisme, les arêtes de $\mathcal{S}$ sont préservées dans $\mathcal{H}$ et donc $\mathcal{H}$ contient aussi les arêtes $(f(v), f(w)), (f(v), f(y))$ et $(f(w), f(y))$. Ainsi, les sommets $f(v), f(w), f(y)$ et k formeraient un K_4 dans $\mathcal{H}$, ce qui n'est pas possible par l'observation 95.

On se souvient que $CSP(\mathcal{H}^+) = CSP(\mathcal{H}^c|_B) \leq_p CSP(\mathcal{H}^c) \leq_p CSP(\mathcal{H})$ et pour cette raison, on a que $\mathcal{H}^+$-coloriage $\leq_p$ $\mathcal{H}$-coloriage. Or, pour respecter les hypothèses sur le choix de $\mathcal{H}$, le problème du $\mathcal{H}^+$-coloriage doit être NP-complet, une contradiction.

Nous pouvons donc accepter l'affirmation $\mathcal{S} \not\to \mathcal{H}$.

Prouvons maintenant que le graphe de la figure 2.2a n'est pas un sous graphe de $\mathcal{H}$, et donc que chaque arête de $\mathcal{H}$ appartient à au plus un triangle.

Soit la relation

$$\mathcal{H}^* = \{(x, y) \in V(\mathcal{H})^2 | \exists v_0, \ldots, v_7 \bigwedge_{(v_i, v_j) \in \mathcal{I}} \mathcal{H}(v_i, v_j) \bigwedge = (x, v_0) \bigwedge = (y, v_7)\},$$

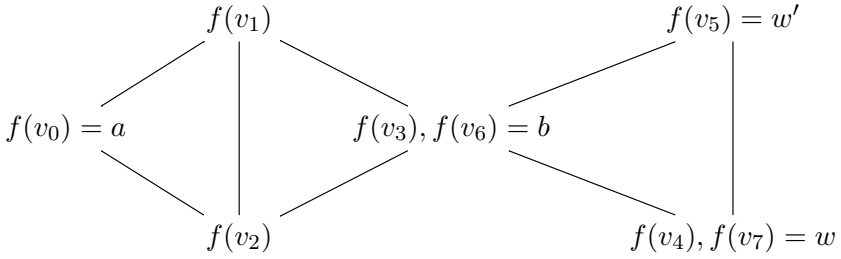
où $\mathcal{I}$ est le graphe de la figure 2.2c. On remarque que $(x, y) \in \mathcal{H}^*$ si et seulement s'il existe un homomorphisme $h : \mathcal{I} \to \mathcal{H}$ tel que $h(v_0) = x$ et $h(v_7) = y$.

Le graphe $\mathcal{H}^*$ n'a pas de boucle. En effet, s'il existait un homomorphisme $h : \mathcal{I} \to \mathcal{H}$ tel que $h(v_0) = h(v_7)$, alors il y aurait un homomorphisme entre $\mathcal{S}$ et $\mathcal{H}$.

Puisque chaque arête de $\mathcal{H}$ appartient à un triangle, $\mathcal{H}^*$ contient toutes les arêtes de $\mathcal{H}$. Si $(x, y) \in \mathcal{H}$ et que w est le sommet qui complète le triangle, alors $f : \mathcal{I} \to \mathcal{H}$ avec $f(v_0) = f(v_3) = f(v_5) = x$, $f(v_1) = f(v_4) = f(v_7) = y$ et $f(v_2) = f(v_6) = w$ est un homomorphisme. Pour cette raison, $\mathcal{H}^*$ est aussi non biparti et contient tous les sommets de $\mathcal{H}$.

Si l'on suppose que $\mathcal{H}$ contient le graphe de la figure 2.2a, on sait que $(a, b) \notin \mathcal{H}$ car $\mathcal{H}$ ne contient pas de K_4. De plus comme $\mathcal{H}$ est un core, b a au moins un voisin w qui n'est pas voisin de a. Dans le cas contraire, la fonction $f : \mathcal{H} \to \mathcal{H}$ avec $f(b) = a$ et $f(v) = v$ pour tout sommet $v \neq b$ serait un homomorphisme.

Soit donc un tel sommet w adjacent à b mais pas à a. Comme chaque arête de $\mathcal{H}$ est comprise dans un triangle, il existe au autre sommet w' tel que b, w, w' forment un triangle.



Il existe un homomorphisme $h : \mathcal{I} \to \mathcal{H}$ tel que $f(v_0) = a$ et $f(v_7) = w$, donc l'arête (a, w) est une arête de $\mathcal{H}^*$. Ce graphe est donc un graphe non biparti sans boucle avec autant de sommets que $\mathcal{H}$ mais plus d'arêtes.

Ainsi, le $\mathcal{H}^*$-coloriage doit être NP-complet pour ne pas contredire nos hypothèses sur le choix de $\mathcal{H}$. Or, comme $\mathcal{H}^*$ est pp-définissable par $\mathcal{H}$, on a que $\mathcal{H}^* \in \langle \mathcal{H} \rangle$ et donc $\mathcal{H}^*$-coloriage $\leq_p$ $\mathcal{H}$-coloriage, une contradiction. □

Maintenant que nous avons en main ces propriétés, nous allons démontrer qu'il existe une sous algèbre B de $\mathcal{H}^c$ telle que le $\mathcal{H}^c|_B$-coloriage est NP-complet.

Théorème 99 *Pour tout homomorphisme $f : \mathcal{T}^k \to \mathcal{H}$, il existe $m \leq k$ tel que le graphe $\mathcal{H}|_{Im(f)}$ formé par les sommets dans l'image de f avec les mêmes arêtes que $\mathcal{H}$ est $\mathcal{T}^m$.*

Montrons qu'il existe un ensemble $I \subseteq \{1, \ldots, k\}$ tel que pour tout $\bar{x}, \bar{y} \in \mathcal{T}^k$,

$$f(\bar{x}) = f(\bar{y}) \Leftrightarrow x_i = y_i \quad \forall i \in I.$$

Cela nous permettra de démontrer que $\mathcal{H}|_{Im(f)} = \mathcal{T}^{|I|}$ car si cette propriété est vraie,

- $|Im(f)| = 3^{|I|}$ donc $\mathcal{H}|_{Im(f)}$ contient autant de sommets que $\mathcal{T}^{|I|}$,
- $\mathcal{H}|_{Im(f)}$ a toutes les arêtes de $\mathcal{T}^{|I|}$,
- $\mathcal{H}|_{Im(f)}$ n'a pas d'autres arêtes que celles de $\mathcal{T}^{|I|}$.
  Si $\mathcal{H}|_{Im(f)}$ a une arête de plus que $\mathcal{T}^{|I|}$, alors il existe deux sommets $\bar{x}, \bar{y}$ tels que $(\bar{x}, \bar{y}) \in \mathcal{H}|_{Im(f)}$ mais $(\bar{x}, \bar{y}) \notin \mathcal{T}^{|I|}$. Comme $(\bar{x}, \bar{y}) \notin \mathcal{T}^{|I|}$, il existe deux sommets $\bar{z}, \bar{w}$ de $\mathcal{T}^{|I|}$ qui sont des voisins communs de $\bar{x}$ et $\bar{y}$. On a vu au point précédent que toutes les arêtes de $\mathcal{T}^{|I|}$ sont dans $\mathcal{H}|_{Im(f)}$ donc ce graphe contient les triangles $\bar{x}, \bar{y}, \bar{w}$ et $\bar{x}, \bar{y}, \bar{u}$, une contradiction avec la propriété 98.

Posons I comme le plus grand ensemble des indices dans $\{1, \ldots, k\}$ tel que

$$f(\bar{x}) = f(\bar{y}) \quad \Rightarrow \quad x_i = y_i \quad \forall i \in I.$$

On peut supposer qu'un tel I existe car pour $I = \emptyset$, cette implication est toujours vérifiée. Aussi, on remarque que si $I = \{1, \ldots, k\}$, alors f est injective et l'image de f est $\mathcal{T}^k$.

De plus, comme I est maximal, pour tout $j \notin I$, il existe $\bar{x}, \bar{y} \in V(\mathcal{T}^k)$ tels que $f(\bar{x}) = f(\bar{y})$ mais $x_j \neq y_j$.

Si $I = \{1, \ldots, k\}$, l'image de f est $\mathcal{T}^k$ et la preuve est terminée. Sinon, si $I \neq \{1, \ldots, k\}$, il existe des éléments qui ne sont pas dans I. Pour alléger la notation, nous allons supposer sans perte de généralité que $I = \{1, \ldots, m\}$.

Montrons que $\forall j \notin I$,

$$x_i = y_i \quad \forall i \in \{1, \ldots, k\} - j \Rightarrow f(\bar{x}) = f(\bar{y}).$$

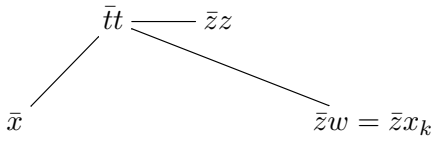
Nous ferons la preuve avec $j = k$. Cette preuve peut facilement être modifiée pour toutes les autres possibilités de j.

On veut démontrer que pour tout $z_1, \ldots, z_{k-1}, z, w \in \{0, 1, 2\}$, avec $w \neq z$ en posant $\bar{z}z = (z_1, \ldots, z_{k-1}, z)$ et $\bar{z}w = (z_1, \ldots, z_{k-1}, w)$, alors

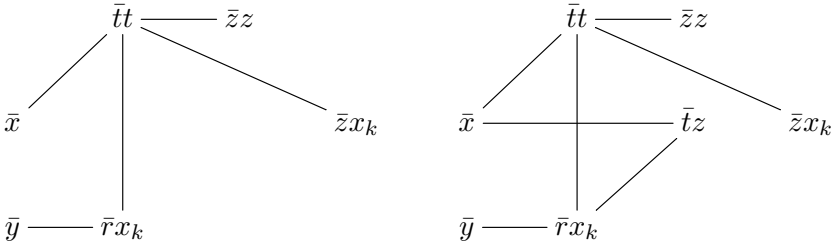
$$f(\bar{z}z) = f(\bar{z}w).$$

Par hypothèse sur la maximalité de l'ensemble I, on sait qu'il existe $\bar{x}$ et $\bar{y}$ tels que $x_k \neq y_k$ mais $f(\bar{x}) = f(\bar{y})$. Supposons sans perte de généralité que $x_k = w$. On aura donc que $x_k \neq z$.

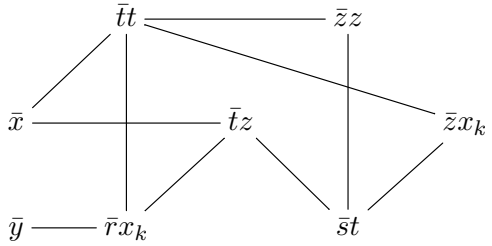
Aussi, il existe un sommet $\bar{t}t = (t_1, \ldots, t_{k-1}, t)$ tel que $t_i \neq x_i$ et $t_i \neq z_i$ pour tout i, tel que $t \neq z$ et $t \neq x_k$. Ce sommet est un voisin de $\bar{x}$, de $\bar{z}z$ et de $\bar{z}w$.



De plus, $\bar{y}$ et $\bar{t}t$ ont un voisin commun $\bar{r}x_k = (r_1, \ldots, r_{k-1}, x_k)$ où $r_i \neq y_i$ et $r_i \neq t_i$ pour tout $i \leq k-1$. Comme $x_k \neq z$ et que $x_i \neq t_i$, le sommet $\bar{t}z$ est un voisin de $\bar{r}x_k$ et de $\bar{x}$.

