Académie de Montpellier
Université Montpellier II
Sciences et Techniques du Languedoc

MÉMOIRE DE STAGE DE MASTERM2

effectué au Laboratoire d'Informatique de Robotique
et de Micro-électronique de Montpellier

Spécialité : **Professionelle et Recherche unifiée en Informatique**

Bornes inférieures pour la kernelization (méthodes et outils)

par **Rémi WATRIGANT**

Date de soutenance : **1er juillet 2011**

Sous la direction de **Christophe PAUL**

Résumé

Dans le contexte de la résolution de problèmes NP-difficiles, les algorithmes polynômiaux de pré-traitement, aussi appelés algorithmes de kernelization, ont été beaucoup utilisés pour leurs bons résultats en pratique, et la complexité paramétrée offre un cadre naturel pour l'étude théorique des performances de ces derniers. Une paramétrisation d'un problème consiste à associer à chaque instance un entier naturel, appelé le paramètre, et un algorithme de kernelization réduit une instance en une instance équivalente mais de taille bornée uniquement par ce paramètre. Dans ce mémoire, on s'intéresse aux méthodes et outils permettant d'établir des bornes inférieures sur la taille des noyaux que l'on peut espérer obtenir pour un problème et une paramétrisation donnés, ainsi que la notion de hiérarchie de paramètres permettant d'étudier les limites des algorithmes de kernelization pour un problème donné. Les problèmes étudiés sont DOMINATING SET, pour lequel une démonstration plus simple est donnée pour un résultat connu, NODE DELETION FOR PROPERTY Π qui généralise des résultats pour plusieurs problèmes ainsi que K-DOMATIC PARTITION pour lequel des résultats sont donnés pour des paramétrisations plus fortes que celles déjà connues.

Abstract

In the context of NP-hard problem solving, polynomial pre-processing algorithms, also known as kernelization algorithms, have been extensively used, due to their good practical results, and the theory of parameterized complexity provides a natural framework for a mathematical analysis of their performances. A parameterization of a problem is a function that assigns an integer, called the parameter, to each instance, and a kernelization algorithm reduces a given instance to an equivalent one with a size bounded by a function of the parameter only. In this report we explore the existing methods and tools to establish lower bounds for the size of kernels that we can obtain, for a given problem and a given parameterization, and the notion of hierarchy of parameters, that permits to look at the limit of pre-processing methods for a given problem. In particular, we study the DOMINATING SET problem, for which we give a simpler proof for a known result, the NODE DELETION FOR PROPERTY Π problem, which is a generalization of results for several problems, and the K-DOMATIC PARTITION problem, for which we propose new results for stronger parameterizations.

Table des matières

Table des figures

Index des définitions de problèmes

Introduction

L'introduction de la théorie de la NP-complétude dans les années 70 [15] a rendu peu probable les espoirs de trouver des algorithmes à la fois *exacts* et *efficaces* pour résoudre toute une classe de problèmes (dits NP-difficiles). Le caractère *exact* d'un algorithme désignera le fait de pouvoir trouver grâce à lui la solution optimale du problème, et le caractère *efficace* signifera le fait de pouvoir trouver cette solution en un temps de calcul raisonnable, plus précisément grâce à un nombre d'opérations polynômial en la taille de la donnée du problème (on parlera aussi de complexité polynômiale).

A partir de ce constat, plusieurs manières d'attaquer un problème NP-difficile peuvent être mises en œuvre, chacune introduisant un nouveau pan de recherche. Une de ces manières consiste à faire des concessions concernant l'exactitude de la solution, et de trouver des algorithmes efficaces permettant de trouver une solution au problème proche de la solution optimale, avec une garantie sur le ratio entre la valeur trouvée et la valeur optimale. Ces notions sont formalisées par la théorie de l'approximabilité.

Une autre manière consiste à faire des concessions concernant l'efficacité des algorithmes mis en place. Alors qu'il est relativement facile d'élaborer des algorithmes de complexité exponentielle, il est intéressant de tenter de réduire coûte que coûte l'explosion combinatoire que cette complexité engendre. Dans cette optique, une analyse théorique précise des propriétés combinatoires du problème est nécessaire, donnant naissance à des algorithmes plus rapides, mais bien souvent complexes à mettre en œuvre. D'autre part, cette analyse du problème nous amène quelques fois à remarquer que la difficulté de la résolution d'une instance provient seulement d'une partie de celle-ci, permettant ainsi de traiter rapidement les autres parties, plus faciles. La notion d'*algorithme de pré-traitement* (ou *kernelization*) apparaît alors, permettant de réduire en un temps polynômial une instance en une instance équivalente mais de taille plus petite, en utilisant principalement des règles de réductions, supprimant les parties faciles du problème. Cependant, l'analyse théorique de ces méthodes a longtemps été délaissée, les considérant ainsi comme de simples heuristiques n'ayant pas de garantie de performance, utilisées principalement dans les solveurs SAT ou les solveurs de problèmes de satisfaction de contraintes.

L'introduction de la complexité paramétrée dans les années 90 [20] changea la donne, en fournissant un cadre naturel à l'étude théorique des algorithmes polynômiaux de pré-traitement. La paramétrisation d'un problème fournit, pour toute instance x de celui-ci, un entier naturel k appelé le paramètre. Un algorithme de kernelization pour un problème consiste alors à réduire en un temps polynômial la taille de l'instance en une fonction f quelconque du paramètre (et du paramètre seulement), en conservant évidemment l'équivalence des solutions entre l'instance retournée et celle donnée en entrée. On jugera de la pertinence d'une kernelization par la forme de la fonction f, un polynôme (ou mieux : une fonction linéaire) étant considéré comme un bon

algorithme de pré-traitement.

L'idée centrale de la complexité paramétrée est la définition de la classe *FPT*. Un problème paramétré est dit FPT si pour toute instance x munie de son paramètre k, on peut résoudre x en un temps $g(k).p(|x|)$, pour une fonction g quelconque et un polynôme p (on appelle ce type d'algorithmes des algorithmes FPT). Il est facile de voir que si un problème paramétré admet un algorithme de kernelization, alors la résolution brute-force d'une instance réduite permet de construire un algorithme FPT. Le résulat fondamental de la complexité paramétrée est que cette implication est en fait une équivalence : si un problème paramétré admet un algorithme FPT, il admet également un noyau. Cependant, la démonstration de cette implication ne construit qu'un noyau de taille exponentielle. Un challenge consiste alors à trouver des noyaux de taille la plus petite possible pour les problèmes FPT.

Dans ce mémoire, nous nous sommes intéressés aux méthodes et outils permettant de borner inférieurement la taille des noyaux qu'il est possible d'obtenir pour un problème paramétré donné. Le reste du mémoire est organisé de la façon suivante : le premier chapitre donne les définitions nécessaires pour la suite, en introduisant formellement le contexte dans lequel s'inscrit notre étude : la complexité paramétrée, ainsi que quelques notions de théorie des graphes. Une sous-section traîte de la notion de paramétrisation, en rappelant les idées classiques sur le sujet et en introduisant et formalisant le récent concept de *hiérarchie de paramètres*. Le chapitre 2 est un état de l'art sur les outils et méthodes possibles pour établir des bornes inférieures sur la taille des noyaux que l'on peut obtenir, pour un problème et une paramétrisation donnés. Le chapitre 3 utilise certaines de ces méthodes pour obtenir de nouveaux résultats ou en améliorer d'autres. On étudie en particulier les problèmes DOMINATING SET, NODE DELETION FOR PROPERTY Π et K-DOMATIC PARTITION. Enfin, l'annexe A regroupe une liste de problèmes paramétrés pour lesquels il n'existe pas de noyau polynômial sous certaines conditions. Pour chacun d'entre eux, une définition ainsi qu'une idée de la démonstration et la référence vers l'article dans lequel se trouve le résultat sont données.

Je remercie mon encadrant Christophe Paul de son aide et de ses conseils tout au long de ce stage, et plus généralement l'équipe AlGCo, permanents et doctorants, qui m'ont chaleureusement accueilli parmi eux durant ce semestre.

Chapitre 1

Complexité paramétrée, noyaux

1.1 Préliminaires, notations

Cette section décrit les bases mathématiques sur lesquelles reposent les résultats énoncés, ainsi que les principales notations qui s'y rattachent.

1.1.1 Théorie des graphes

Notations Un graphe G est caractérisé par $V(G)$ son ensemble de sommets, et $E(G) \subseteq V(G) \times V(G)$ son ensemble d'arêtes. On pourra aussi noter $G = (V, E)$, où V et E seront respectivement son ensemble de sommets et d'arêtes. Sauf mention du contraire, les graphes étudiés dans ce mémoire sont considérés simples (pas d'arêtes multiples entre sommets), sans boucles (pas d'arête d'un sommet vers lui-même) et non orientés.

Sous-structures de graphes Soit $G = (V, E)$ un graphe. On dispose des trois opérations suivantes :

i. suppression d'une arête $\{x, y\}$.

ii. suppression d'un sommet (et des arêtes adjacentes à celui-ci).

iii. contraction d'une arête (et fusion des extrémités de celle-ci en un nouveau sommet).

On a alors les définitions suivantes :
- H est un sous-graphe induit de G si H est obtenu à partir de G en effectuant des séquences de ii.
- H est un sous-graphe partiel de G si H est obtenu à partir de G en effectuant des séquences de i et ii.
- H est un mineur de G si H est obtenu à partir de G en effectuant des séquences de i, ii et iii.

La figure 1.1 donne un exemple de ces 3 définitions pour la grille 3×3.

Enfin, pour un graphe $G = (V, E)$, si X est un sous-ensemble de V, on notera $G[X]$ le sous-graphe induit en supprimant les sommets de $V \backslash X$.