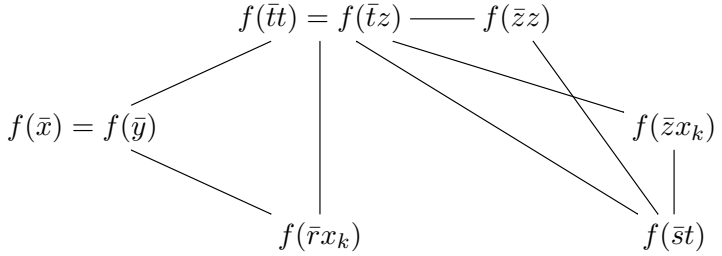


Finalement, comme pour chaque $i \leq k-1$ on sait que $z_i \neq t_i$, on peut trouver s_i tel que $s_i \neq t_i$ et $s_i \neq z_i$. On aura que le sommet $\bar{s}t$ est voisin de $\bar{t}z$, de $\bar{z}w$ et de $\bar{z}z$.



Puisque l'on sait que $f(\bar{x}) = f(\bar{y})$, on trouve que $f(\bar{x}), f(\bar{t}z)$ et $f(\bar{r}x_k)$ forment un triangle dans $\mathcal{H}$ et il en est de même pour $f(\bar{x}), f(\bar{t}t)$ et $f(\bar{r}x_k)$.

Or, nous avions démontré que $\mathcal{H}$ ne contient pas le graphe de la figure 2.2a. Il s'ensuit donc que $f(\bar{t}z) = f(\bar{t}t)$.



Nous avons encore une fois le même problème, c'est-à-dire que l'arête $(f(\bar{t}t), f(\bar{s}t))$ est comprise dans deux triangles, ce qui n'est pas possible dans $\mathcal{H}$. On doit donc conclure que $f(\bar{z}z) = f(\bar{z}x_k)$.

Il ne nous reste qu'à montrer que pour tout $\bar{x}, \bar{y} \in \mathcal{T}^k$,

$$x_i = y_i \quad \forall i \in I \Rightarrow f(\bar{x}) = f(\bar{y}).$$

Soit $x_1, \ldots, x_m, x_{m+1}, y_{m+1}, \ldots, x_k, y_k \in \{0, 1, 2\}$, montrons que

$$f(x_1, \ldots, x_k) = f(x_1, \ldots, x_m, y_{m+1}, \ldots, y_k).$$

Or, on sait que

$$\begin{aligned}
f(x_1, \ldots, x_k) &= f(x_1, \ldots, x_{k-1}, y_k) && \text{car } x_i = y_i \quad \forall i \neq k \Rightarrow f(\bar{x}) = f(\bar{y}) \\
&= f(x_1, \ldots, x_{k-2}, y_{k-1}, y_k) && \text{car } x_i = y_i \quad \forall i \neq k-1 \Rightarrow f(\bar{x}) = f(\bar{y}) \\
&\vdots \\
&= f(x_1, \ldots, x_m, y_{m+1}, \ldots, y_k) && \text{car } x_i = y_i \quad \forall i \neq m+1 \Rightarrow f(\bar{x}) = f(\bar{y}).
\end{aligned}$$

□

Théorème 100 *$\mathcal{H}^c$ a une sous algèbre B telle que $T^k = \mathcal{H}^c|_B$ pour un certain $k \in \mathbb{N}$.*

DÉMONSTRATION : Soit B_1 contenant les trois sommets d'un triangle de $\mathcal{H}$. Construisons une suite de sous ensembles $B_1 \subset B_2 \subset \cdots \subset B_j \subseteq V(\mathcal{H})$ telle que $\mathcal{H}|_{B_i} = T^{m_i}$ pour tout $i \leq j$ de la manière suivante.

Étant donné le sous ensemble $B_i \subset V(\mathcal{H})$ tel que $\mathcal{H}|_{B_i} = T^m$ pour un certain m, si B_i est une sous algèbre de $\mathcal{H}^c$, alors on pose $j = i$ et on a terminé.

Sinon, comme B_i n'est pas une sous algèbre de $\mathcal{H}^c$, il existe $v_1, \ldots, v_n \in B_i$ et un polymorphisme f de $\mathcal{H}^c$ tel que $f(v_1, \ldots, v_n) \notin B_i$.

On pose

$$B_{i+1} = \{x \in V(\mathcal{H}) | \exists y_1, \ldots, y_n \in B_i \text{ t.q. } f(y_1, \ldots, y_n) = x\}.$$

Montrons que B_{i+1} est tel que $\mathcal{H}|_{B_{i+1}} = T^{m'}$ pour un certain m' et que $B_i \subset B_{i+1}$.

Considérons la fonction

$$\begin{aligned}
g : (\mathcal{H}|_{B_i})^n &\to \mathcal{H} \\
(x_1, \ldots, x_n) &\mapsto f(x_1, \ldots, x_n).
\end{aligned}$$

Cette fonction est un homomorphisme car si

$$((x_1, \ldots, x_n), (y_1, \ldots, y_n)) \in (\mathcal{H}|_{B_i})^n \subseteq \mathcal{H}^n,$$

alors $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{H}$ et comme f est un polymorphisme de $\mathcal{H}$,

$$(f(x_1, \ldots, x_n), f(y_1, \ldots, y_n)) \in \mathcal{H}.$$

Or, comme $(\mathcal{H}|_{B_i})^n = (T^m)^n$, la fonction g est un homomorphisme de $T^{m \cdot n}$ dans $\mathcal{H}$. Par le théorème 99, on a que $\mathcal{H}|_{B_{i+1}} = T^{m'}$, pour un certain $m' \leq m \cdot n$.

De plus, B_{i+1} contient au moins un sommet de plus que B_i, le sommet $f(v_1, \ldots, v_n)$. Aussi, comme f est un polymorphisme de $\mathcal{H}^c$, la fonction f préserve la relation C_v pour tout $v \in V(\mathcal{H})$. Cela implique que $f(v, \ldots, v) = v$ pour tout $v \in B_i$, donc que $v \in B_{i+1}$. On a bien $B_i \subset B_{i+1}$.

Comme le graphe $\mathcal{H}$ est un graphe fini et que les ensembles B_i sont toujours de plus en plus gros, on trouvera éventuellement une sous algèbre de $\mathcal{H}^c$. □

Théorème 101 *Pour tout $k \in \mathbb{N}$, 3-coloriage $\leq_p$ $\mathcal{T}^k$-coloriage.*

DÉMONSTRATION : Soit $\mathcal{G}$ une instance du problème de 3-coloriage. Montrons que

$$\mathcal{G} \to \mathcal{T}^1 \iff \mathcal{G}^k \to \mathcal{T}^k.$$

La construction de $\mathcal{G}^k$ se fait en temps $O(|V(\mathcal{G})|^k) + O(|E(\mathcal{G})|^k)$.

($\Rightarrow$) Soit un homomorphisme $f : \mathcal{G} \to \mathcal{T}$. Alors

$$\begin{array}{rcl} g : \mathcal{G}^k & \to & \mathcal{T}^k \\ (x_1, \ldots, x_k) & \mapsto & (f(x_1), \ldots, f(x_k)) \end{array}$$

est un homomorphisme. En effet,

$$\begin{array}{rcl} ((x_1, \ldots, x_k), (y_1, \ldots, y_k)) \in \mathcal{G}^k & \Rightarrow & (x_1, y_1), \ldots, (x_k, y_k) \in \mathcal{G} \\ & \Rightarrow & (f(x_1), f(y_1)), \ldots, (f(x_k), f(y_k)) \in \mathcal{T} \\ & \Rightarrow & ((f(x_1), \ldots, f(x_k)), (f(y_1), \ldots, f(y_k))) \in \mathcal{T}^k \\ & \Rightarrow & (g(x_1, \ldots, x_k), g(y_1, \ldots, y_k)) \in \mathcal{T}^k. \end{array}$$

($\Leftarrow$) Supposons que $f : \mathcal{G}^k \to \mathcal{T}^k$ est un homomorphisme. Montrons que

$$\begin{array}{rcl} g : \mathcal{G} & \to & \mathcal{T} \\ x & \mapsto & \pi_1(f(x, \ldots, x)) \end{array}$$

est un homomorphisme.

$$\begin{aligned}(x,y) \in \mathcal{G} &\Rightarrow ((x,\ldots,x),(y,\ldots,y)) \in \mathcal{G}^k \\ &\Rightarrow (f(x,\ldots,x), f(y,\ldots,y)) \in \mathcal{T}^k \\ &\Rightarrow \pi_1(f(x,\ldots,x)) \neq \pi_1(f(y,\ldots,y)) \\ &\Rightarrow (g(x),g(y)) \in \mathcal{T}\end{aligned}$$

□

La preuve est maintenant terminée. Nous avons supposé qu'il existait des graphes $\mathcal{G}$ non bipartis sans boucle tels que le $\mathcal{G}$-coloriage n'est pas NP-complet. Nous avons choisi le graphe de cet ensemble qui a le plus petit nombre de sommets. Si plusieurs graphes avaient le même nombre de sommets, nous avons choisi celui avec le plus d'arêtes. Si l'ensemble n'est pas vide, ce graphe $\mathcal{H}$ existe, est un core et est connexe.

On a prouvé que $\mathcal{H}^c$ a une sous algèbre B telle que $\mathcal{H}^c|_B = \mathcal{T}^k$ pour un certain $k \in \mathbb{N}$ et on sait que le $\mathcal{T}^k$-coloriage est un problème NP-complet. Par le théorème 86, on sait que $CSP(\mathcal{H}^c)$ est aussi NP-complet. Finalement, puisque $\mathcal{H}$ est un core, par le théorème 88, $CSP(\mathcal{H})$ est NP-complet, c'est-à-dire que le $\mathcal{H}$-coloriage est NP-complet.

Nous avons une contradiction avec l'hypothèse sur le choix de $\mathcal{H}$. Il n'existe donc pas de graphe $\mathcal{G}$ sans boucle et non biparti pour lesquels le $\mathcal{G}$-coloriage n'est pas NP-complet. On peut conclure que si $\mathcal{G}$ est un graphe sans boucle et non biparti, alors le $\mathcal{G}$-coloriage est NP-complet.

Chapitre 3

Les problèmes d'homomorphisme avec une structure de départ fixée

Dans le chapitre précédent, nous avons étudié des résultats sur les problèmes d'homomorphismes où la structure d'arrivée était fixée. Nous avions un graphe d'arrivée $\mathcal{H}$ et nous voulions savoir quels étaient les graphes $\mathcal{G}$ tels que $\mathcal{G} \to \mathcal{H}$, c'est-à-dire tels qu'il existe un homomorphisme de $\mathcal{G}$ vers $\mathcal{H}$.

Par exemple, nous avons étudié le problème de 3-coloriage de graphe. Nous avons vu que la question «Est-il possible de 3-colorier le graphe $\mathcal{G}$» est équivalente à la question «Existe-il un homomorphisme de $\mathcal{G}$ vers le triangle ?». Un algorithme qui résoud le problème de 3-coloriage prendrait en entrée le graphe $\mathcal{G}$. La taille du graphe $\mathcal{G}$ est la seule à être considérée lors de l'évaluation du temps de calcul de l'algorithme.

Nous allons maintenant nous attarder à un autre type de problème d'homomorphisme, le problème $HOM(C, -)$. Encore une fois, on se questionera sur l'existence d'un homomorphisme entre deux structures mais cette fois, la structure de départ de l'homomorphisme sera connue par le problème et la structure d'arrivée sera une structure quelconque.

Par exemple, si la classe C contient tous les graphes bipartis, la question «Existe-t-il un homomorphisme du cycle à quatre sommets vers le triangle ?» est une instance du problème $HOM(C, -)$. En effet, le cycle à quatre sommets appartient à la classe C et le triangle est un graphe quelconque.

Si la classe C contient des graphes à largeur d'arbre bornée, le problème $HOM(C, -)$ est résoluble en temps polynomial. Ce résultat a été démontré par Freuder (1990). Ensuite, dans un article publié par Dalmau et collab. (2002), il est démontré qu'en fait, le problème $HOM(C, -)$ est résoluble en temps polynomial si les structures dans C ont une largeur d'arbre bornée modulo équivalence homomorphique. Ce résultat est plus général que le précédent.

Finalement, Grohe (2007) démontre en quelques sortes la contraposée de ce problème, c'est-à-dire que si la classe C n'a pas une largeur d'arbre bornée modulo équivalence hoomomorphique, alors $HOM(C, -)$ est $W[1]$ difficile. Nous allons présenter ce résultat dans le chapitre qui suit.

Nous allons ensuite voir que si la classe C contient des graphes à largeur de chemin bornée et qu'une décomposition en chemin du graphe $\mathcal{G} \in C$ donné en entrée est connue, alors on peut résoudre en espace logarithmique non déterministe le problème $HOM(C, -)$.

Nous allons ensuite présenter un algorithme qui résoud le problème $HOM(C, -)$ si C contient des graphes à largeur de bande bornée.

3.1 La complexité de $HOM(C, -)$

Grohe (2007) a établi que la complexité du problème $HOM(C, -)$ était déterminée par la largeur d'arbre des structures dans C.

Aussi, on se servira de la version paramètrée du problème $HOM(C, -)$ suivante.

Définition 102 (p-$HOM(C, -)$) *Soit C un ensemble de τ-structures. Le problème suivant est un problème paramétré :*

- *Entrée : $(\mathcal{A}, \mathcal{B})$ où $\mathcal{A} \in C$ et $\mathcal{B} \in STR[\tau]$*
- *Paramètre : $\kappa(\mathcal{A}, \mathcal{B}) = |\mathcal{A}|$*
- *Problème : Déterminer si $\mathcal{A} \to \mathcal{B}$*

Nous aurons aussi besoin de définir ce qu'est un ensemble récursivement énumérable.

Définition 103 (Ensemble récursivement énumérable) *Un ensemble est récursivement énumérable s'il existe une machine de Turing qui peut énumérer sur son ruban tous les éléments de cet ensemble.*

Autrement dit, un ensemble C est récursivement énumérable s'il existe une machine de Turing T telle que pour tout $x \in C$, il existe un instant t à partir duquel x apparait sur le ruban de T. Aussi, si $x \notin C$, alors x n'apparaitra jamais sur le ruban de T.

Par exemple, les racines rationelles d'un polynôme $p(x)$ sont récursivement énumérables car il est possible de passer en revue tous les nombres rationnels à l'aide d'une diagonalisation (voir figure 3.1) et de tester, pour tout $a \in \mathbb{Q}$ si $p(a) = 0$. Si tel est le cas, la machine écrira alors le nombre a sur son ruban.

Voici le théorème de Martin Grohe :

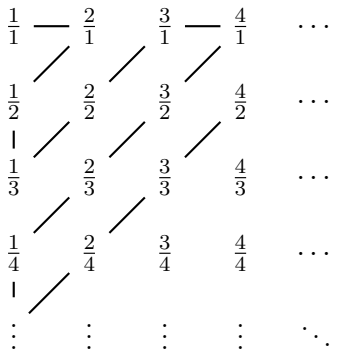


FIGURE 3.1 – Une diagonalisation

Théorème 104 *Supposons que FPT≠ W[1]. Pour toute classe C récursivement énumérable de τ-structures d'arités bornées, les énoncés suivants sont équivalents :*

1. *C a une largeur d'arbre bornée modulo équivalence homomorphique*
2. *$HOM(C,-) \in P$*
3. *$p\text{-}HOM(C,-) \in FPT$*

Si un de ces énoncés est faux, alors $p\text{-}HOM(C,-)$ est $W[1]$-difficile sous les réductions FPT.

On démontrera que $(1 \Rightarrow 2)$ à la section 3.1.1. On peut obtenir le résultat $(2 \Rightarrow 3)$ directement de la définition de la classe FPT. En effet, s'il existe une constante c et un algorithme qui détermine en temps $O((|\mathcal{A}| + |\mathcal{B}|)^c)$ si $\mathcal{A} \to \mathcal{B}$ pour tout $\mathcal{A} \in C$ et pour tout $\mathcal{B} \in STR[\tau]$, alors cet algorithme est un algorithme FPT pour $p\text{-}HOM(C,-)$.

Pour démontrer que $(3 \Rightarrow 1)$, on doit passer par la contraposée. Il est possible de démontrer que si l'énoncé 1 est faux et que C n'a pas une largeur d'arbre bornée modulo équivalence homomorphique, alors il existe une réduction FPT de p-clique à $p\text{-}HOM(C,-)$.

On se sert alors de la définition 21 pour conclure que dans ce cas $p\text{-}HOM(C,-)$ est $W[1]$-difficile. On obtiendra le résultat $(3 \Rightarrow 1)$ par notre supposition FPT≠W[1].

La réduction FPT de p-clique à $p\text{-}HOM(C,-)$ est assez complexe. Nous verrons comment faire cette réduction sur des graphes à la section 3.1.2. Le cas général pour les τ-structures est sensiblement le même.

3.1.1 Le cas polynomial

La première partie du résultat de l'article de Martin Grohe à démontrer est :

Théorème 105 *Pour toute classe C de τ-structures, si C est de largeur d'arbre bornée modulo équivalence homomorphique, alors $HOM(C,-) \in P$*

Cette partie du résultat avait aussi déjà été démontrée par Dalmau, Kolaitis et Vardi.

Le lecteur peut remarquer que les hypothèses du théorème 105 n'impliquent aucunement qu'une décomposition en arbre des structures reçues en entrée soit connue. Il n'est donc pas nécéssaire d'avoir une telle décomposition. En ayant la certitude qu'elle existe, il est possible de déterminer s'il y a un homomorphisme entre les deux structures reçues en entrée.

Pour ce faire, nous considérerons un jeu, le jeu existentiel à k jetons sur les τ-structures $(\mathcal{A}, \mathcal{B})$. Ce jeu se joue à deux joueurs, J1 et J2.

Le joueur 1 et le joueur 2 sont assis autour d'une table sur laquelle se trouve un dessin de $\mathcal{A}$ et un dessin de $\mathcal{B}$. Chacun des k jetons du joueur 1 est associé à un des k jetons du joueur 2. Au départ, aucun jeton n'est sur la table.

À la première ronde du jeu, le joueur 1 dépose certains de ses jetons sur des sommets de $\mathcal{A}$. Le joueur 2 doit ensuite déposer sur des sommets de $\mathcal{B}$ ses jetons qui correspondent à ceux mis en jeu par le joueur 1. Il doit faire en sorte que si J1 a déposé un jeton sur deux sommets de $\mathcal{A}$ reliés par une arête, alors les jetons correspondants sur $\mathcal{B}$ se retrouvent sur des sommets qui sont reliés par une arête. La fonction qui associe les jetons du joueur 1 aux jetons correspondants du joueur 2 est donc un homomorphisme entre les structure $\mathcal{A}$ et $\mathcal{B}$.

Pour les rondes suivantes, le joueur 1 peut mettre en jeu de nouveaux jetons, retirer des jetons du jeu, déplacer des jetons déjà présents ou laisser en place des jetons déjà joués. Le joueur 2 doit ajuster ses jetons correspondants pour conserver un homomorphisme. Par contre, si le joueur 1 n'a pas déplacé un jeton qui était en jeu à la ronde précédente, le joueur 2 ne peut déplacer le jeton qui lui correspond.

Le joueur 1 gagne si à une certaine ronde le joueur 2 ne réussit pas à placer ses jetons pour obtenir un homomorphisme.

Voici une définition plus formelle de ce jeu.

Définition 106 (Jeu existentiel à k jetons) *Soient $\mathcal{A}$ et $\mathcal{B}$ des τ-structures.*

- *Les configurations du jeu sont des paires (S, h) avec $S \subseteq \boldsymbol{A}$, $|S| \leq k$ et $h : S \to \boldsymbol{B}$, un homomorphisme.*

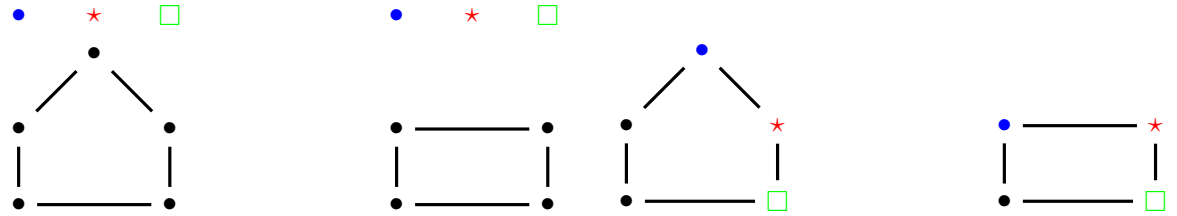
- *La configuration initiale du jeu est la paire (S_0, h_0) avec $S_0 = \emptyset$ et $h_0 : \emptyset \to \emptyset$.*

- *À chaque ronde, J1 choisit $S \subseteq \boldsymbol{A}$ avec $|S| \leq k$. Ensuite, J2 détermine la fonction $h : S \to \boldsymbol{B}$ telle que si (S', h') était la configuration précédente, alors pour tout $a \in S \cap S'$, $h(a) = h'(a)$.*

– *J1 gagne s'il réussit à choisir un ensemble S pour lequel J2 ne peut trouver de fonction h règlementaire.*

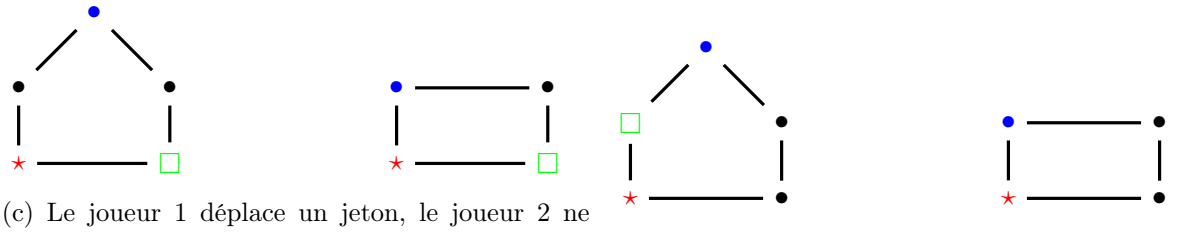
– *J2 a une stratégie gagnante s'il est en mesure de jouer indéfiniment peu importe les choix de J1.*

La figure 3.2 est un exemple du jeu existentiel à 3 jetons sur le cycle à 5 sommets et le cycle à 4 sommets pour lequel le joueur 1 gagne.



(a) Au début, les structures sont vides et les joueurs disposent de 3 jetons

(b) Le joueur 1 place ses jetons et le joueur 2 trouve un homomorphisme



(c) Le joueur 1 déplace un jeton, le joueur 2 ne déplace que le jeton correspondant

(d) Le joueur 2 ne peut trouver d'homomorphisme

FIGURE 3.2 – Une partie du jeu existentiel à 3 jetons

Théorème 107 *Il existe un algorithme qui détermine si J1 peut gagner le jeu existentiel à k jetons sur $(\mathcal{A}, \mathcal{B})$ en temps polynomial par rapport à la taille de $(\mathcal{A}, \mathcal{B})$.*

Maintenant que l'on sait qu'il est possible de déterminer le gagnant en temps polynomial par rapport à la taille des τ-structures, il ne nous reste qu'à montrer que connaître ce gagnant nous permet de déterminer s'il existe un homomorphisme entre les structures.

Théorème 108 *Soit une τ-structure $\mathcal{A}$. Si $\mathcal{A}$ est homomorphiquement équivalente à une τ-structure de largeur d'arbre au plus k, alors pour toute τ-structure $\mathcal{B}$,*

$$\mathcal{A} \to \mathcal{B} \Leftrightarrow \text{ J2 a une stratégie gagnante au jeu existentiel à } k+1 \text{ jetons}$$

DÉMONSTRATION :

($\Rightarrow$) Il est évident que si $\mathcal{A} \to \mathcal{B}$ alors J2 a une stratégie gagnante. En effet, si $h : \mathbf{A} \to \mathbf{B}$ est un homomorphisme, pour tout choix $S_i \subseteq \mathbf{A}$ de J1, J2 choisit $h|_{S_i} : S_i \to \mathbf{B}$.