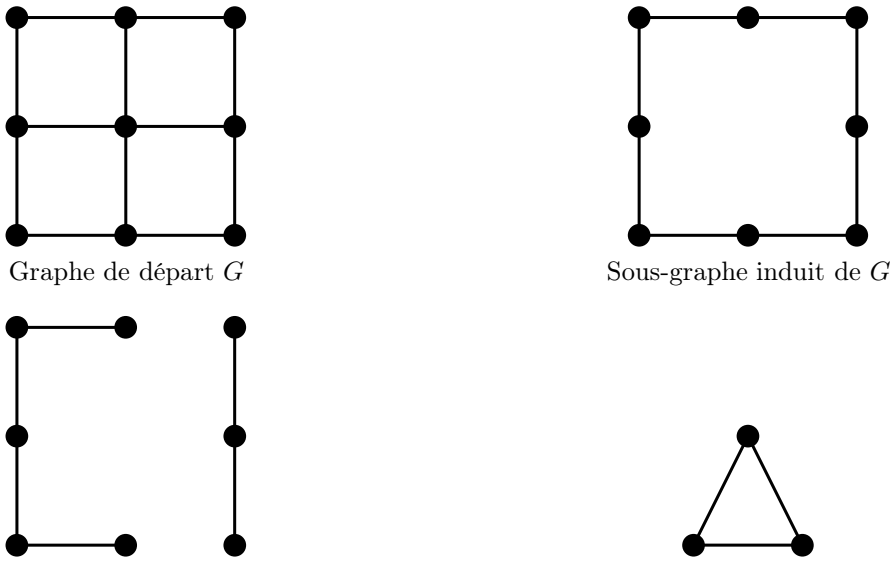


FIGURE 1.1 – Différentes définitions de sous-graphes d'un graphe

Décompositions arborescentes, tree-width Ce paragraphe définit formellement les notions de décomposition arborescente et de treewidth d'un graphe, introduites par Robertson et Seymour. Pour plus de précisions, le lecteur pourra se reporter à [40].

Définition 1. *Soit $G = (V, E)$ un graphe. Une décomposition arborescente de G est un arbre T tel que :*
- *Chaque nœud X de T correspond à un sous-ensemble de V.*
- *Si x est un sommet de G, alors l'ensemble des nœuds de T qui contiennent x est un sous-arbre (connexe) de T.*
- *Si $\{x, y\}$ est une arête de G, alors il existe un nœud X de T qui contient x et y.*

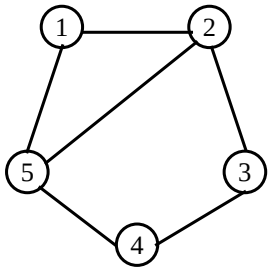
Définition 2. *Soit G un graphe et $\mathcal{T}$ une décomposition arborescente de G. On définit la largeur de $\mathcal{T}$, notée $width(\mathcal{T})$ comme la taille maximale moins 1 d'un nœud de $\mathcal{T}$.*

Définition 3. *Soit G un graphe. On définit la treewidth de G, notée $tw(G)$, comme la plus petite largeur d'une décomposition arborescente de G.*

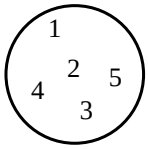
La figure 1.2 montre un exemple de décompositions arborescentes.

On remarque que les graphes de treewidth 0 sont exactement les graphes sans arêtes, les graphes de treewidth 1 sont exactement les forêts, et les graphes de treewidth 2 sont exactement les graphes dont les composantes 2-connexes sont des graphes série-parallèles. D'une manière générale, les graphes de treewidth inférieure à t sont les t-arbres partiels.

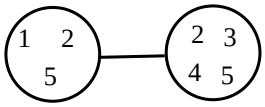
Clique-width Ce paragraphe décrit la notion de clique-width, introduite par Courcelle et al. [17].



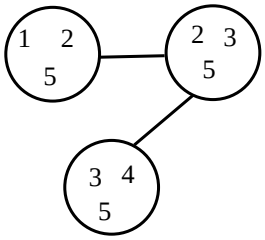
Graphe de départ G



Décomposition de largeur 4



Décomposition de largeur 3



Décomposition de largeur 2

FIGURE 1.2 – Décompositions arborescentes d'un graphe

Un étiquetage d'un graphe $G = (V, E)$ est une fonction $\gamma : V \to C$, où C est un ensemble fini (appelé ensemble des étiquettes). Un graphe étiqueté est un triplet $G = (V, E, \gamma)$ avec γ un étiquetage de (V, E).

Soit les 4 opérations suivantes :
- $create_i$ pour $i \in C$: créer un nœud étiqueté i,
- $\rho_{i\to j}$ pour $i, j \in C$: étiqueter j tous les nœuds étiquetés i.
- $\eta_{i\to j}$ pour $i, j \in C$: ajouter les arêtes entre tous les nœuds étiquetés i et tous les nœuds étiquetés j.
- $G_1 \oplus G_2$: union disjointe de deux graphes étiquetés G_1 et G_2.

Une séquence τ de ces 4 opérations précédentes est appelée *expression*, et permet de construire un graphe étiqueté. Il est facile de voir que deux graphes (non étiquetés) construits à partir d'une même expression τ sont isomorphes. On note ainsi $val(\tau)$ l'ensemble des graphes (non étiquetés) constructibles à partir d'une expression τ. On note d'autre part $T(C)$ l'ensemble des expressions constructibles à partir de l'ensemble des étiquettes C. On a alors la définition suivante :

Définition 4. *Soit $G = (V, E)$ un graphe. On appelle clique-width de G le nombre noté $cw(G)$ et défini par :*

$$cw(G) = \min\{|C| : G \in val(\tau) \text{ tel que } \tau \in T(C)\}$$

Autrement dit, c'est le nombre minimum d'étiquettes nécessaires pour construire G à partir des opérations $create_i$, $\rho_{i\to j}$, $\eta_{i\to j}$ et $\oplus$.

Il est à noter que les graphes de clique-width 2 sont exactement les cographes (graphes sans chemin induit à 4 sommets), et les arbres sont de clique-width inférieure à 3 [17]. Les cliques étant des cographes, celles-ci sont aussi de clique-width 2 : l'expression suivante permet de construire par exemple la clique K_4 : $create_1, create_2, \eta_{1\to 2}, \rho_{2\to 1}, create_2, \eta_{1\to 2}, \rho_{2\to 1}, create_2, \eta_{1\to 2}$.

1.1.2 Problèmes, problème paramétrés

Dans ce qui suit, Σ est un alphabet fini, et Σ^* désigne l'ensemble des mots finis constructibles à partir de Σ. Un problème est un ensemble $Q \subseteq \Sigma^*$, représentant exactement les instances positives de celui-ci. Le problème de décision associé à Q consiste à décider si une instance $x \in \Sigma^*$ appartient ou non à Q.

Définition 5. *Une paramétrisation de Σ^* est une fonction $\kappa : \Sigma^* \to \mathbb{N}$ calculable en temps polynômial.*

Définition 6. *Un problème paramétré est un couple (Q, κ), où $Q \subseteq \Sigma^*$ et κ est une paramétrisation de Σ^*. Une instance du problème paramétré (Q, κ) est un couple $(x, \kappa(x)) \in \Sigma^* \times \mathbb{N}$, dans lequel le second terme est appelé le paramètre de l'instance.*

Exemple Un exemple de problème paramétré est le problème du VERTEX COVER paramétré par la taille de la solution :

> VERTEX COVER
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$.
> **question :** Existe-t-il un sous-ensemble de sommets X de taille inférieure à k tel que chaque arête a au moins une extrémité dans X ?
> **paramètre :** k

Ici, une instance est un couple (G, k) composé d'un graphe et d'un entier, et la paramétrisation décrite ici est la fonction $(G, k) \mapsto k$. Une paramétrisation plus structurelle pourrait être celle de la treewidth (définie plus haut) : $(G, k) \mapsto tw(G)$.

1.2 Noyaux et FPT

Comme énoncé informellement précédemment, un noyau consiste à réduire une instance en une instance équivalente dont la taille ne va dépendre que du paramètre de la première instance. La définition formelle est la suivante :

Définition 7. *Un noyau pour un problème paramétré (Q, κ) est une fonction $K : \Sigma^* \to \Sigma^*$ telle que, pour tout $x \in \Sigma^*$:*
- *$K(x)$ est calculable en temps polynômial.*
- *$K(x) \in Q \Leftrightarrow x \in Q$.*
- *$|K(x)| \leq f(\kappa(x))$ pour une certaine fonction $f : \mathbb{N} \to \mathbb{N}$.*

On dira que (Q, κ) admet un noyau linéaire, polynômial, simple exponentiel et exponentiel si la fonction f est respectivement de la forme $f(k) = O(k)$, $f(k) = k^{O(1)}$, $f(k) = 2^{O(k)}$, $f(k) = 2^{k^{O(1)}}$. Par abus de langage, on dira que (Q, κ) admet un noyau de taille $f(k)$ (par exemple, un noyau de taille $2k^2$).

La principale définition de la théorie de la complexité paramétrée est la classe des problèmes FPT (Fixed Parameter Tractable), regroupant les problèmes paramétrés pouvant être résolus par un algorithme restreignant l'explosion combinatoire au paramètre :

Définition 8. *Un problème paramétré (Q, κ) appartient à la classe FPT si et seulement s'il existe un algorithme qui, étant donné une instance $x \in \Sigma^*$, décide si $x \in Q$ en temps $O(f(\kappa(x)).p(|x|))$,*

où $f : \mathbb{N} \to \mathbb{N}$ est une fonction quelconque, et où $p : \mathbb{N} \to \mathbb{N}$ est un polynôme. Un tel algorithme est appelé algorithme FPT.

Il est facile de voir que si un problème paramétré admet un algorithme de kernelization, alors l'application d'un algorithme brute-force sur une instance réduite constitue un algorithme FPT pour celui-ci, sa taille ne dépendant uniquement que du paramètre. Le principal résultat de la théorie de la complexité paramétrée est que la réciproque est également vraie : si un problème paramétré admet un algorithme FPT, il admet également un noyau. On a donc le résultat suivant :

Théorème 1. *[24] Un problème paramétré (Q, κ) est dans la classe FPT si et seulement s'il admet un noyau.*

Preuve.

$\Rightarrow$ Soit $(x, \kappa(x)) \in \Sigma^* \times \mathbb{N}$. Comme (Q, κ) est FPT, on peut décider x en un temps $O(p(|x|).f(\kappa(x)))$ avec p un polynôme et f une fonction quelconque.
Si $|x| \geq f(\kappa(x))$, alors l'algorithme décide x en temps $O(q(|x|))$ pour un polynôme q. Il suffit alors de retourner une instance trivialement vrai si $x \in Q$, et trivialement fausse sinon.
Si $|x| < f(\kappa(x))$, alors l'instance est déjà un noyau.
On a ainsi dans les deux cas un noyau obtenu en temps polynômial.

$\Leftarrow$ Soit $(x, \kappa(x)) \in \Sigma^* \times \mathbb{N}$, et $K : \Sigma^* \to \Sigma^*$ un noyau pour (Q, κ). Par définition, $|K(x)| \leq f(\kappa(x))$ pour une certaine fonction f, et décider si $x \in Q$ équivaut à décider si $K(x) \in Q$, or cela est réalisable en un temps $g(|K(x)|) \leq g(f(\kappa(x)))$ pour une certaine fonction g. On a donc bien que (Q, κ) appartient à la classe FPT. □

Ce théorème prouve l'existence théorique d'un noyau pour tous les problèmes FPT, cependant, il ne permet de construire que des noyaux de taille exponentielle. Le challenge de la kernelization consiste alors à chercher des noyaux de taille la plus petite possible, une taille linéaire étant le "graal" à obtenir.