($\Leftarrow$) Pour voir que si $\mathcal{A} \not\to \mathcal{B}$ alors J1 a une stratégie gagnante, il faut d'abord se rappeler que par hypothèse, il existe une sous-structure $\mathcal{A}'$ de $\mathcal{A}$ avec $tw(\mathcal{A}') \leq k$ telle que $\mathcal{A} \to \mathcal{A}'$ et $\mathcal{A}' \to \mathcal{A}$. Supposons que $(\mathcal{T}, \beta)$ est une décomposition en arbre de ce $\mathcal{A}'$ de taille au plus k.

Soit t un sommet de $\mathcal{T}$. Posons $\mathcal{T}_t$ le sous arbre de $\mathcal{T}$ ayant t comme racine et $\mathcal{A}_t$ la sous structure de $\mathcal{A}$ ayant pour univers $\{a : \exists t' \in \mathcal{T}_t \text{ t.q } a \in \beta(t')\}$.

La stratégie de J1 sera :

- À la première ronde, choisir $S_1 = \beta(r)$, où r est la racine de $\mathcal{T}$.
- Si $(\beta(t_{i-1}), h_{i-1})$ est la configuration de la ronde précédente, à la i-ième ronde, choisir t_i, un fils de t_{i-1}, pour lequel il n'existe pas d'homomorphisme $h' : \mathbf{A}_{t_i} \to \mathbf{B}$ avec $h'(a) = h_{i-1}(a)$ pour tout $a \in \beta(t_{i-1})$.
- Poser $S_i = \beta(t_i)$.

Montrons qu'il est toujours possible de faire de tels choix et que cette stratégie mène toujours à une victoire de J1.

Comme $\mathcal{A} \not\to \mathcal{B}$, si à la première ronde J2 trouve un homomorphisme $h_1 : \beta(r) \to \mathcal{B}$, cette fonction ne peut pas être étendue à un homomorphisme $h : \mathcal{A} \to \mathcal{B}$ où $\forall a \in \beta(r), h(a) = h_1(a)$.
Il y a donc un enfant t de r pour lequel toute fonction $f : \mathbf{A}_t \to \mathbf{B}$ avec $f(a) = h_1(a)$ si $a \in \beta(r)$ n'est pas un homomorphisme.
En effet, si r n'avait pas un tel fils, il existerait pour chaque fils t de r un homomorphisme $h_t : \mathbf{A}_t \to \mathbf{B}$. La fonction $h : \mathbf{A} \to \mathbf{B}$ avec $h(a) = h_t(a)$ si $a \in \mathbf{A}_t$ et $h(a) = h_1(a)$ si $a \in \beta(r)$ serait un homomorphisme, une contradiction avec l'hypothèse de départ.

À la deuxième ronde, il sera encore possible pour J1 de faire un tel choix. En effet, nous avons conclu que si t est le fils de r choisi précédemment, $\mathcal{A}_t \not\to \mathcal{B}$ et par un raisonnement identique à celui fait plus haut, t a un fils t' qui respecte la stratégie.
En continuant ainsi, J1 finira par choisir à la ronde i l'ensemble $\beta(l)$, où l est une feuille de $\mathcal{T}$. Comme l a été choisie en respectant la stratégie, toute fonction $f : \mathbf{A}_l \to \mathbf{B}$ avec $f(a) = h_{i-1}(a)$ si $a \in \beta(t_{i-1})$ n'est pas un homomorphisme. Or, $\mathbf{A}_l = \beta(l)$ et J2 ne peut trouver de fonction règlementaire.

La figure 3.3 montre un exemple de cette stratégie sur le graphe de la figure 1.5 et le cycle à quatre sommets.

□

Nous avons maintenant tous les outils en main pour démontrer le théorème 105.

Démonstration du théorème 105 Soit C un ensemble de τ-structures de largeur d'arbre bornée modulo équivalence homomorphique. Par définition, pour tout élément $\mathcal{A}$ de C, il existe une structure $\mathcal{A}'$ de largeur d'arbre au plus k telle que $\mathcal{A} \to \mathcal{A}'$ et $\mathcal{A}' \to \mathcal{A}$.

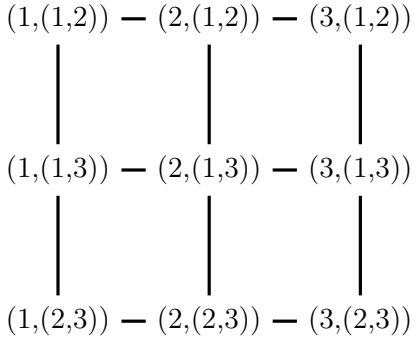
Soit une instance $(\mathcal{A}, \mathcal{B})$ de $HOM(C, -)$ avec $\mathcal{A} \in C$ et $\mathcal{B} \in STR[\tau]$. Par le théorème 108, $\mathcal{A} \to \mathcal{B}$ si et seulement si le joueur 2 a une stratégie gagnante au jeu existentiel à $k+1$ jetons sur $\mathcal{A}$ et $\mathcal{B}$. Le théorème 107 nous assure qu'il est possible de déterminer si c'est le cas en temps polynomial. □

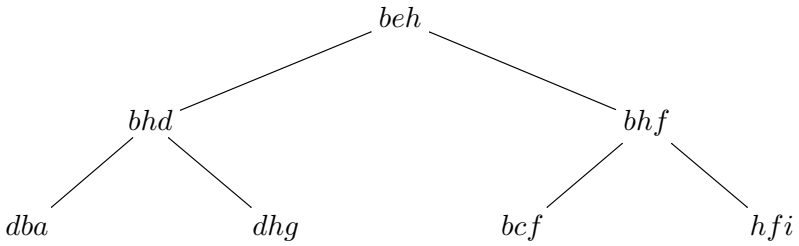
3.1.2 Le cas non polynomial

Voici une idée de la réduction FPT de p-clique à $HOM(C, -)$ dans le cas où les structures sont des graphes.

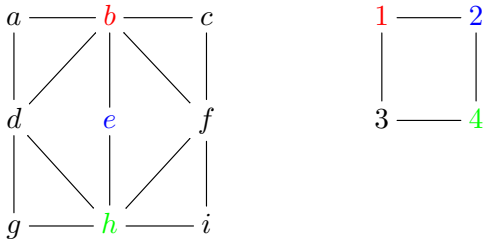
Soit $(k, \mathcal{G})$, une instance de p-clique. Posons $K = \binom{k}{2}$. Comme C n'est pas de largeur d'arbre bornée modulo équivalence homomorphique, il existe un graphe $\mathcal{A}$ dans C de largeur d'abre au moins $w(K)$. Le théorème 63 nous assure alors que la (k, K)-grille est un mineur de $\mathcal{A}$.

Il sera plus pratique pour le reste de considérer l'ensemble K' des $K = \binom{k}{2}$ paires d'éléments de $\{1, \ldots, k\}$. Par exemple, si $k = 3$, on aura $K' = \{(1,2), (1,3), (2,3)\}$. On peut nommer les sommets de la (k, K')-grille comme des éléments de $\{1, \ldots, k\} \times K'$. On obtiendra, par exemple :

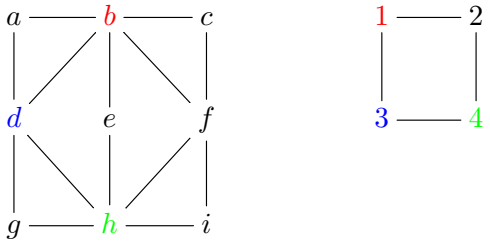




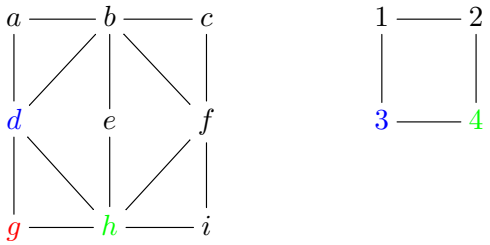
(a) Une décomposition en arbre de largeur 2 de $\mathcal{A}$



(b) Les graphe $\mathcal{A}$ et $\mathcal{B}$. J1 choisit les éléments de la racine de l'arbre en 3.3a et le joueur 2 trouve un homomorphisme



(c) J1 choisit les éléments d'un enfant de la racine de l'arbre en 3.3a et le joueur 2 trouve un homomorphisme



(d) J1 choisit maintenant les éléments d'une feuille de l'arbre en 3.3a et le joueur 2 ne peut pas trouver d'homomorphisme

FIGURE 3.3 – Un exemple de la stratégie du joueur 1 quand $\mathcal{A} \not\to \mathcal{B}$

Aussi, comme la (k, K)-grille est un mineur de $\mathcal{A}$, il existe une fonction mineur $\mu : \{1, \ldots, k\} \times K' \to \{0, 1\}^{|V(\mathcal{A})|}$ telle que

– pour tout $v \in \{1, \ldots, k\} \times K'$, $\mu(v) \neq \emptyset$

– pour tout $v, w \in \{1, \ldots, k\} \times K'$ où $v \neq w$, $\mu(v) \cap \mu(w) = \emptyset$

– pour chaque arête (v, w) de la (k, K)-grille, $\exists v' \in \mu(v)$ et $w' \in \mu(w)$ tels que $(v', w') \in E(\mathcal{A})$

On peut maintenant construire le graphe $\mathcal{B}$ tel que $\mathcal{A} \to \mathcal{B} \Leftrightarrow \mathcal{G}$ contient une k-clique. L'univers de $\mathcal{B}$ sera

$$V(\mathcal{B}) = \{(v, e, i, p, a) : v \in V(\mathcal{G}), e \in E(\mathcal{G}), i \in \{1, \ldots k\}, p \in K', a \in \mu(i, p) \text{ et } v \in e \Leftrightarrow i \in p\}.$$

On construira les arêtes de $\mathcal{B}$ de la manière suivante. Si $(a, a') \in E(\mathcal{A})$, alors on ajoutera à $E(\mathcal{B})$ toutes les paires $((v, e, i, p, a), (v', e', i', p', a'))$ de sommets de $V(\mathcal{B})$ telles que

C1 : Si $i = i'$, alors $v = v'$

C2 : Si $p = p'$, alors $e = e'$

La figure 3.4 est un exemple de la construction de $\mathcal{B}$.

Proposition 109 *Si $\mathcal{G}$ contient une k-clique alors $\mathcal{A} \to \mathcal{B}$.*

DÉMONSTRATION : Soit $v_1, v_2, \ldots, v_k$ les sommets que $\mathcal{G}$ qui forment une k-clique. Nommons $e_{(i,j)}$ l'arête entre v_i et v_j. On remarque que chaque paire (i, j) de K' est associée à une et une seule arête $e_{(i,j)}$.

La fonction $h : V(\mathcal{A}) \to V(\mathcal{B})$ telle que $h(a) = (v_i, e_p, i, p, a)$ pour $a \in \mu(i, p)$ est un homomorphisme.

– h est bien définie : Soit $a \in V(\mathcal{A})$. Par les propriétés de la fonction mineur μ, a est dans l'image d'une seule paire (i, p). De plus, par la construction des arêtes e_p de la clique, si $p = (i, j)$ pour un certain j, alors $v_i \in e_{(i,j)}$. Si $p = (j, j')$ pour $j \neq i$ et $j' \neq i$, alors $v_i \notin e_p$. On a donc que $h(a)$ est bien un élément de $V(\mathcal{B})$.

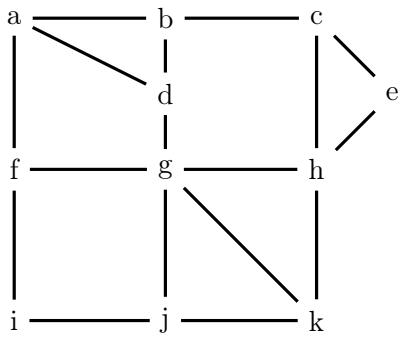
– Pour tout $(a_1, a_2) \in E(\mathcal{A})$, $(h(a_1), h(a_2)) \in E(\mathcal{B})$: Supposons que

$$(h(a_1), h(a_2)) = ((v_{i_1}, e_{p_1}, i_1, p_1, a_1), (v_{i_2}, e_{p_2}, i_2, p_2, a_2))$$

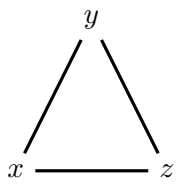
pour $a_1 \in \mu(i_1, p_1)$ et $a_2 \in \mu(i_1, p_2)$. On a que

1. si $i_1 = i_2$, alors $v_{i_1} = v_{i_2}$
2. si $p_1 = p_2$, alors $e_1 = e_2$

Comme toutes les conditions sont remplies, il y a une arête entre $h(a_1)$ et $h(a_2)$. □



(a) Le graphe $\mathcal{A}$



(b) Le graphe $\mathcal{G}$

$$
\begin{array}{rcl}
\mu : (k, K') & \to & \{0,1\}^{|V(\mathcal{A})|} \\
(1,(1,2)) & \mapsto & \{\text{a}\} \\
(2,(1,2)) & \mapsto & \{\text{b,d}\} \\
(3,(1,2)) & \mapsto & \{\text{c,e}\} \\
(1,(1,3)) & \mapsto & \{\text{f}\} \\
(2,(1,3)) & \mapsto & \{\text{g}\} \\
(3,(1,3)) & \mapsto & \{\text{h}\} \\
(1,(2,3)) & \mapsto & \{\text{ i}\} \\
(2,(2,3)) & \mapsto & \{\text{ j}\} \\
(3,(2,3)) & \mapsto & \{\text{k}\}
\end{array}
$$

(c) Une fonction mineur de la $(3,3)$-grille vers $\mathcal{A}$

$(x,(x,y),$ 1, (1,2), a)	$(x,(x,z),$ 1, (1,2), a)	$(y,(x,y),$ 1, (1,2), a)
$(x,(x,y),$ 1, (1,3), f)	$(x,(x,z),$ 1, (1,3), f)	$(y,(x,y),$ 1, (1,3), f)
$(x,(x,y),$ 2, (1,2), b)	$(x,(x,z),$ 2, (1,2), b)	$(y,(x,y),$ 2, (1,2), b)
$(x,(x,y),$ 2, (1,2), d)	$(x,(x,z),$ 2, (1,2), d)	$(y,(x,y),$ 2, (1,2), d)
$(x,(x,y),$ 2, (2,3), j)	$(x,(x,z),$ 2, (2,3), j)	$(y,(x,y),$ 2, (2,3), j)
$(x,(x,y),$ 3, (1,3), h)	$(x,(x,z),$ 3, (1,3), h)	$(y,(x,y),$ 3, (1,3), h)
$(x,(x,y),$ 3, (2,3), k)	$(x,(x,z),$ 3, (2,3), k)	$(y,(x,y),$ 3, (2,3), k)
$(z,(x,z),$ 1, (1,2), a)	$(y,(y,z),$ 1, (1,2), a)	$(z,(y,z),$ 1, (1,2), a)
$(z,(x,z),$ 1, (1,3), f)	$(y,(y,z),$ 1, (1,3), f)	$(z,(y,z),$ 1, (1,3), f)
$(z,(x,z),$ 2, (1,2), b)	$(y,(y,z),$ 2, (1,2), b)	$(z,(y,z),$ 2, (1,2), b)
$(z,(x,z),$ 2, (1,2), d)	$(y,(y,z),$ 2, (1,2), d)	$(z,(y,z),$ 2, (1,2), d)
$(z,(x,z),$ 2, (2,3), j)	$(y,(y,z),$ 2, (2,3), j)	$(z,(y,z),$ 2, (2,3), j)
$(z,(x,z),$ 3, (1,3), h)	$(y,(y,z),$ 3, (1,3), h)	$(z,(y,z),$ 3, (1,3), h)
$(z,(x,z),$ 3, (2,3), k)	$(y,(y,z),$ 3, (2,3), k)	$(z,(y,z),$ 3, (2,3), k)
$(x,(y,z),$ 1, (2,3), i)	$(y,(x,z),$ 1, (2,3), i)	$(z,(x,y),$ 1, (2,3), i)
$(x,(y,z),$ 2, (1,3), g)	$(y,(x,z),$ 2, (1,3), g)	$(z,(x,y),$ 2, (1,3), g)
$(x,(y,z),$ 3, (1,2), c)	$(y,(x,z),$ 3, (1,2), c)	$(z,(x,y),$ 3, (1,2), c)
$(x,(y,z),$ 3, (1,2), e)	$(y,(x,z),$ 3, (1,2), e)	$(z,(x,y),$ 3, (1,2), e)

(d) Les éléments dans $\mathcal{B}$

FIGURE 3.4 – Un exemple de la construction du graphe $\mathcal{B}$

Proposition 110 *Si $\mathcal{A}$ est un core et $h : V(\mathcal{A}) \to V(\mathcal{B})$ est un homomorphisme, alors pour tout $a \in V(\mathcal{A})$ tel que $a \in \mu(i,p)$ pour un certain $i \in \{1,\dots,k\}$ et un certain $p \in \{1,\dots,K\}$,*

$$h(a) = (v_a, e_a, i, p, a)$$

pour $v_a \in V(\mathcal{G})$ et $e_a \in E(\mathcal{G})$ tels que $v_a \in e_a \Leftrightarrow i \in p$.

DÉMONSTRATION : Soit la fonction $\Pi : \mathcal{B} \to \mathcal{A}$ avec $\Pi(v,e,i,p,a) = a$ pour tout $(v,e,i,p,a) \in V(\mathcal{B})$. Par la construction de $\mathcal{B}$, on a que Π est un homomorphisme. En effet, si

$$((v,e,i,p,a),(v',e',i',p',a')) \in E(\mathcal{B}),$$

c'est que $(a,a') \in E(\mathcal{A})$.

La fonction $f : V(\mathcal{A}) \to V(\mathcal{A})$ telle que $f(a) = \Pi(h(a))$ est un homomorphisme de $\mathcal{A}$ vers lui-même. Comme $\mathcal{A}$ est un core, f est bijective et on peut supposer que f est la fonction identité.

Finalement, puisque $f(a) = a = \Pi(h(a))$ et que $\Pi(v,e,i,p,a) = a$, on a que h doit envoyer chaque sommet a sur un élément (v_a, e_a, i, p, a) avec i,p tels que $a \in \mu(i,p)$ et v_a, e_a tels que $v_a \in e_a \Leftrightarrow i \in p$. □

Observation 111 *Si $\mathcal{A}$ est un core et $h : V(\mathcal{A}) \to V(\mathcal{B})$ est un homomorphisme, alors*

1. *pour tout $i \in \{1,\dots,k\}$, pour tout $p \in K'$, si $a,a' \in \mu(i,p)$, $v_a = v_{a'}$ et $e_a = e_{a'}$*
2. *pour tout $i,i' \in \{1,\dots,k\}$, pour tout $p \in K'$, si $a \in \mu(i,p)$ et $a' \in \mu(i',p)$ alors $e_a = e_{a'}$*
3. *pour tout $i \in \{1,\dots,k\}$, pour tout $p,p' \in K'$, si $a \in \mu(i,p)$ et $a' \in \mu(i,p')$ alors $v_a = v_{a'}$*

DÉMONSTRATION :

1. Soit $i \in \{1,\dots,k\}$, $p \in K'$, et $a,a' \in \mu(i,p)$. Si $(a,a') \in E(\mathcal{A})$, alors comme h est un homomorphisme, $(h(a),h(a')) \in E(\mathcal{B})$. Or, en construisant $\mathcal{B}$, nous nous sommes assurés par C1 et C2 qu'il n'y ait une arête entre deux sommets de la forme $h(a) = (v_a, e_a, i, p, a)$ et $h(a') = (v_{a'}, e_{a'}, i, p, a')$ que si $v_a = v_{a'}$ et si $e_a = e_{a'}$.
   Si $(a,a') \notin E(\mathcal{A})$, on a quand même par les propriétés de la fonction mineur μ que l'ensemble $\mu(i,p)$ est connexe dans $\mathcal{A}$. Il y a donc un chemin $(a,a_1,\dots,a_n,a')$ de a vers a' constitué de sommets dans $\mu(i,p)$. Il suffit d'appliquer le résultat précédent n fois.

2. Soient $i,i' \in \{1,\dots,k\}$, $p \in K'$, $a \in \mu(i,p)$ et $a' \in \mu(i',p)$.
   Si $i' = i+1$, alors il y a dans la (k,K')-grille une arête entre (i,p) et (i',p). Ainsi, par les propriétés de la fonction mineur μ, on sait qu'il existe $a,a' \in V(\mathcal{A})$ tels que

$a \in \mu(i,p)$, $a' \in \mu(i',p)$ et $(a,a') \in E(\mathcal{A})$. Comme h est un homomorphisme, on a que $(h(a),h(a')) \in E(\mathcal{B})$. Aussi, comme $h(a) = (v_a, e_a, i, p, a)$ et $h(a') = (v_{a'}, e_{a'}, i', p, a')$ C2 n'est respecté que si $e_a = e_{a'}$.

Si $i' \neq i+1$, alors il suffit d'appliquer ce résultat $|i'-i|$ fois.

3. On démontre l'énoncé 3 par un argument identique à celui utilisé pour l'énoncé précédent. □

Proposition 112 *Si $\mathcal{A}$ est un core et si $\mathcal{A} \to \mathcal{B}$, alors $\mathcal{G}$ contient une k-clique*

DÉMONSTRATION : Soit $h : \mathcal{A} \to \mathcal{B}$, un homomorphisme. Posons les fonctions suivantes :

- $v : \{1,\ldots,k\} \to V(\mathcal{G})$ telle que si $a \in \mu(i,p)$ pour un certain $p \in K'$ et que $h(a) = (v_a, e_a, i, p, a)$ alors $v(i) = v_a$
- $e : K' \to E(\mathcal{G})$ telle que si $a \in \mu(i,p)$ pour un certain $i \in \{1,\ldots,k\}$ et que $h(a) = (v_a, e_a, i, p, a)$, alors $e(p) = e_a$

Ces fonctions sont bien définies :

- si $a \in \mu(i,p)$ et $a' \in \mu(i,p)$, alors par 1, $v_a = v_{a'}$ et $e_a = e_{a'}$.
- si $a \in \mu(i,p)$ et $a' \in \mu(i,p')$, alors par 3, $v_a = v_{a'}$.
- si $a \in \mu(i,p)$ et $a' \in \mu(i',p)$, alors par 2, $e_a = e_{a'}$.

On a que $\{v(1), v(2), \ldots, v(k)\}$ est une clique de taille k dans $\mathcal{G}$. En effet, soient $i,j \in \{1,\ldots,k\}$. Par les propriétés de la fonction μ, il existe $a \in \mu(i,(i,j))$ et il existe $a' \in \mu(j,(i,j))$. Ainsi, on a que

- $h(a) = (v(i), e(i,j), i, (i,j), a) \in V(\mathcal{B})$
- $h(a') = (v(j), e(i,j), j, (i,j), a') \in V(\mathcal{B})$.

Nous avons construit les sommets de $\mathcal{B}$ de sorte que

- $v(i) \in e(i,j) \Leftrightarrow i \in (i,j)$
- $v(j) \in e(i,j) \Leftrightarrow j \in (i,j)$.

On a donc $(v(i), v(j)) = e(i,j)$ pour tout $i,j \in \{1,\ldots,k\}$. Finalement, comme le graphe $\mathcal{G}$ ne contient pas de boucle, la fonction v est injective et l'ensemble $\{v(1), v(2), \ldots, v(k)\}$ contient une clique de k sommets de $\mathcal{G}$. □

On peut maintenant construire une FPT-réduction de p-clique vers $HOM(C, -)$. Soit $(\mathcal{G}, k)$ une instance de p-clique. Comme par hypothèse C est récursivement énumérable, on peut l'énumérer jusqu'à ce que l'on trouve un certain $\mathcal{A}$ de largeur d'arbre plus grande ou égale à $w(k)$. Le temps de calcul pour ce faire n'est pas polynomial mais dépend uniquement du paramètre k.