1.3 Paramétrisations

Un élément important dans la démarche de résolution d'un problème à l'aide de la complexité paramétrée est le choix de la paramétrisation. En effet, face à une instance d'un problème donné, plusieurs paramètres peuvent en être extraits : partie de l'entrée, taille de la solution à trouver, invariant de graphes pour des problèmes de graphes, taille maximale des clauses pour des problèmes de satisfiabilité...etc, la seule condition à respecter est que la fonction chargée d'extraire le paramètre d'une instance doit être polynômiale. Le choix de la paramétrisation va évidemment influencer les résultats que l'on peut obtenir ; par exemple, concernant le problème INDEPENDENT SET défini par :

> INDEPENDENT SET
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$.
> **question :** Existe-t-il un sous-ensemble de sommets de taille supérieure à k tel que X induise un graphe sans arêtes ?

Lorsqu'il est paramétré par la taille de la solution à trouver, il est $W[1]$-difficile, c'est à dire qu'il est peu probable de pouvoir trouver un algorithme paramétré pour le résoudre. En revanche, il devient FPT lorsqu'il est paramétré par la treewidth du graphe d'entrée [38]. Enfin, le fait de pouvoir résoudre de manière efficace un problème à l'aide d'une certaine paramétrisation peut

aider à comprendre la difficulté intrinsèque de ce problème. Les paragraphes qui suivent donnent des exemples de paramétrisations (la liste n'est bien sûr pas exhaustive), leurs motivations, ainsi que des relations entre elles.

1.3.1 Paramétrisations "classiques" : taille de la solution

Une manière naturelle de paramétrer un problème de décision dérivé d'un problème d'optimisation est de choisir la taille de la solution à obtenir. Par exemple, pour le problème FEEDBACK VERTEX SET :

> FEEDBACK VERTEX SET
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$.
> **question :** Existe-t-il un sous-ensemble de sommets de taille inférieure à k tel que sa suppression laisse un graphe sans cycle ?
> **paramètre :** k

Ici, le paramètre choisi est k, la taille de l'ensemble X à trouver. Cette paramétrisation s'avère efficace dans le cadre de la complexité paramétrée : le problème est FPT [31], et admet même un noyau de taille polynômiale [42]. Cependant, ce n'est pas le cas de tous les problèmes :

> DOMINATING SET
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$.
> **question :** Existe-t-il un sous-ensemble de sommets X de taille inférieure à k tel que pour tout sommet x, soit $x \in X$, soit x est adjacent à un sommet $y \in X$?
> **paramètre :** k

Un résultat classique en complexité paramétrée est que ce problème est complet pour la classe $W[2]$, c'est à dire qu'il est très peu probable d'obtenir un algorithme paramétré pour cette paramétrisation. Il est donc également très peu probable d'obtenir un noyau (même de taille exponentielle).

1.3.2 Paramétrisations structurelles, distance à trivialité

Une autre manière de paramétrer un problème est d'étudier la structure et les spécificités de ses objets en entrée, et de choisir l'une d'elles habilement. Une idée [28] est de considérer les cas "faciles" de ces problèmes, et de prendre comme paramètre la distance entre l'objet d'entrée et ce cas facile. La difficulté consiste alors d'une part à identifier les cas triviaux, et d'autre part à définir une bonne distance entre l'entrée du problème et ces derniers. Des exemples simples sont les problèmes de graphes, où beaucoup de problèmes deviennent faciles si le graphe d'entrée appartient à une classe de graphes particulière, comme par exemple les arbres, les graphes bipartis, les cliques...etc. Prenons par exemple le problème suivant :

> CLIQUE
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$.
> **question :** Existe-t-il une clique de taille supérieure à k dans G ?

Lorsque paramétré par la taille de la clique à trouver, ce problème a été montré comme $W[1]$-difficile [20]. Cependant, un cas trivial apparaît lorsque G est une union disjointe de cliques : un *graphe cluster* : en effet, un simple parcours des adjacences des sommets permet d'exhiber la présence ou non d'une clique d'une certaine taille. A partir de ce constat, on peut alors tenter de définir la distance entre un graphe donné et un graphe cluster. Une distance naturelle est de considérer le nombre d'arêtes à enlever pour le transformer en un tel graphe, et il suffit d'ajouter

à l'entrée du problème un tel ensemble d'arêtes. Muni de cette paramétrisation, le problème devient FPT [28].

Une autre catégorie de distances à trivialité à considérer, importante en théorie des graphes, est la notion de *-width, comme par exemple la tree-width, clique-width, path-width...etc. Ces dernières mesurent d'une certaine manière la distance entre un graphe et une classe de graphes particulière, la classe des arbres pour la treewidth par exemple. Certains problèmes, tels que VERTEX COVER, pouvant être résolus efficacement dans cette dernière, il est intéressant de considérer comme paramètre la treewidth du graphe en entrée. Les résultats de Courcelle [16] vont d'ailleurs dans ce sens, puisque beaucoup de problèmes de graphes deviennent polynômiaux sur des graphes à treewidth bornée. Plus précisément cela concerne tous les problèmes s'exprimant à l'aide de la logique monadique de second ordre (MSO), c'est à dire avec des propositions utilisant des opérateurs logiques ($\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$) et des quantificateurs ($\forall$, $\exists$) sur des variables (sommets pour le cas des graphes par exemple) et des ensembles de variables (arêtes, ensembles de sommets, ensembles d'arêtes pour le cas des graphes par exemple). C'est par exemple le cas de problèmes tels que VERTEX COVER, 3-COLORATION, CIRCUIT HAMILTONIEN DANS LES GRAPHES NON ORIENTES, mais ce n'est pas le cas de problèmes tels que CIRCUIT HAMILTONIEN DANS LES GRAPHES ORIENTES.

Une précision à apporter concernant les paramétrisations structurelles de problèmes est que celles-ci doivent pouvoir être calculées en temps polynômial, pour respecter la définition d'une part, et pour pouvoir avoir un intérêt pratique d'autre part. Or, calculer la treewidth d'un graphe donné est un problème NP-difficile, tout comme le calcul du vertex cover minimum ou d'autres ensembles de sommets à retirer pour appartenir à une classe de graphes donnée. Cependant, le lecteur remarquera que ces problèmes admettent dans la plupart des cas des algorithmes approximables à facteur constant (une 2-approximation pour le vertex cover par exemple). Ainsi, cela a du sens de considérer de telles paramétrisations, un noyau de taille polynômiale en la taille d'une structure approchée à facteur constant de l'optimum sera également un noyau de taille polynômiale en la valeur optimale de la structure recherchée. Finalement, il suffit que le problème associé à la paramétrisation choisie appartienne à la classe Poly-APX (contenant APX) des problèmes admettant un algorithme polynômial calculant une solution inférieure (ou supérieure pour un problème de maximisation) en un polynôme de la valeur de la solution optimale. On peut même, d'un point de vue pratique, considérer des paramétrisations dont une valeur optimale peut être calculée par un algorithme FPT, comme un ensemble d'arêtes à supprimer pour obtenir un graphe cluster pour le problème précédent CLIQUE[8], ou bien des paramétrisations dont une solution approchée peut être calculée par un algorithme FPT, comme la treewidth.

1.3.3 Hiérarchie de paramètres

Comme on a pu le voir, du choix du paramètre va dépendre la difficulté de résolution d'un problème. Intuitivement, plus un paramètre aura une grande valeur, plus il sera facile de l'aborder à l'aide des outils de la complexité paramétrée, car on se permet alors "plus d'explosion combinatoire". Il est également intéressant d'observer l'impact de ce choix sur la taille des noyaux que l'on peut obtenir, lorsqu'il y en a un (classe des problèmes FPT). L'intuition précédente se vérifie d'une certaine manière, comme le montre la proposition précédente :

Proposition 1. *Soit $P \subseteq \Sigma^*$ un problème et $\kappa_1, \kappa_2 : \Sigma^* \to \mathbb{N}$ deux paramétrisations de P telles que $\forall x \in \Sigma^*$, $p(\kappa_1(x)) \geq \kappa_2(x)$ pour un certain polynôme $p : \mathbb{N} \to \mathbb{N}$. Si (P, κ_2) admet un noyau polynômial, alors (P, κ_1) admet un noyau polynômial.*

Preuve. On note $\mathcal{A}$ l'algorithme de kernelization de (P, κ_2), et soit $x \in \Sigma^*$. $\mathcal{A}(x)$ est une instance équivalente à x telle que $|\mathcal{A}(x)| \leq q(\kappa_2(x))$ pour un certain polynôme q, mais comme $\kappa_2(x) \leq p(\kappa_1(x))$, on a $|\mathcal{A}(x)| \leq q(p(\kappa_1(x)))$, d'où le problème (P, κ_1) admet un noyau polynômial. □

Un exemple d'un tel ordre entre les paramétrisations est le cas du vertex cover et de la treewidth d'un graphe. En effet, si un graphe $G = (V, E)$ admet un vertex cover X de taille k, alors il admet une décomposition arborescente de taille k : il suffit pour cela de calculer une décomposition arborescente de $G[V \backslash X]$, puis d'ajouter X dans tous les nœuds de celle-ci. Le calcul de la première décomposition est triviale, car $G[V \backslash X]$ est un ensemble indépendant, et celle-ci aura une largeur égale à 0 (chaque nœud de l'arbre ne contient qu'un sommet). En ajoutant X, on a bien une décomposition arborescente de taille k. On obtient alors le corollaire suivant :

Corollaire 1. *Soit Q un problème de graphes, i.e. un problème dont les instances acceptables contiennent un graphe. Si Q admet un noyau polynômial lorsqu'il est paramétré par largeur d'une décomposition arborescente du graphe d'entrée, alors il admet un noyau polynômial lorsqu'il est paramétré la la taille d'un vertex cover de ce graphe.*

Ce corollaire pouvant notamment être utilisé de la manière suivante pour le problème DOMINATING SET défini par :

> DOMINATING SET
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$.
> **question :** Existe-t-il $D \subseteq V$ avec $|D| \leq k$ et tel que pour tout $x \in V$, soit $x \in D$, soit x est adjacent à un sommet $y \in D$?

Proposition 2. DOMINATING SET *n'admet probablement pas de noyau polynômial lorsqu'il est paramétré par la treewidth du graphe d'entrée.*

Preuve. En effet, la contraposée du corollaire précédent et le fait que DOMINATING SET n'admette probablement pas de noyau polynômial lorsqu'il est paramétré par la taille d'un vertex cover ([19], ou bien théorème 11 pour une démonstration alternative) permettent d'obtenir le résultat. L'utilisation du mot "probablement" sera éclaircie au prochain chapitre. □

Pour le cas du problème du VERTEX COVER, celui-ci n'admet probablement pas de noyau polynômial s'il est paramétré par la treewidth [1], mais il en admet un s'il est paramétré par la taille d'un vertex cover (taille de la solution) [11], et il en admet également un s'il est paramétré par la taille d'un feedback vertex set [30]. On peut généraliser ceci en introduisant l'invariant $d_{tw \leq t}$ défini pour tout $t \in \mathbb{N}$ et pour tout graphe $G = (V, E)$ par :

$$d_{tw \leq t}(G) = min\{|X| : X \subseteq V \wedge tw(G[V \backslash X]) \leq t\}$$

Autrement dit, $d_{tw \leq t}(G)$ est le nombre minimum de sommet à enlever afin que G soit de treewidth inférieure à t.
On remarque alors que la taille minimum d'un vertex cover d'un graphe G est égale à $d_{tw \leq 0}(G)$, et que la taille minimum d'un feedback vertex set d'un graphe G est égale à $d_{tw \leq 1}(G)$. On a enfin la hiérarchie suivante, valable pour tout graphe G :

$$\underbrace{d_{tw \leq 0}(G)}_{\text{vertex cover}} \geq \underbrace{d_{tw \leq 1}(G)}_{\text{feedback vertex set}} \geq ... \geq d_{tw \leq t}(G) \geq tw(G) - t$$

On a alors le problème ouvert suivant :

Problème ouvert 1. *Existe-t-il $t \geq 2$ tel que* VERTEX COVER *n'admette pas de noyau polynômial s'il est paramétré par $d_{tw \leq t}(G)$, où G est le graphe d'entrée ?*

Concernant la version pondérée WEIGHTED VERTEX COVER, dans lequel une fonction de poids sur les sommets est donnée en entrée, et où le but n'est plus de trouver un vertex cover d'une certaine taille, mais d'un certain poids, le problème est fermé : en effet, celui-ci admet un noyau linéaire lorsqu'il est paramétré par la taille de la solution (et donc un vertex cover)[13], mais il n'en admet probablement pas lorsqu'il est paramétré par la taille d'un feedback vertex set (ni même par la taille d'un ensemble dont la suppression laisse un graphe avec des sommets de degré au plus 1, qui est "entre" le vertex cover et le feedback vertex set, dans la hiérarchie).

1.3.4 Le cas des graphes : distance à classe de graphe donnée et modulateurs

Dans le cas des graphes, nous avons pu voir que certains problèmes étant décidables efficacement sur des classes de graphes particulières, une paramétrisation intéressante est de considérer la distance à cette classe de graphe. Ce concept est généralisé par la notion de *modulateur*, introduite par Cai [9] :

Définition 9. *Soit $\mathcal{C}$ une classe de graphes donnée et $G = (V, E)$ un graphe quelconque. Un modulateur de G pour la classe $\mathcal{C}$ est un ensemble $X \subseteq V$ tel que $G[V \backslash X] \in \mathcal{C}$.*

Par exemple, un vertex cover pour un graphe représente un modulateur pour la classe des graphes sans arêtes, et un feedback vertex set représente un modulateur pour la classe des forêts.

Etant donné un problème de graphes Q et une classe de graphes $\mathcal{C}$, on définit la paramétrisation $\mathcal{C}{+}kv : \Sigma^* \to \mathbb{N}$ représentant la taille d'un modulateur des graphes d'entrée pour la classe $\mathcal{C}$, autrement dit, le nombre de sommets à enlever au graphe d'entrée pour appartenir à la classe $\mathcal{C}$. De manière encore plus générale, on peut utiliser les paramètres suivants :

- $\mathcal{C}{+}kv$: nombre de sommets à enlever pour appartenir à $\mathcal{C}$.
- $\mathcal{C}{-}kv$: nombre de sommets à ajouter pour appartenir à $\mathcal{C}$.
- $\mathcal{C}{+}ke$: nombre d'arêtes à enlever pour appartenir à $\mathcal{C}$.
- $\mathcal{C}{-}ke$: nombre d'arêtes à ajouter pour appartenir à $\mathcal{C}$.
- $\mathcal{C}{\pm}ke$: nombre de modifications d'aretes pour appartenir à $\mathcal{C}$.

Afin d'utiliser le concept de hiérarchie de paramètres présenté précédemment, on peut utiliser la proposition suivante :

Proposition 3. *Soit $\mathcal{A}$ et $\mathcal{B}$ deux classes de graphes telles que $\mathcal{A} \subseteq \mathcal{B}$, et $G = (V, E)$ un graphe quelconque. Alors $\mathcal{B}{+}kv(G) \leq \mathcal{A}{+}kv(G)$.*

Preuve. En effet, en remarquant qu'un modulateur pour $\mathcal{A}$ est également un modulateur pour $\mathcal{B}$, le résultat est immédiat. □

Ce qui nous donne la proposition suivante concernant l'existence de noyau polynômial :

Proposition 4. *Soit $\mathcal{A}$ et $\mathcal{B}$ deux classes de graphes telles que $\mathcal{A} \subseteq \mathcal{B}$, et $Q \subseteq \Sigma^*$ un problème de graphes. Si Q admet un noyau polynômial s'il est paramétré par $\mathcal{B}{+}kv$, alors il admet également un noyau polynômial s'il est paramétré par $\mathcal{A}{+}kv$.*

Par exemple, la classe des graphes sans arêtes étant incluse dans la classe des forêts, on a pour tout graphe G l'inégalité $Forests{+}kv(G) \leq Independent\ set{+}kv(G)$ (un vertex cover est

également un feedback vertex set). Ainsi, démontrer l'existence d'un noyau pour un problème de graphes paramétré par la taille d'un feedback vertex set implique l'existence d'un noyau pour ce même problème paramétré par la taille d'un vertex cover.

La figure 1.3 utilise cette idée pour établir une hiérarchie entre plusieurs paramètres issus de classes de graphes et invariants de graphes classiques. La liste suivante apporte certaines précisions sur des classes de graphes utilisées :

- La classe Edges & vertices est la classe des graphes dont les sommets ont un degré au plus 1.
- La classe Linear forest est la classe des graphes étant l'union de chemins.
- La classe C-split est la classe des graphes dont les composantes connexes sont des split graphs, autrement dit des graphes dont l'ensemble des sommets peut être partitionné en deux ensembles induisant respectivement une clique et un ensemble indépendant.
- La classe Cluster est la classe des graphes dont les composantes connexes sont des cliques.
- La classe Co-cluster est la classe des graphes étant des complémentaires de cluster graphs.

Une flèche d'une paramétrisation A vers une paramétrisation B signifie que pour tout graphe G, on a $B(G) \leq p(A(G))$ pour un polynôme $p : \mathbb{N} \to \mathbb{N}$.

Motivations L'idée d'établir une telle hiérarchie entre ces paramétrisations est d'explorer les limites des algorithmes polynômiaux de pré-traitement pour un problème. En effet, pour un problème donné, si un noyau polynômial est trouvé pour une paramétrisation κ, il est intéressant d'étudier la taille des noyaux lorsque l'on utilise des paramétrisations plus petites. D'un autre côté, si une borne inférieure est établie pour une certaine paramétrisation, on s'intéressera alors à l'existence de noyaux pour des paramétrisations plus grandes.

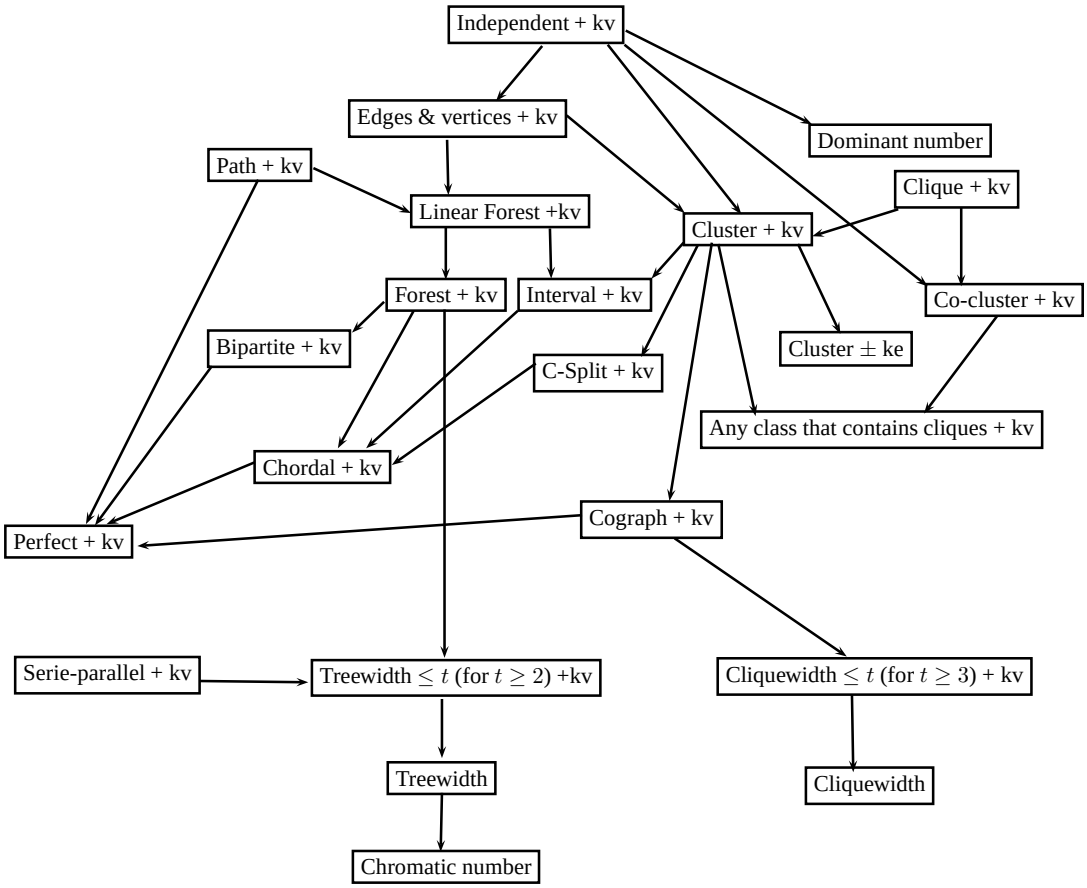


FIGURE 1.3 – Hiérarchie de paramétrisations pour des classes de graphes et invariants de graphes classiques

Chapitre 2

Méthodes de bornes inférieures de noyaux

Dans ce mémoire, on s'intéresse principalement à l'existence de noyaux polynômiaux. En effet, il est pertinent de savoir si, pour un problème et une paramétrisation donnés, celui-ci peut être attaqué en pratique à l'aide de routines de pré-traitement s'exécutant en temps polynômial. Cependant, la tâche consistant à caractériser exactement l'ensemble des problèmes admettant un noyau de taille polynômiale semble compromise, car prouver qu'un problème paramétré n'admet pas de tel noyau impliquerai $P \neq NP$. En effet, si $P = NP$, alors il existe un algorithme polynômial pour décider un problème Q, et il suffirait de retourner une instance trivialement vraie ou fausse selon qu'elle appartienne à Q ou non.