Ensuite, on calcule le core $\mathcal{A}'$ de $\mathcal{A}$ et une fonction mineur μ de la (k, K')-grille à $\mathcal{A}'$, toujours en temps ne dépendant que du paramètre k. On construit ensuite le graphe $\mathcal{B}$ en fonction de k, de μ et du graphe $\mu(\mathcal{A}')$. Par construction, $|V(\mathcal{B})| \leq |V(\mathcal{A})| \cdot |V(\mathcal{G})| \cdot |E(\mathcal{G})|$ et donc $|E(\mathcal{B})| \leq (|V(\mathcal{A})| \cdot |V(\mathcal{G})| \cdot |E(\mathcal{G})|)^2$. On peut donc construire le graphe $\mathcal{B}$ en temps polynomial par rapport à $|\mathcal{G}|$ et $|\mathcal{A}|$.

On a que cette réduction de p-clique vers $HOM(C, -)$ est une réduction FPT et on a que $HOM(C, -) \in \mathrm{W}[1]$.

3.2 Les graphes à largeur de chemin bornée et NL

Nous allons maintenant nous intéresser à trouver des conditions sur une classe de graphes C qui nous permettraient de classer le problème $HOM(C, -)$ dans NL.

Il est possible de vérifier en espace logarithmique si une suite d'ensembles de sommets $P_1, \ldots, P_m$ est une décomposition en chemin d'un graphe $\mathcal{G}$ de taille k. En effet, il suffit de vérifier que

- chaque ensemble P_i contient au plus k sommets,
- chaque sommet de $\mathcal{G}$ appartient à au moins un ensemble P_i ,
- si un sommet v appartient à P_i et P_j, alors v appartient à tout les ensembles P_l pour $i \leq l \leq j$.

L'algorithme 4 fait toutes ces vérifications en n'utilisant que $O(\log(n))$ bits en espace mémoire, où n est le nombre de sommets de $\mathcal{G}$. On obtient cet ordre de grandeur en remarquant que $m \leq n$, que $|E(\mathcal{G})| \leq n^2$ et donc que $\log(n^2) = 2\log(n)$.

Définition 113 (k-chemin-$HOM(C, -)$) *Soit C un ensemble de graphes à largeur de chemin bornée par une constante k*

- *Entrée : Une décomposition en chemin $P = P_1, \ldots, P_m$ d'un graphe $\mathcal{G} \in C$ et un graphe $\mathcal{H}$ quelconque*
- *Problème : Déterminer s'il existe un homomorphisme $h : \mathcal{G} \to \mathcal{H}$.*

Ce problème est dans NL car l'algorithme 5 le résoud. Cet algorithme construit non déterministement un homomorphisme $h : \mathcal{G} \to \mathcal{H}$ en choisissant une image à chaque sommet dans

Algorithme 4 : Vérification de décomposition en chemin

Entrées : Une suite d'ensembles de sommets $P = P_1, \ldots, P_m$ et un graphe $\mathcal{G}$
Sorties : «ACCEPTER» si P est une décomposition en chemin de taille k de $\mathcal{G}$,
«REJETER» sinon.

```
// vérifier que chaque ensemble P_i contient au plus k éléments
pour chaque i = 1, ..., m faire
    si |P_i| > k alors retourner REJETER;
// vérifier que chaque sommet appartient à au moins un ensemble P_i
// vérifier que si un sommet appartient à P_i et P_j alors il appartient à tous
   les ensembles P_l avec i ≤ l ≤ j
pour chaque v ∈ V(G) faire
    si v ∈ P_1 alors trouvé← 1;
    sinon trouvé← 0;
    pour chaque i = 2, ..., m faire
        si v ∈ P_i et v ∉ P_{i-1} alors trouvé←trouvé +1;
    si trouvé ≠ 1 alors retourner REJETER;
// vérifier que chaque arête appartient à au moins un ensemble P_i
pour chaque (u, v) ∈ E(G) faire
    trouvé← faux;
    pour chaque i = 1, ..., m faire
        si u, v ∈ P_i alors trouvé←vrai;
    si trouvé = faux alors retourner REJETER;
retourner ACCEPTER;
```

l'ensemble P_1, puis à ceux dans P_2 et ainsi de suite jusqu'à P_m. Si un sommet x est à la fois dans P_i et dans P_{i+1}, on gardera en mémoire l'image qui lui avait été assignée.

Grâce aux propriétés de la décomposition en chemin, il est possible de vérifier si la fonction construite ainsi est bien un homomorphisme. Il suffit de vérifier après le choix de chaque fonction partielle $h_i : P_i \to \mathcal{H}$ si, pour tous $x, y \in P_i$ tels que $(x, y) \in E(\mathcal{G})$, on a bien $(h_i(x), h_i(y)) \in E(\mathcal{H})$.

Cette vérification est suffisante pour s'assurer de l'existence d'un homomorphisme. En effet s'il existe une exécution de l'algorithme 5 qui accepte l'entrée $(P, \mathcal{G}, \mathcal{H})$, alors $h : \mathcal{G} \to \mathcal{H}$ telle que $h(x) = h_i(x)$ si $x \in P_i$ est un homomorphisme.

En effet, comme

$$x \in P_i \cap P_{i+1} \Rightarrow h_i(x) = h_{i+1}(x)$$

et que

$$x \in P_i \text{ et } x \notin P_{i+1} \Rightarrow x \notin P_j \quad \forall j > i,$$

on sait que h est une relation déterministe. Aussi, comme chaque sommet de $\mathcal{G}$ apparait dans

au moins un des ensembles P_i, on a que h est totale. Finalement, comme

$$\forall (x, y) \in E(\mathcal{G}), \exists P_i \text{ tel que } x, y \in P_i$$

et que l'algorithme vérifie que $(h_i(x), h_i(y)) \in E(\mathcal{H})$, on s'assure que h est un homomorphisme.

De plus, l'espace mémoire nécéssaire à l'exécution de l'algorithme 5 est bien logarithmique par rapport à la taille de $\mathcal{H}$ car il est seulement nécéssaire de garder en mémoire les $k+1$ sommets de $\mathcal{H}$ qui forment l'image de h_i.

Finalement, comme nous avons vu plus haut qu'il est possible de vérifier déterministement si une suite d'ensemble $P = (P_1, \ldots, P_n)$ est bien une décomposition en chemin d'un graphe $\mathcal{G}$, on peut alors vérifier si la décomposition en chemin fournie est valide et ainsi s'assurer que si l'entrée est acceptée, c'est bien parce qu'il existe un homomorphisme de $\mathcal{G}$ vers $\mathcal{H}$. □

Algorithme 5 : Homomorphisme par décomposition en chemin

Entrées : Une décomposition en chemin $P = P_1, \ldots, P_m$ de taille k d'un graphe $\mathcal{G}$ et un graphe $\mathcal{H}$

Sorties : «ACCEPTER» si $\mathcal{G} \to \mathcal{H}$, «REJETER» si l'algorithme ne termine pas.

```
1 F[1,...,k+1] ← choisir non-déterministement k+1 sommets de H;
2 pour i = 1...m faire
3     pour chaque x, y ∈ P_i tels que (x, y) ∈ E(G) faire
4         si (F[x], F[y]) ∉ E(H) alors
5             retourner REJETER ;
6     si i < m alors
7         pour chaque x ∈ P_i tel que x ∉ P_{i+1} faire
8             F[x] ← choisir non-déterministement un sommet de H;
9 retourner ACCEPTER;
```

3.3 Les graphes à largeur de bande bornée et NL

Nous avons vu que la largeur de chemin est une propriété plus générale que la largeur de bande. À la section précédente, nous avons montré un algorithme NL pour résoudre le problème $HOM(C, -)$ si C contient des graphes à largeur de chemin bornée par une constante k et si une décomposition en chemin du graphe dans C est fournie.

Nous allons maintenant voir que si C contient plutôt des graphes à largeur de bande bornée par une constante, il n'est pas nécéssaire d'avoir une décomposition en bande pour résoudre ce problème en espace logarithmique non déterministe.

En effet, on va présenter un algorithme NL qui calcule une décomposition en bande d'un graphe reçu en entrée. En ayant cet algorithme en main, on pourra alors suivre le même principe qu'à la section précédente pour obtenir un algorithme NL qui résout le problème $HOM(C, -)$.

Définition 114 (k-bande) *Soit une constante k.*

- *Entrée : Un graphe $\mathcal{G}$ connexe.*
- *Problème : Déterminer si $bw(\mathcal{G}) \leq k$.*

Le problème k-bande est dans NL. Étant donnée une constante k, il est possible de déterminer en espace logarithmique si un graphe a une largeur de bande inférieure ou égale à k.

Lors de l'exécution de l'algorithme, il y aura trois types de sommets de $\mathcal{G}$: ceux qui ont été visités, ceux qui sont en attente d'être visités et finalement tous les autres sommets, qu'on pourrait qualifier de contaminés. Au départ, tous les sommets sont dans la catégorie des sommets contaminés. Pour passer à la catégorie des sommets visités, un sommet doit avoir tous ses voisins dans la catégorie des «visités» ou des «en attente».

À chaque itération, on choisira un nouveau sommet à placer parmi les sommets visités. Pour ce faire, on déplacera ses voisins contaminés vers l'ensemble des sommets en attente. L'algorithme se termine lorsque tous les sommets du graphe sont dans la catégorie des sommets visités.

Pour respecter les contraintes liées à l'espace, on exigera que le nombre de sommets en attente soit limité par k. De plus, on ne gardera en mémoire que $k + 1$ sommets de l'ensemble des sommets visités. Nous verrons que cet algorithme nous permet de trouver un ordre sur le graphe $\mathcal{G}$ de taille au plus k.

Dans la figure 3.5, on montre un exemple de cet algorithme sur la $(3, 3)$-grille. Les sommets contaminés sont coloriés en rouge, les sommets en attente en jaune et les sommets visités en vert.

Imaginons que la mémoire de l'algorithme pour résoudre ce problème soit organisée à la manière de la table 3.1, où les sommets s_i et $v_{i,j}$ sont des sommets du graphe reçu en entrée.

ligne #	sommets visités	voisins en attente
1	s_1	$v_{1,1}, v_{1,2}, \ldots, v_{1,i}$
2	s_2	$v_{2,1}, v_{2,2}, \ldots, v_{2,i}$
$\vdots$	$\vdots$	$\vdots$
k	s_k	$v_{k,1}, v_{k,2}, \ldots, v_{k,i}$