Les sous-sections qui suivent constituent un état de l'art des outils permettant de montrer la non-existence de noyau polynômial sous certaines hypothèses, ou bien de montrer la non existence de noyau d'une certaine forme. Dans chaque cas, les preuves classiques des résultats sont données, ainsi que les références vers les articles introduisant ces notions.

2.1 Non-existence de noyaux polynômiaux sous certaines hypothèses

2.1.1 OU-compositions

Méthode

Les OU-compositions [1] constituent les premiers outils ayant permis d'établir, sous certaines hypothèses, des bornes inférieures pour la kernelization. Ces hypothèses concernent des notions de complexité classiques et s'appuient sur un résultat de [25] concernant des algorithmes de *distillation* pour des problèmes NP-complets. Une définition analogue de ces algorithmes dans la théorie de la complexité paramétrée est ensuite proposée afin d'obtenir les résultats sur les noyaux.

Définition 10. *Soit $Q \subseteq \Sigma^*$ un problème et $t \in \mathbb{N}$. On définit l'ensemble $OR(Q)$ par*

$$OR(Q) := \{(x_1, ..., x_t) \in (\Sigma^*)^t : \exists i \in \{1, .., t\} : x_i \in Q\}$$

Autrement dit, c'est l'ensemble des séquences d'instances de Q contenant une instance positive.

Définition 11. *Soit $A, B \subseteq \Sigma^*$. Un algorithme de OU-distillation de A vers B est une fonction calculable $\mathcal{D}$ telle que pour toute séquence finie $(x_1, ..., x_t) \in (\Sigma^*)^t$ avec $n = \max_{i=1..t} |x_i|$, $\mathcal{D}$ retourne un élément de Σ^* tel que :*

- *$\mathcal{D}(x_1, ..., x_t)$ se calcule en temps polynômial en $t * n$.*
- *$\mathcal{D}(x_1, ..., x_t) \in B \Leftrightarrow (x_1, ..., x_t) \in OR(A)$.*
- *$|\mathcal{D}(x_1, ..., x_t)| \leq p(n)$ pour un polynôme $p : \mathbb{N} \to \mathbb{N}$.*

Autrement dit, un algorithme de OU-distillation prend en entrée une séquence d'instances, et retourne en temps polynômial une instance de taille bornée par un polynôme de la plus grande instance d'entrée, et qui est positive si et seulement si une des instances d'entrée est positive.

Le théorème sur lequel se basent les résultats obtenus par la suite est le suivant :

Théorème 2. *[25] (admis) Soit $A \subseteq \Sigma^*$ un problème NP-difficile. S'il existe un algorithme de OU-distillation de A vers un ensemble $B \subseteq \Sigma^*$, alors $PH = \Sigma_p^3$ (la hiérarchie polynômiale s'effondre au troisième niveau).*

Remarquons que ce théorème traduit le fait qu'il est très peu probable qu'un problème NP-complet admette un algorithme de OU-distillation, car un effondrement de la hiérarchie polynômiale au 3ème niveau (et donc des niveaux supérieurs), même s'il n'impliquerai pas $P = NP$, est considéré comme invraisemblable. Ceci explique l'utilisation précédente du mot "probablement".

Les deux définitions qui suivent permettent d'établir le lien entre complexité classique et paramétrée.

Définition 12. *Soit (Q, κ) un problème paramétré. On appelle version non paramétrée de (Q, κ) le problème (classique) $\tilde{Q} \subseteq \Sigma^*$ défini par $\tilde{Q} := \{x\#1^{\kappa(x)} : x \in Q\}$, où $\#$ représente le caractère blanc, $1 \in \Sigma$ est un caractère quelconque et $1^p = \underbrace{1...1}_{p}$ pour $p \in \mathbb{N}$.*

Remarquons qu'une instance du problème classique est obtenue simplement en concaténant le paramètre codé en écriture unaire à l'instance.

On adapte ensuite la définition d'algorithme de OU-distillation pour les problèmes paramétrés :

Définition 13. *Un algorithme de OU-composition pour un problème paramétré (Q, κ) est une fonction calculable $\mathcal{C}$ telle que pour toute séquence $(x_1, .., x_t) \in (\Sigma^*)^t$ avec $\kappa(x_i) = \kappa(x_j) =: k$ pour tout $1 \leq i, j \leq t$, $\mathcal{C}$ retourne un élément de Σ^* tel que :*

- *$\mathcal{C}(x_1, .., x_t)$ est calculable en temps polynômial en $\sum_{i=1}^{t} |x_i| + k$.*
- *$\mathcal{C}(x_1, ..., x_t) \in Q \Leftrightarrow (x_1, ..., x_t) \in OR(Q)$.*
- *en posant $k^* := \kappa(\mathcal{C}(x_1, .., x_t))$, on a $k^* \leq p(k)$ pour un polynôme $p : \mathbb{N} \to \mathbb{N}$.*

On dit qu'un problème paramétré (Q, κ) est OU-composable s'il existe un algorithme de OU-composition pour (Q, κ).

Remarquons ici qu'un algorithme de composition prend en entrée des instances ayant la même valeur de paramètre, et que le paramètre de l'instance retournée est inférieure à un polynôme du paramètre des instances d'entrée. Enfin, comme pour un algorithme de OU-distillation, l'instance

retournée est positive si et seulement si une des instance d'entrée l'est.

On a ensuite le théorème suivant, montrant qu'il est peu probable qu'il existe un algorithme de OU-composition pour un problème paramétré ayant un noyau polynômial :

Théorème 3. *[1] Soit (Q, κ) un problème paramétré OU-composable tel que $\tilde{Q}$ est NP-complet. Si Q admet un noyau polynômial, alors il existe un algorithme de OU-distillation de $\tilde{Q}$ vers lui-même.*

Preuve. Soit (Q, κ) un problème paramétré composable tel que $\tilde{Q}$ est NP-complet et tel que (Q, κ) admette un noyau polynômial. Montrons qu'il existe un algorithme de OU-distillation de $\tilde{Q}$ vers lui-même.

- Comme $\tilde{Q}$ est NP-complet, nous disposons des deux réductions suivantes :
  · $\Phi : \Sigma^* \to \Sigma^*$ telle que $x \in \tilde{Q} \Leftrightarrow \Phi(x) \in \text{SAT}$
  · $\Psi : \Sigma^* \to \Sigma^*$ telle que $x \in \text{SAT} \Leftrightarrow \Psi(x) \in \tilde{Q}$

- Comme (Q, κ) est composable, on a l'algorithme de OU-composition $\mathcal{C} : (\Sigma^*)^t \to \Sigma^*$

- Comme (Q, κ) admet un noyau polynômial, on a enfin l'algorithme de kernelization $\mathcal{K}$: $\Sigma^* \to \Sigma^*$

Soit $(x_1, ..., x_t) \in (\Sigma^*)^t$. L'algorithme suivant permet de construire une instance équivalente (dans le sens d'une distillation) à la séquence d'instances précédente :

Etape (1) Pour $i = 1...t$, le paramètre $k_i := \kappa(x_i)$ prend sa valeur dans $\{v_1, ..., v_r\}$. On regroupe les instances ayant la même valeur de paramètre entre elles. En notant L_i le nombre d'instances ayant leur paramètre de valeur v_i, on a la séquence suivante :

$$\left[\left((x^1_{v_1}, v_1), ..., (x^{L_1}_{v_1}, v_1) \right), ..., \left((x^1_{v_r}, v_r), ..., (x^{L_r}_{v_r}, v_r) \right) \right]$$

Etape (2) Pour chaque séquence $\big((x^1_{v_i}, v_i), ..., (x^{L_i}_{v_i}, v_i)\big)$ on applique l'algorithme de composition $\mathcal{C}$ qui nous donne une instance (y_i, k'_i). On a donc la séquence suivante :

$$((y_1, k'_1), ..., (y_r, k'_r))$$

Etape (3) Pour chaque instance (y_i, k'_i), on applique l'algorithme de kernelization $\mathcal{K}$ qui nous donne une instance (z_i, k''_i). On a donc la séquence suivante :

$$((z_1, k''_1), ..., (z_r, k''_r))$$

Etape (4) Chaque instance paramétrée (z_i, k''_i) est transformée en son équivalent dans $\tilde{Q}$: $z_i \# 1^{k''_i}$. On a la séquence suivante :

$$(z_1 \# 1^{k''_1}, ..., z_r \# 1^{k''_r})$$

Etape (5) Chaque instance $z_i \# 1^{k''_i}$ est transformée en une instance de SAT à l'aide de Φ. On a la séquence suivante :

$$(\Phi(z_1 \# 1^{k''_1}), ..., \Phi(z_r \# 1^{k''_r}))$$

Etape (6) On construit enfin une instance de $\tilde{Q}$ ainsi :