TABLE 3.1 – Organisation de la mémoire de l'algorithme

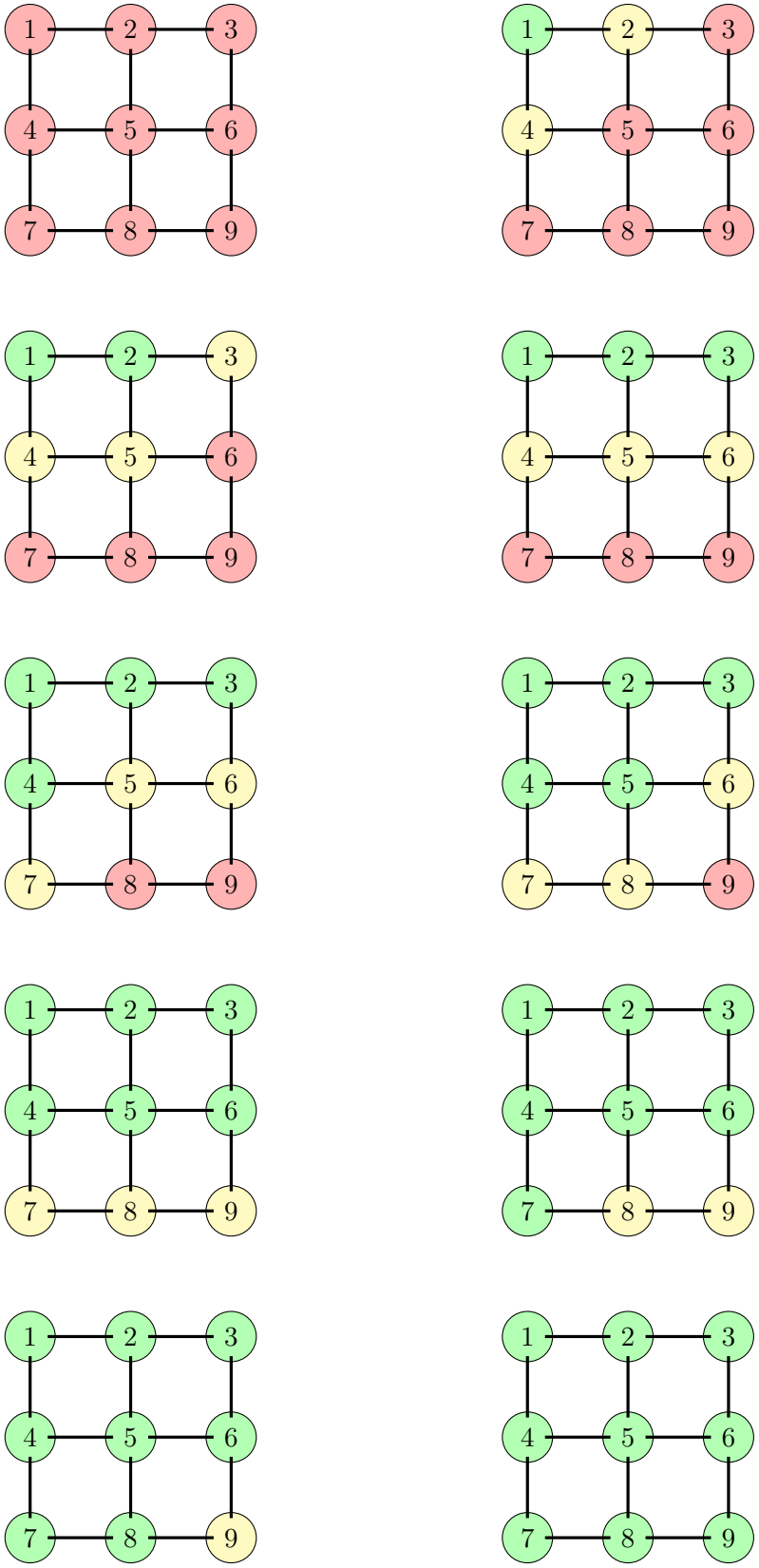


FIGURE 3.5 – Exemple d'exécution de l'algorithme sur la $(3, 3)$-grille

L'idée de l'algorithme est de trouver non déterministement un ordre λ de taille au plus k sur les sommets de $\mathcal{G}$. Il n'est pas possible pour un algorithme NL de garder en mémoire cet ordre au complet (cela prendrait $n \cdot \log(n)$ bits de mémoire, n étant la taille du graphe $\mathcal{G}$), mais grâce aux propriété de largeur de bande, il n'est pas nécéssaire de le faire.

L'algorithme 3.2 commence avec un tableau de mémoire vide. Il prend en entrée un graphe $\mathcal{G}$ et retourne «ACCEPTER» si l'algorithme a trouvé un ordre de taille k sur les sommets de $\mathcal{G}$, «REJETER» sinon.

1. Choisir un sommet s_1 avec au plus k voisins, noter ce sommet dans la première case de la première ligne
2. Noter dans la deuxième colonne de la première ligne tous les voisins de s_1
3. Initialiser le compteur *cpt* à 1
4. Tant que $cpt \neq |V(\mathcal{G})|$, faire les étapes suivantes :
   a) Choisir un sommet s dans la deuxième colonne de la mémoire sur n'importe quelle ligne
   b) Noter le sommet s sur une nouvelle ligne de la mémoire dans la colonne de gauche
   c) Garder en mémoire dans la colonne de droite de cette ligne tous les voisins de s n'apparaissant pas déjà dans la colonne des sommets visités
   d) Effacer s de la colonne des sommets en attente
   e) Si v est un voisin de s et que v apparait à la fois dans la colonne des sommets en attente sur la ligne de s et dans la colonne des sommets en attente sur une ligne précédente, effacer v de la ligne précédente
   f) Dans le cas où plus de k lignes sont remplies :
      i. Vérifier si la colonne des sommets en attente de la première ligne est vide.
      ii. Si c'est le cas, effacer la première ligne.
      iii. Sinon, retourner «REJETER».
   g) S'il y a plus de k sommets dans la colonne de droite, retourner «REJETER».
   h) Poser $cpt = cpt + 1$.
5. Si la colonne de droite est vide pour toutes les lignes de la mémoire, retourner «ACCEPTER», sinon, «REJETER».

Algorithme 3.2 – Décomposition en k-bande

La figure 3.6 donne un exemple d'exécution de cet algorithme sur la $(3, 3)$-grille.

Démontrons que cet algorithme fonctionne.

Remarque 115 *Soit $\mathcal{G}$ un graphe connexe. Si l'algorithme 3.2 se termine correctement sur $\mathcal{G}$, c'est que chaque sommet du graphe $\mathcal{G}$ est apparu une et une seule fois dans la colonne des sommets visités.*

sommets visités	voisins en attente
1	2, 4

(a) Itération 1

sommets visités	voisins en attente
1	~~2~~, 4
2	~~1~~, 3, 5

(b) Itération 2

sommets visités	voisins en attente
1	4
2	~~3~~, 5
3	~~2~~, 6

(c) Itération 3

sommets visités	voisins en attente
~~1~~	~~4~~
2	~~5~~
3	6
4	~~1~~, 5, 7

(d) Itération 4

sommets visités	voisins en attente
~~2~~	
3	~~6~~
4	~~5~~, 7
5	~~2~~, ~~4~~, 6, 8

(e) Itération 5

sommets visités	voisins en attente
~~3~~	
4	7
5	~~6~~, 8
6	~~3~~, ~~5~~, 9

(f) Itération 6

sommets visités	voisins en attente
~~4~~	~~7~~
5	~~8~~
6	9
7	~~4~~, 8

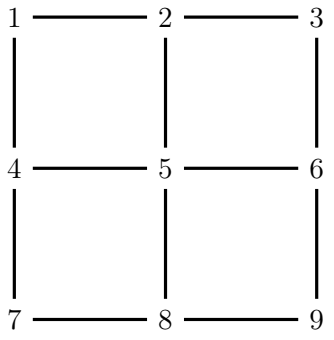
(g) Itération 7

sommets visités	voisins en attente
~~5~~	
6	~~9~~
7	~~8~~
8	~~5~~, ~~7~~, 9

(h) Itération 8

sommets visités	voisins en attente
~~6~~	
7	
8	~~9~~
9	~~6~~, ~~8~~

(i) Itération 9



(j) La $(3, 3)$-grille

FIGURE 3.6 – Exemple d'exécution de l'algorithme sur la $(3, 3)$-grille. Ici, on a $k = 3$.

En effet, supposons que l'algorithme 3.2 se termine correctement et qu'un sommet s du graphe $\mathcal{G}$ n'est pas apparu dans la colonne des sommets visités. Supposons sans perte de généralité que v est un voisin de s qui est apparu dans la colonne des sommets visités. Un tel voisin existe puisque $\mathcal{G}$ est connexe et qu'au moins un sommet a été noté dans cette colonne.

Puisque s n'a jamais été dans la colonne des sommets visités, lorsque l'on a noté v dans cette colonne, on a aussi noté s dans la colonne des sommets en attente car s est un voisin de v. Or, comme s n'est pas apparu dans la colonne de gauche, on a que s n'a jamais quitté la mémoire de l'algorithme et donc que celui-ci n'a pas terminé avec succès à l'étape 4.f ou à l'étape 5.

Finalement, comme exactement $|V|$ sommets ont été notés dans la colonne des sommets visités et qu'aucune répétition n'est possible, on a bien que chaque sommet apparait à cet endroit une et une seule fois. □

Remarque 116 *Soit $\mathcal{G}$ un graphe connexe. Si l'algorithme 3.2 se termine correctement sur $\mathcal{G}$, alors l'ordre d'apparition des sommets dans la colonne des sommets visités est un ordre sur $\mathcal{G}$.*

Soit la fonction bijective $\lambda : V(\mathcal{G}) \to \{1, \ldots, |V(\mathcal{G})|\}$ telle que pour tout i, si s_i est le sommet chosi par l'algorithme à sa i-ème itération, alors $\lambda(s_i) = i$. Cette fonction ordonne les sommets de $\mathcal{G}$ selon l'ordre dans lequel ils ont été choisis par l'algorithme. Par la remarque 115, cette fonction est bel et bien une bijection.

Montrons par induction que λ est un ordre sur $\mathcal{G}$, c'est à dire que

$$\forall s, v \in V(\mathcal{G}) \text{ tels que } (s, v) \in E(\mathcal{G}), \quad |\lambda(s) - \lambda(v)| \leq k.$$

Soit $s_1 \in V(\mathcal{G})$, tel que $\lambda(s_1) = 1$. Supposons qu'il existe un voisin v de s_1 tel que $|\lambda(v) - \lambda(s_1)| > k$, c'est-à-dire que v est choisi plus de k itérations après s_1.

Comme le tableau de mémoire conserve seulement la trace des k itérations précédentes, la ligne de ce tableau qui contenait s_1 a été effacée avant que v ne soit choisi à l'itération $\lambda(v)$.

Comme s_1 n'apparaît plus dans la mémoire après la k-ième itération, lorsque v est choisi à la $\lambda(v)$-ième itération et est noté dans la colonne de gauche, l'algorithme ajoute s_1 dans la colonne des sommets en attente sur la même ligne que v car s_1 est un voisin de v.

Il y a ensuite deux possibilités, ou bien s_1 sera à nouveau choisi pour être noté dans la colonne de gauche, une contradiction avec l'hypothèse que l'algorithme termine correctement et la remarque 115, ou bien s_1 n'est jamais noté dans cette colonne et l'algorithme ne termine pas avec succès à l'étape 5.

Soit maintenant $i \in \{2, \ldots, n\}$. Supposons que pour tout sommet s tel que $\lambda(s) < i$, on a bien que tout voisin v de s le distance d'au plus k, c'est-à-dire que

$$\forall s \in V(\mathcal{G}) \text{ tel que } \lambda(s) < i, \text{ alors } (v, s) \in E(\mathcal{G}) \Rightarrow |\lambda(s) - \lambda(v)| \leq k.$$

Montrons que si s_i est le sommet de $\mathcal{G}$ tel que $\lambda(s_i) = i$, alors pour tout voisin v de s_i, $|\lambda(s_i) - \lambda(v)| \leq k$.

Soit donc v tel que $(v, s_i) \in E(\mathcal{G})$. Si $\lambda(v) < i$, c'est-à-dire que l'algorithme a choisi v avant de choisir s_i, alors par l'hypothèse d'induction, on a que $|\lambda(s_i) - \lambda(v)| \leq k$ car v est à distance au plus k de tous ses voisins.

Si $\lambda(v) > i$, c'est-à-dire que v est choisi après s_i, alors on peut utiliser la même argumentation que celle donnée plus haut pour montrer que tous les voisin de s_1 sont à distance au plus k. □

Remarque 117 *Soit $\mathcal{G}$ un graphe connexe tel que $bw(\mathcal{G}) \leq k$. Alors l'algorithme 3.2 peut terminer correctement sur $\mathcal{G}$.*

Si $\mathcal{G}$ est tel que $bw(\mathcal{G}) \leq k$, alors il existe un ordre $\lambda : V(\mathcal{G}) \to \{1, \ldots, |V(\mathcal{G})|\}$ de taille inférieure ou égale à k. Si l'algorithme choisit à l'itération i le sommet s tel que $\lambda(s) = i$, alors l'algorithme termine correctement.