$$\Psi(\bigvee_{i=1}^{r} \Phi(z_i \# 1^{k''_i}))$$

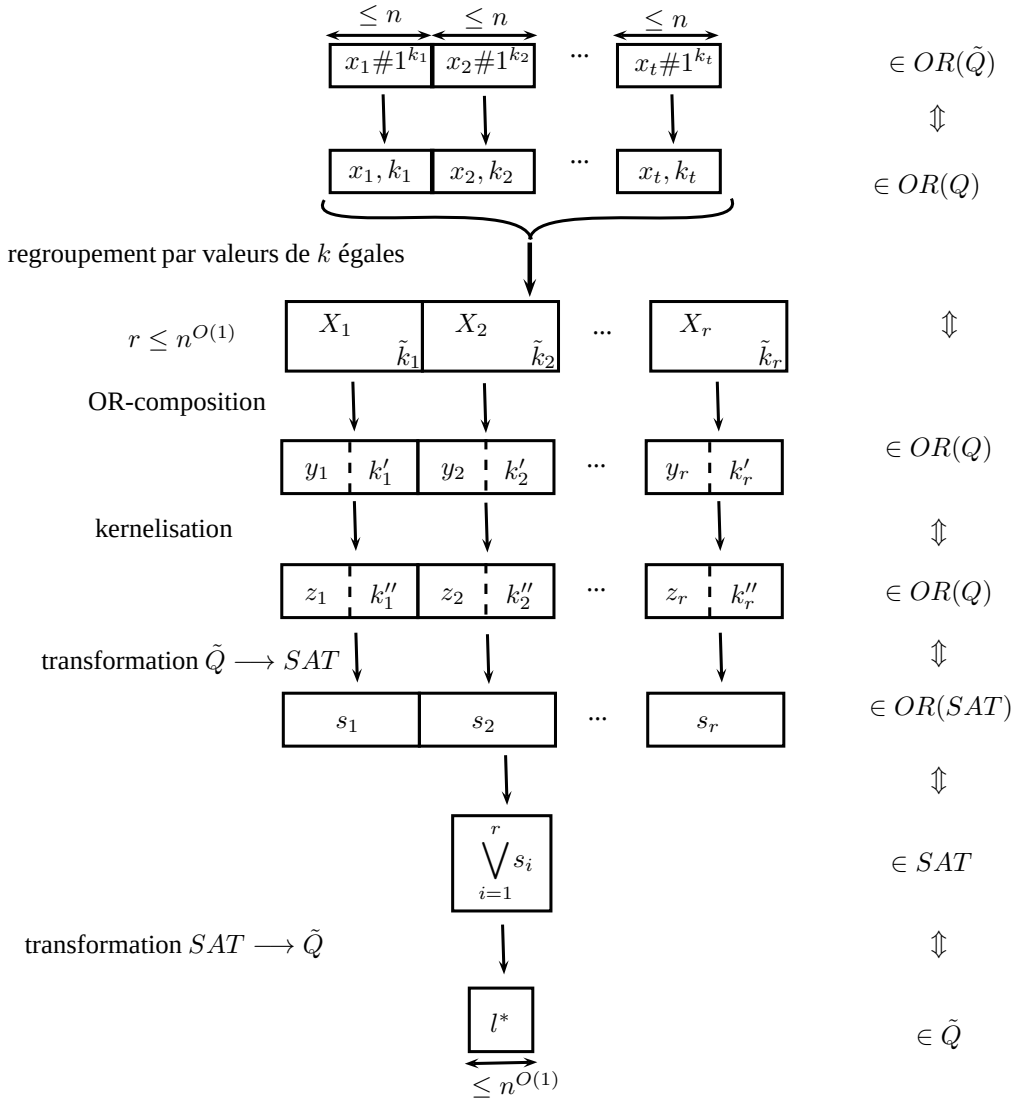


FIGURE 2.1 – Schéma de la preuve du théorème 3

La figure 2.1 résume la construction réalisée.
Montrons que cet algorithme est bien un algorithme de OU-distillation :

– Tout d'abord, l'instance obtenue est bien équivalente à la séquence d'instances en entrée :

$$\begin{aligned}
\Psi(\bigvee_{i=1}^{r}\Phi(z_i\#1^{k''_i}))\in\tilde{Q} &\Leftrightarrow \bigvee_{i=1}^{r}\Phi(z_i\#1^{k''_i})\in \text{SAT}\\
&\Leftrightarrow \exists i\in\{1,..,r\}:\Phi(z_i\#1^{k''_i})\in \text{SAT}\\
&\Leftrightarrow \exists i\in\{1,..,r\}:z_i\#1^{k''_i}\in\tilde{Q}\\
&\Leftrightarrow \exists i\in\{1,..,r\}:z_i\in Q\\
&\Leftrightarrow \exists i\in\{1,..,r\}:y'_i\in Q\\
&\Leftrightarrow \exists i\in\{1,..,r\}:\exists j\in\{1,..,L_i\}:x^j_{v_i}\in Q\\
&\Leftrightarrow \exists i\in\{1,..,t\}:x_i\in Q\\
&\Leftrightarrow \exists i\in\{1,..,t\}:x_i\#1^{k_i}\in\tilde{Q}
\end{aligned}$$

– Ensuite, par définition, les algorithmes Φ, Ψ, $\mathcal{C}$ et $\mathcal{K}$ sont polynômiaux en la taille de leur(s) entée(s) respectives, donc l'algorithme construit est polynômial.

– Il reste à montrer que l'instance retournée est polynômiale en $\max_{i=1..t}|\tilde{x_i}|$:
  – Tout d'abord, comme $\{v_1,...,v_r\}$ contient les valeurs distinctes que prennent les k_i, on a $r\leq k=\max_{i=1..t}(k_i)\leq n$. On a donc au plus n instances dans les étapes Etape (2) à Etape (5).
  – Comme $\mathcal{C}$ est un algorithme de OU-composition, chaque k'_i obtenu à l'étape Etape (2) est polynômial en $v_i\leq k\leq n$.
  – Comme $\mathcal{K}$ est une kernelization polynômiale, la taille de chaque $(z_i,k''_i)=z_i\#1^{k''_i}$ est bornée par un polynôme en k'_i. Il s'en suit que chaque $z_i\#1^{k''_i}$ est de taille polynômiale en n. Φ et Ψ étant des réductions polynômiales, l'instance retournée par notre algorithme est de taille polynômiale en n.

□

En combinant les théorèmes 2 et 3, on obtient le corollaire suivant :

Corollaire 2. *Sauf si $PH=\Sigma^3_p$, un problème paramétré Q OU-composable tel que $\tilde{Q}$ est NP-complet n'a pas de noyau polynômial.*

Ainsi, afin de montrer qu'il est peu probable d'avoir un noyau polynômial pour un problème paramétré, il suffit de montrer que sa version classique est NP-complète, et qu'il existe un algorithme de OU-composition pour ce problème.

Exemple

L'exemple classique de OU-composition est le problème du K-PATH :

> K-PATH
> **entrée :** Un graphe $G=(V,E)$, $k\leq|E|$.
> **question :** Existe-t-il un chemin de longueur k dans G ?
> **paramètre :** k

Tout d'abord, la version non paramétrée de ce problème est NP-complète [26]. Ensuite, étant donné t instances $((G_1,k),...,(G_t,k))$ ayant toutes le même paramètre k, il suffit de retourner

l'union disjointe de ces graphes, ainsi que ce même k en paramètre. Cette construction est bien réalisée en temps polynômial, et il existe un chemin de longueur k dans l'union disjointe si et seulement s'il existe un chemin de longueur k dans un des graphes d'entrée. Enfin, le paramètre est bien inférieur à un polynôme du premier puisqu'il reste inchangé. On a donc la proposition suivante :

Proposition 5. *Le problème* K-PATH *paramétré par la longueur du chemin à trouver n'admet pas de noyau polynômial, sauf si* $PH = \Sigma_p^3$.

En fait, plus généralement, si L est un problème prenant en entrée un couple (G, k), avec G un graphe et $k \in \mathbb{N}$ tel que pour deux instances (G_1, k) et (G_2, k) on a $(G_1 \cup G_2, k) \in L \Leftrightarrow (G_1, k) \in L \vee (G_2, k) \in L$, alors, grâce à la même idée que précédemment, L paramétré par k n'admet pas de noyau polynômial sauf si $PH = \Sigma_p^3$.

ET-compositions

Alors que certains problèmes comme K-PATH admettent une OU-composition naturelle, ce n'est pas le cas d'autres problèmes, tels que TREEWIDTH $\leq$ K :

TREEWIDTH $\leq$ K
entrée : Un graphe $G = (V, E)$, $k \leq |V|$.
question : $tw(G) \leq k$?
paramètre : k

En effet, soit $((G_1, k), ..., (G_t, k))$ une séquence d'instances de ce problème, et posons $G' = \bigcup_{i=1}^{t} G_i$ l'union disjointe des G_i. G' aura une treewidth inférieure à k si et seulement si pour tout $i = 1..t$, G_i a une treewidth inférieure à k. L'algorithme ainsi construit est appelé algorithme de ET-composition, dans le sens où seul le quantificateur universel remplace l'existentiel dans la définition de la OU-composition.

On peut alors tenter d'étendre les notions précédentes au cas des ET-compositions : une ET-composition pour un problème paramétré dont la version classique est NP-complète implique un algorithme de ET-distillation pour un problème NP-complet, lequel impliquerait également l'existence d'un algorithme de OU-distillation pour les problèmes coNP-complets [1]. Cependant, ce résultat n'est pas (pour l'instant) connu pour impliquer une chose inattendue en complexité telle que $PH = \Sigma_p^3$, et la question de savoir s'il existe un problème NP-complet admettant un algorithme de ET-distillation est à ce jour encore ouverte. On a donc simplement le théorème suivant, moins fort que sa "version conjonctive" :

Théorème 4. *[1] Soit (Q, κ) un problème admettant un algorithme de ET-composition et tel que $\tilde{Q}$ est NP-complet. Alors (Q, κ) n'admet pas de noyau polynômial, sauf si tous les problèmes coNP-complets admettent un algorithme de OU-distillation.*

Ainsi que l'un des principaux problèmes ouverts dans le domaine de la kernelization :

Problème ouvert 2. *Existe-t-il un problème paramétré (Q, κ) dont la version non paramétrée est NP-complète et qui admette à la fois un algorithme de ET-composition et un noyau polynômial ?*

Dans le cas contraire, une contradiction avec une hypothèse de la théorie de la complexité, du même type que $PH = \Sigma_p^3$, serait satisfaisante.

2.1.2 Transformations paramétrées polynômiales

Les transformations paramétrées polynômiales, introduites par [6], permettent entre autres de transmettre l'existence ou non d'un noyau polynômial d'un problème vers un autre problème. Celles-ci diffèrent en revanche des réductions paramétrées [24] permettant par exemple de montrer l'appartenance d'un problème à une classe de la W-hiérarchie.