En effet, remarquons que l'algorithme ne termine pas correctement si et seulement si l'une des trois situations suivantes arrive :

1. Plus de k lignes du tableau de mémoire sont remplies et la colonne des sommets en attente de la première ligne n'est pas vide (Étape 4.f.)
2. Il y a plus de k sommets dans la colonne des sommets en attente (Étape 4.g.)
3. Après $|V(\mathcal{G})|$ itérations, la colonne des sommets en attente n'est pas vide (Étape 5.)

Supposons que pour tout $i \in \{1, \ldots, n\}$, s_i est le sommet de $\mathcal{G}$ tel que $\lambda(s_i) = i$. Montrons qu'aucune de ces situations ne peut subvenir si l'algorithme choisit toujours à la i-ième itération le sommet s_i.

Remarquons que si l'algorithme fait de tels choix, alors sur chaque ligne du tableau de mémoire, si s_i apparait dans la colonne de gauche et s_j dans la colonne de droite, alors $i < j$.

En effet, comme $bw(\mathcal{G}) \leq k$, si $j < i$, alors $i - j \leq k$ et le sommet s_j a été visité au plus k itérations avant s_i. Il était donc encore dans le tableau de la mémoire au moment de noter s_i et s_j n'a pas pu apparaitre dans la colonne des voisins en attente.

1. Supposons donc qu'à la $k+j$-ième itération, pour $j \geq 1$, sur la première ligne, la colonne des voisins en attente n'est pas vide. Le tableau de mémoire aurait alors la forme suivante.

sommets visités	voisins en attente
s_j	s_u
s_{j+1}	$s_v, \ldots, s_w$
$\vdots$	$\vdots$
s_{j+k}	$s_r, \ldots, s_t$

   Ceci ne peut arriver que si $(s_u, s_j) \in E(\mathcal{G})$. On a vu que $u > j$ et par hypothèse, $|\lambda(s_u) - \lambda(s_j)| = u - j \leq k$. Ainsi, $j + 1 \leq u \leq j + k$ et s_u apparait dans la colonne des sommets visités. s_u ne peut donc être dans la colonne des voisins en attente.

2. Supposons qu'à l'itération j, pour $j \in \{1, \ldots, n\}$, le tableau compte plus de k sommets dans la colonne des voisins en attente.

sommets visités	voisins en attente
s_3	s_6
s_4	s_7
s_5	s_8, s_9

   Les doublons sont effacés à l'étape 4.e. de l'algorithme et après la j-ième itération, si s_i est dans la colonne des voisins en attente, alors $i > j$. On a donc que parmi les voisins en attente, on retrouve au moins un sommet s_u tel que $u > j + k$.

   Ceci implique qu'il existe un sommet s_t, avec $j-k \leq t \leq j$ et un sommet s_u avec $u > j+k$ tels que $(s_t, s_u) \in E(\mathcal{G})$. On aurait donc que $|\lambda(s_u) - \lambda(s_t)| \geq j + k + 1 - j = k + 1 > k$, une contradiction.

3. Finalement, la troisième situation ne peut pas survenir puisque nous avons vu qu'après la j-ième itération, si v est un sommet dans la colonne des voisins en attente, alors $\lambda(v) > j$. Ainsi, si v est un sommet dans la colonne des voisins en attente après la $|V(\mathcal{G})|$-ième itération, on aurait que $\lambda(v) > |V(\mathcal{G})|$, une contradiction.

Ceci termine la démonstration. □

Remarque 118 *L'algorithme 3.2 nécessite une mémoire logarithmique par rapport à la taille n du graphe $\mathcal{G}$.*

En effet, en plus du compteur *cpt* qui nécessite au plus $O(\log(n))$ bits, la seule mémoire utilisée est celle nécéssaire au stockage du tableau des sommets choisis et des sommets en attente. Ce

tableau compte au plus $k+1$ lignes. Chaque ligne contient un sommet dans la colonne de gauche, donc au plus $k+1$ sommets sont stockés à la fois dans la colonne de gauche, un espace en $O((k+1)\cdot\log(n))$. Aussi, par l'étape 4.g., il y a au plus k sommets dans la colonne de droite, un espace en $O(k\cdot\log(n))$. L'espace total est donc en $O((k+3)\cdot\log(n)) = O(\log(n))$. □

Ainsi, on a démontré que cet algorithme résout le problème k-bande en NL.

Définition 119 (k-bande-$Hom(C,-)$) *Soit C un sensemble de graphes à largeur de bande bornée par une constante k.*

- *Entrée : Un graphe $\mathcal{G} \in C$ et un graphe $\mathcal{H}$ quelconque.*
- *Problème : Déterminer si $\mathcal{G} \to \mathcal{H}$.*

Une légère modification à l'algorithme 3.2 permet de résoudre ce problème tout en restant dans NL. Il suffit de rajouter une colonne «Image» au tableau de mémoire. Lors de l'éxécution de l'algorithme 3.2 sur le graphe $\mathcal{G}$, à chaque fois qu'un sommet s est ajouté dans la colonne des sommets visités, on choisit non déterministement un sommet v de $\mathcal{H}$ que l'on note dans la nouvelle colonne sur la même ligne que s.

On vérifie ensuite que pour tout sommet s_i, s_j inscrits dans la colonne des sommets visités, si $(s_i, s_j) \in E(\mathcal{G})$, alors les sommets v_i et v_j de $\mathcal{H}$ qui figurent sur la même ligne dans la nouvelle colonne «Image» sont reliés par une arête dans $\mathcal{H}$. Si ce n'est pas le cas, on termine l'algorithme en retournant «REJETER».

Ce nouvel ajout ne demande que $k\cdot\log(|V(\mathcal{H})|)$ cases de mémoire de plus que l'espace mémoire nécéssaire à l'algorithme original. L'espace mémoire reste donc de taille logarithmique par rapport à la taille de l'entrée.

Finalement, s'il existe un homomorphisme $h : \mathcal{G} \to \mathcal{H}$ et que pour chaque sommet s placé dans la colonne des sommets visités, l'algorithme choisit de placer $h(s)$ dans la case «Image» correspondante, l'algorithme se termine en retournant «ACCEPTER».

Aussi, si le nouvel algorithme retourne «ACCEPTER» sur l'entrée $(\mathcal{G}, \mathcal{H})$, c'est que $\mathcal{G} \to \mathcal{H}$. En effet, comme chaque sommet u de $\mathcal{G}$ apparait dans la colonne des sommets visitée à une et une seule étape de l'exécution, on n'associe à u qu'un seul sommet de $\mathcal{H}$ dans la case «Image» qui lui correspond. On peut donc dire que la fonction $h : \mathcal{G} \to \mathcal{H}$ où $h(u)$ est le sommet de la case «Image» correspondant à u est une fonction bien définie.

De plus, h est un homomorphisme car si $(u, v) \in E(\mathcal{G})$, l'algorithme a vérifié que $(h(u), h(v)) \in E(\mathcal{H})$. Cette vérification est possible car comme u et v sont reliés par une arête, il y a au plus k itérations qui séparent le moment où u est visité de celui où v est visité.

Comme l'algorithme garde en mémoire $k + 1$ lignes du tableau, la valeur de $h(u)$ sera encore en mémoire lorsqu'on devra vérifier si $(h(u), h(v)) \in E(\mathcal{G})$.

Ce résultat est plus fort que celui trouvé à la fin de la section précédente puisqu'ici il n'est pas nécéssaire d'avoir un ordre sur les sommet du graphe $\mathcal{G}$ pour déterminer si $\mathcal{G} \to \mathcal{H}$.

Conclusion

Dans ce mémoire, nous avons étudié la complexité des problèmes d'homomorphisme. Dans le premier chapitre, nous avons défini différentes classes de complexité qui nous permettent de quantifier la difficulté d'un problème. Savoir qu'un problème appartient à une de ces classes nous donne la connaissance des ressources qui sont suffisantes pour le résoudre.

Par exemple, si un problème appartient à la classe L et que l'on dispose d'un espace mémoire logarithmique par rapport à la taille de notre entrée, on sait que l'on pourra résoudre ce problème.

Par la suite, pour deux types de problèmes d'homomorphisme, nous avons présenté des critères qui nous permettent d'affirmer qu'un problème répondant à ce critère appartient à une classe de complexité.

Le deuxième chapitre était à propos des problèmes d'homomorphisme de graphes où le graphe d'arrivée est fixé et où le graphe de départ est considéré comme l'entrée du problème. Pour ce problème, nous avons fait la classification suivante.

Si le graphe d'arrivée...	Alors le problème est...
contient une boucle	trivial, il y a toujours un homomorphisme
est biparti	L-complet
n'est pas biparti et ne contient pas de boucle	NP-complet

Pour les problèmes d'homomorphisme où la structure de départ appartient à un certain ensemble fixé C, nous avons vu que c'était cet ensemble qui nous permet de définir la complexité du problème.

En effet, nous avons présenté un résultat de Martin Grohe qui dit que si l'ensemble C a une largeur d'arbre bornée modulo équivalence homomorphique, alors $HOM(C, -)$ est dans P. Sinon, la version paramétrée de ce problème est dans $W[1]$.

Nous avons aussi vu que si C ne contient que des graphes à largeur de bande bornée, le problème est alors dans NL.

Cependant, le cas où C ne contient que des graphes à largeur de chemin bornée est moins

tranché. On a vu un algorithme qui permet de résoudre le problème en espace logarithmique non déterministe, mais uniquement si une décomposition en chemin du graphe de départ est déjà connue.

Serait-il possible d'améliorer cet algorithme afin qu'il ne soit plus nécéssaire de connaître une décomposition en chemin du graphe de départ ? Seul l'avenir nous le dira, et il sera sans aucun doute très intéressant de suivre les développements de la recherche en ce sens !

Bibliographie

Allender, E., M. Bauland, N. Immerman, H. Schnoor et H. Vollmer. 2009, «The complexity of satisfiability problems : Refining schaefer's theorem», *J. Comput. Syst. Sci.*, vol. 75, n° 4, doi :10.1016/j.jcss.2008.11.001, p. 245–254, ISSN 0022-0000.

Bulatov, A. A. 2002, «A dichotomy theorem for constraints on a three-element set», dans *FOCS*, p. 649–658.

Bulatov, A. A. 2005, «H-coloring dichotomy revisited», *Theor. Comput. Sci.*, vol. 349, n° 1, p. 31–39.

Bulatov, A. A., A. A. Krokhin et P. Jeavons. 2000, «Constraint satisfaction problems and finite algebras», dans *ICALP*, p. 272–282.

Dalmau, V., P. G. Kolaitis et M. Y. Vardi. 2002, «Constraint satisfaction, bounded treewidth, and finite-variable logics», dans *CP*, p. 310–326.

Feder, T. et M. Y. Vardi. 1993, «Monotone monadic snp and constraint satisfaction», dans *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, New York, NY, USA, ISBN 0-89791-591-7, p. 612–622.

Freuder, E. C. 1990, «Complexity of k-tree structured constraint satisfaction problems», dans *AAAI*, p. 4–9.

Grohe, M. 2007, «The complexity of homomorphism and constraint satisfaction problems seen from the other side», *J. ACM*, vol. 54, n° 1.

Hell, P. et J. Nešetřil. 1990, «On the complexity of h-coloring», *J. Comb. Theory, Ser. B*, vol. 48, n° 1, p. 92–110.

Jeavons, P., D. A. Cohen et M. Gyssens. 1995, «A unifying framework for tractable constraints», dans *CP*, p. 276–291.

Jeavons, P., D. A. Cohen et M. Gyssens. 1997, «Closure properties of constraints», *J. ACM*, vol. 44, n° 4, p. 527–548.

Larose, B. et P. Tesson. 2009, «Universal algebra and hardness results for constraint satisfaction problems», *Theor. Comput. Sci.*, vol. 410, n° 18, p. 1629–1647.

Schaefer, T. J. 1978, «The complexity of satisfiability problems», dans *Proceedings of the tenth annual ACM symposium on Theory of computing*, STOC '78, ACM, p. 216–226.