Définition 14. *Soit (P, κ_P) et (Q, κ_Q) deux problèmes paramétrés et une fonction calculable $\mathcal{F} : \Sigma^* \to \Sigma^*$. On dit que $\mathcal{F}$ est une transformation paramétrée polynômiale (PTP) de P vers Q, et on note $(P, \kappa_P) \leq_{PTP} (Q, \kappa_Q)$ si et seulement si, pour tout $x \in \Sigma^*$ on a, en posant $x' := \mathcal{F}(x)$:*
- *x' se calcule en temps polynômial.*
- *$x \in P \Leftrightarrow x' \in Q$.*
- *$\kappa_Q(x') \leq p(\kappa_P(x))$, pour un polynôme $p : \mathbb{N} \to \mathbb{N}$.*

Le lien avec les noyaux est donné par le théorème suivant :

Théorème 5. *[6] Soit (P, κ_P) et (Q, κ_Q) deux problèmes paramétrés, tels que $\tilde{P}$ est NP-difficile et $\tilde{Q}$ est dans NP. Supposons en outre $(P, \kappa_P) \leq_{PTP} (Q, \kappa_Q)$. Si (Q, κ_Q) a un noyau polynômial, alors (P, κ_P) a un noyau polynômial.*

Preuve. D'après les définition, on dispose des fonctions calculables suivantes :
- $\mathcal{F} : \Sigma^* \to \Sigma^*$: la transformation polynômiale paramétrée de (P, κ_P) vers (Q, κ_Q).
- $\mathcal{K} : \Sigma^* \to \Sigma^*$: la kernelization de (Q, κ_Q).
- $\mathcal{R} : \Sigma^* \to \Sigma^*$: la réduction polynômiale de $\tilde{Q}$ à $\tilde{P}$.

Alors la fonction calculable $\mathcal{A} := \mathcal{R} \circ \mathcal{K} \circ \mathcal{F}$ est un noyau polynômial pour (P, κ_P).

En effet, tout d'abord il est clair qu'étant donné que les fonctions utilisées sont toutes polynômiales en la taille de leur entrée, $\mathcal{A}$ l'est également. Ensuite, montrons que c'est bien un algorithme de kernelization : pour tout $x \in \Sigma^*$, on a :

$$\begin{aligned} x \in P &\Leftrightarrow \mathcal{F}(x) \in Q \\ &\Leftrightarrow \mathcal{K}(\mathcal{F}(x)) \in Q \\ &\Leftrightarrow \mathcal{R}(\mathcal{K}(\mathcal{F}(x)) \# 1^{\kappa_Q(\mathcal{K}(\mathcal{F}(x)))}) \in \tilde{P} \\ &\Leftrightarrow \mathcal{A}(x) \in P \end{aligned}$$

Concernant la taille de l'instance retournée, notons qu'étant donné que $\mathcal{K}$ est une kernelization polynômiale, sa sortie est polynômiale en la taille du paramètre de l'instance d'entrée. Cette dernière est polynômiale en $\kappa_P(x)$ car $\mathcal{F}$ est une transformation paramétrée polynômiale. Enfin, $\mathcal{R}$ étant polynômiale, l'instance retournée est de taille bornée par un polynôme en $\kappa_P(x)$, ce qui termine la démonstration. □

Un exemple de transformation paramétrée polynômiale est donnée plus bas par le théorème 16.

2.1.3 Cross-compositions

Méthode

La plupart des preuves de non existence hypothétique de noyau polynômial combine plusieurs techniques, principalement les OU-compositions et les transformations paramétrées polynômiales.

En effet, beaucoup utilisent d'abord une OU-composition d'un "problème gadget", c'est à dire un problème moins connu pouvant présenter un intérêt indépendant, ou bien d'une version modifiée du problème (problème de compression d'une solution, version colorée du problème, ou bien un cas particulier de celui-ci) et utilisent ensuite des transformations pour revenir au problème initial. Le lecteur pourra notamment se référer à [6, 18, 19, 22, 23, 27, 29, 36, 37]. La technique qui suit tente d'unifier cette manière de procéder en un résultat plus général.

En effet, un algorithme de cross-composition [2] aura le même principe qu'une OU-composition, en permettant cependant de partir d'une séquence d'instances d'un problème classique NP-complet étant "équivalentes" entre elles. Les définitions suivantes permettent de formaliser cette idée :

Définition 15. *Une relation d'équivalence $\mathcal{R}$ sur Σ^* est une relation d'équivalence polynômiale si :*

- *il existe une fonction calculable $f : \Sigma^* \times \Sigma^* \to \{0,1\}$ telle que pour $x, y \in \Sigma^*$:*
  - *$f(x,y) = 1 \Leftrightarrow x\mathcal{R}y$.*
  - *$f(x,y)$ se calcule en temps $O((|x|+|y|)^{O(1)})$.*
- *pour $S \subseteq \Sigma^*$, $\mathcal{R}$ partitionne les éléments de S en moins de $\max_{x\in S}|x|^{O(1)}$ classes.*

Définition 16. *Soit $L \subseteq \Sigma^*$ un problème, et (Q,κ) un problème paramétré. On dit que L se cross-compose en (Q,κ) s'il existe une relation d'équivalence polynômiale $\mathcal{R}$, et un algorithme prenant en entrée t éléments $(x_1,..,x_t) \in (\Sigma^*)^t$ appartenant à la même classe d'équivalence de $\mathcal{R}$, et retournant en temps $(\sum_{i=1}^{t}|x_i|)^{O(1)}$ une instance $x^* \in \Sigma^*$ telle que :*

- *$x^* \in Q \Leftrightarrow (x_1,..,x_t) \in OR(L)$.*
- *$\kappa(x^*) \leq (\max_{i=1..t}|x_i| + \log t)^{O(1)}$.*

Enfin, le théorème suivant permet d'utiliser les cross-compositions pour permettre de montrer des non existences de noyau polynômial :

Théorème 6. *[2] Soit $L \subseteq \Sigma^*$ un problème NP-complet qui se cross-compose en un problème paramétré (Q,κ) dont la version non paramétrée $\tilde{Q}$ est dans NP. Si (Q,κ) a un noyau polynômial, alors il existe un algorithme de OR-distillation de L vers lui-même.*

Preuve. On suppose que (Q,κ) admet un noyau polynômial. Montrons que l'on peut construire un algorithme de OR-distillation de L vers lui-même :

- Comme L est NP-complet, on a la réduction suivante : $\Phi : \Sigma^* \to \Sigma^*$ telle que $x \in \tilde{(Q)} \Leftrightarrow \Phi(x) \in \text{SAT}$.
- Comme $\tilde{Q}$ est NP-complet, on a la réduction suivante : $\Psi : \Sigma^* \to \Sigma^*$ telle que $x \in \text{SAT} \Leftrightarrow \Psi(x) \in L$
- Comme (Q,κ) admet un noyau polynômial, on a l'algorithme de kernelization $\mathcal{K} : \Sigma^* \to \Sigma^*$
- Comme L se cross-compose en (Q,κ), on a une relation d'équivalence $\mathcal{R}$ sur Σ^* et l'algorithme de cross-composition $\mathcal{C}$ de L vers $\tilde{Q}$ utilisant $\mathcal{R}$.

Soit $(x_1,..,x_t) \in (\Sigma^*)^t$. On pose $m = \max_{i=1..t}|x_i|$. On suppose $t \leq (|\Sigma|+1)^m$, c'est à dire $\log(t)$ est de l'ordre de $O(m)$. Dans le cas contraire, il y a alors des éléments de la séquence identiques, que l'on retire.

Etape (1) Grâce à la relation d'équivalence polynômiale $\mathcal{R}$, on partitionne en temps polynômial $\{x_1, .., x_t\}$. On obtient alors une séquence $(X_1, .., X_r)$ telle que $\forall i = 1..t$, $\forall x, y \in X_i$ $x\mathcal{R}y$. On a $r \leq m^{O(1)}$.

Etape (2) Grâce à l'algorithme de cross-composition C, on transforme polynômialement chaque X_i en un élément $y_i \in \Sigma^*$. Pour tout $i = 1..r$ on a $\kappa(y_i) \leq m^{O(1)}$.

Etape (3) Grâce à l'algorithme de kernelization $\mathcal{K}$, on transforme polynômialement chaque y_i en son noyau y_i^*. On a pour tout $i = 1..r$ $|y_i^*| \leq \kappa(y_i) \leq m^{O(1)}$.

Etape (4) Pour tout $i = 1..r$, on pose $\tilde{y}_i := y_i^* \# 1^{\kappa(y_i^*)}$.

Etape (5) On retourne enfin la chaîne $l^* := \Psi(\bigvee_{i=1}^{r} \Phi(\tilde{y}_i))$.

La figure 2.2 résume la construction réalisée.
L'algorithme ainsi construit est bien polynômial en $t * n$, et on a bien $l^* \leq m^{O(1)}$. Enfin :

$$\begin{aligned}
l^* \in L &\Leftrightarrow \Psi(\bigvee_{i=1}^{r} \Phi(\tilde{y}_i)) \in L \\
&\Leftrightarrow \bigvee_{i=1}^{r} \Phi(\tilde{y}_i) \in SAT \\
&\Leftrightarrow \exists i \in \{1, .., r\} : \Phi(\tilde{y}_i) \in SAT \\
&\Leftrightarrow \exists i \in \{1, .., r\} : \tilde{y}_i \in \tilde{Q} \\
&\Leftrightarrow \exists i \in \{1, .., r\} : y_i^* \in Q \\
&\Leftrightarrow \exists i \in \{1, .., r\} : y_i \in Q \\
&\Leftrightarrow \exists i \in \{1, .., r\} : \exists x \in X_i : x \in L \\
&\Leftrightarrow \exists i \in \{1, .., t\} : x_i \in L
\end{aligned}$$

D'où il existe un algorithme de OR-distillation de L vers L. □

Corollaire 3. *Sauf si $PH = \Sigma_p^3$, s'il existe un algorithme de cross-composition d'un problème NP-complet vers un problème paramétré (Q, κ) dont $\tilde{Q}$ est NP-complet, alors (Q, κ) n'a pas de noyau polynômial.*

Ainsi, afin de montrer qu'un problème paramétré n'admet probablement pas de noyau polynômial, il suffit de trouver un problème classique NP-complet qui se cross-compose en notre problème de départ. Un exemple de cross-composition est donné plus bas par le théorème 11

Comme annoncé précédemment, la technique de cross-composition tente d'unifier les deux principales techniques précédentes de OU-composition et de transformation paramétrée polynômiale.
En effet, nous pouvons tout d'abord remarquer qu'un algorithme de OU-composition est un cas particulier de cross-composition, comme le montre la proposition suivante :

Proposition 6. *Soit (P, κ) un problème paramétré et C un algorithme de OR-composition pour celui-ci. Alors il existe un algorithme de cross-composition de $\tilde{P}$ vers (P, κ).*

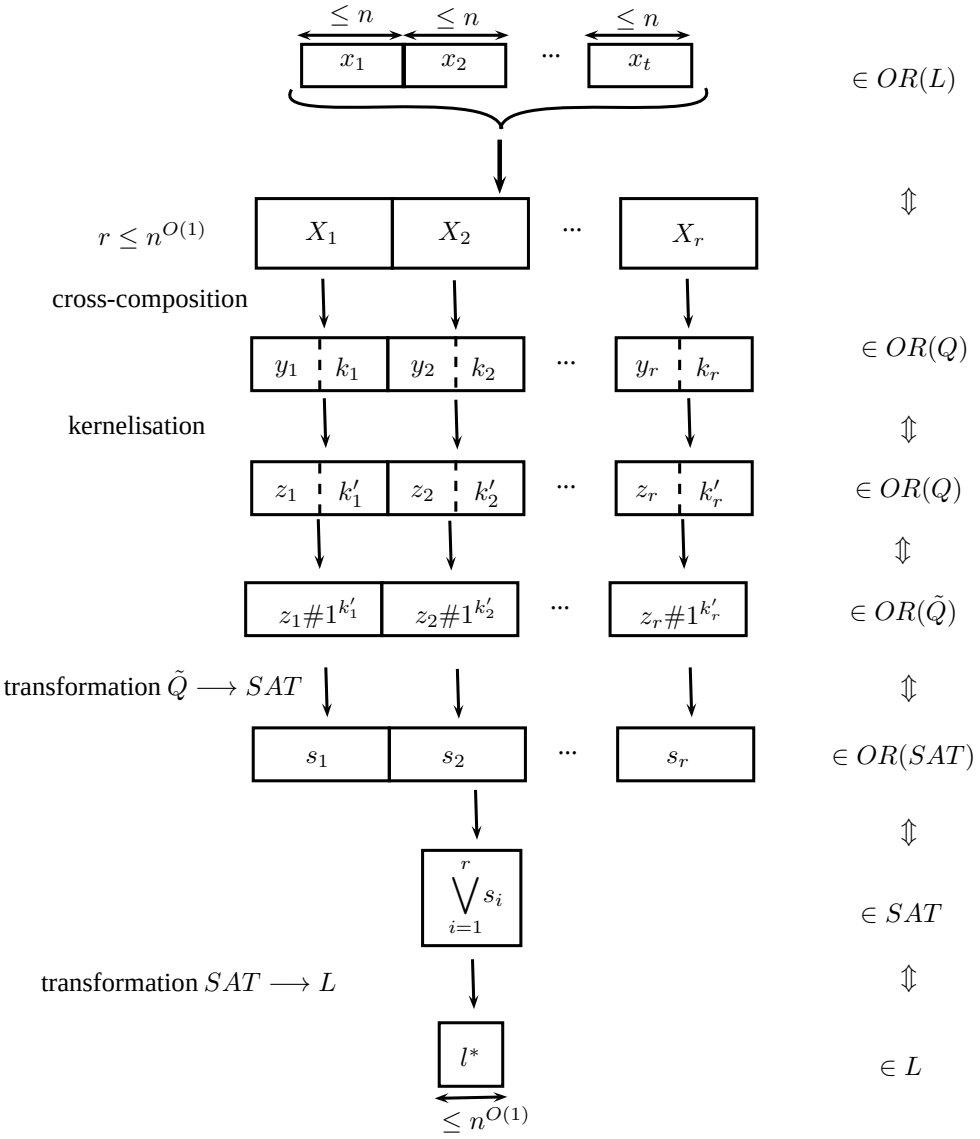


FIGURE 2.2 – Schéma de la preuve du théorème 6

Preuve. On définit la relation d'équivalence $\mathcal{R}$ de la manière suivante : deux chaînes z_1, z_2 sont équivalentes si elles ne sont pas de la forme $x\#1^k$, ou bien si $z_1 = x_1\#1^k$ et $z_2 = x_2\#1^k$ pour un certain $k \in \mathbb{N}$. C'est bien une relation d'équivalence polynômiale car elle partitionnera un ensemble $S = \{z_i := x_i\#1^{k_i} : i = 1..t\}$ en au pire $\max_{i=1..t} k_i \leq \max_{i=1..t} |z_i|$ classes (les chaînes ne codant pas des objets de la forme $x\#1^k$ seront dans une classe supplémentaire), d'où un nombre de classes de l'ordre de $O(\max_{i=1..t} |z_i|)$.

Soit $(z_1, .., z_t) \in (\Sigma^*)^t$ avec $z_i := x_i\#1^k$ pour un certain $k \in \mathbb{N}$. A l'aide de l'algorithme de OR-

composition C, on transforme la séquence d'instances en une instance (x', k') (si la séquence ne contient que des chaînes mal formées, on retourne une chaîne quelconque n'appartenant pas à Q).

- l'algorithme est bien polynômial en $t * \max_{i=1..t} |z_i|$.
- on a bien $(x', k') \in P \Leftrightarrow ((x_1, k_1), ..., (x_t, k_t)) \in OR(P) \Leftrightarrow (z_1, .., z_t) \in OR(\tilde{P})$.
- $k' \leq p(k)$ pour un polynôme p.

D'où il existe bien un algorithme de cross-composition de $\tilde{P}$ vers P. □

Enfin, la proposition suivante montre que la présence d'un algorithme de OU-composition pour un problème, suivit d'une transformation paramétrée polynômiale vers un autre problème (technique utilisée dans la majorité des cas afin de montrer la non-existence de noyau polynômial) revient également à construire une cross-composition :

Proposition 7. *Soit (P, κ_P) et (Q, κ_Q) deux problèmes paramétrés, $\mathcal{C}$ un algorithme de OR-composition pour (P, κ_P), et $\mathcal{T}$ une transformation paramétrée polynômiale de (P, κ_P) vers (Q, κ_Q), autrement dit $(P, \kappa_P) \leq_{PPT} (Q, \kappa_Q)$. Alors il existe un algorithme de cross-composition de $\tilde{P}$ vers (Q, κ_Q).*

Preuve. Soit $(z_1, .., z_t)$ avec $z_i = x_i \# 1^k$ pour tout $i = 1..t$. On définit la même relation d'équivalence $\mathcal{R}$ que précédemment, définie ainsi : deux chaînes z_1, z_2 sont équivalentes si elles ne sont pas de la forme $x\#1^k$, ou bien si $z_1 = x_1\#1^k$ et $z_2 = x_2\#1^k$ pour un certain $k \in \mathbb{N}$. Montrons que $(z_1, .., z_t) \mapsto \mathcal{T}(\mathcal{C}(z_1, .., z_t))$ est un algorithme de cross-composition :

- $\mathcal{R}$ est bien une relation d'équivalence polynômiale.
- $\mathcal{T}(\mathcal{C}(z_1, .., z_t))$ est calculable en temps $(\sum_{i=1}^{t} |x_i| + k)^{O(1)} \leq (\sum_{i=1}^{t} |z_i|)^{O(1)}$.
- $\mathcal{T}(\mathcal{C}(z_1, .., z_t)) \in Q \Leftrightarrow \mathcal{C}(z_1, .., z_t) \in P \Leftrightarrow (z_1, .., z_t) \in OR(\tilde{P})$.
- en posant $(x', k') := \mathcal{C}(z_1, .., z_t)$ et $(x^*, k^*) := \mathcal{T}(x', k')$ on a $k^* \leq k'^{O(1)} \leq k^{O(1)} \leq (\sum_{i=1..t} |z_i|)^{O(1)}$.

□

2.2 Autres résultats

Cette partie traite d'autres résultats en rapport avec les noyaux. Dans un premier temps, nous verrons les conséquences de l'existence de noyaux linéaires pour des problèmes duaux. Ensuite, nous présenterons une manière d'affiner la forme des noyaux que l'on ne peut pas obtenir, et enfin une méthode pour montrer l'existence de noyaux dits "forts".

2.2.1 Problème dual

Méthode

Cette partie traite des contraintes sur la taille d'un noyau lorsqu'un problème et son dual en admettent un de taille linéaire. Les résultats sont issus de [10]. Les définitions suivantes permettent de définir ce qu'est le problème dual d'un autre problème.

Définition 17. *Soit (Q, κ) un problème paramétré, et $s : \Sigma^* \to \mathbb{N}$ une fonction calculable. s est une fonction de taille pour (Q, κ) si et seulement si pour tout $x \in \Sigma^*$ on a :*

- $0 \leq \kappa(x) \leq s(x)$.
- $s(x) \leq |x|$.

– *s est indépendante de κ.*

Définition 18. *Soit (Q, κ) un problème paramétré et s une fonction de taille de Q. Le problème dual de (Q, κ) est le problème (Q, κ_d), avec pour tout $x \in \Sigma^*$, $\kappa_d(x) = s(x) - \kappa(x)$.*

Le principal résultat concernant les problèmes duaux est le suivant :

Théorème 7. *[10] Soit (P, κ) un problème paramétré tel que $\tilde{P}$ est NP-difficile, et $s : \Sigma^* \to \mathbb{N}$ une fonction de taille de (P, κ). Supposons que (P, κ) admette un noyau linéaire, avec α comme constante et que son dual (P, κ_d) admette également un noyau linéaire, avec α_d comme constante, et tels que $\alpha, \alpha_d \geq 1$.*
Si $(\alpha - 1)(\alpha_d - 1) < 1$, alors $P = NP$.

Preuve. Soit (Q, κ) un tel problème, et s sa fonction de taille. On note $r : \Sigma^* \to \Sigma^*$ son algorithme de kernelization, et $r_d : \Sigma^* \to \Sigma^*$ celui de son dual (Q, κ_d). Montrons que l'on peut décider Q en temps polynômial.
Soit $R : \Sigma^* \to \Sigma^*$ la fonction calculable définie pour tout $x \in \Sigma^*$ par

$$R(x) = \begin{cases} r(x) & \text{si } \kappa(x) \leq \frac{\alpha_d}{\alpha+\alpha_d} s(x) \\ r_d(x) & \text{sinon} \end{cases}$$

Notons $k := \kappa(x)$, $x' = R(x)$ et $k' := \kappa(x')$. On a alors :
– cas 1 : si $k \leq \frac{\alpha_d}{\alpha+\alpha_d}$, alors $s(x') \leq \alpha k \leq \frac{\alpha\alpha_d}{\alpha+\alpha_d} s(x)$
– cas 2 : si $k > \frac{\alpha_d}{\alpha+\alpha_d}$, alors

$$\begin{aligned} s(x') &\leq \alpha_d(s(x) - k) \\ &< \alpha_d(s(x) - \frac{\alpha_d}{\alpha + \alpha_d} s(x)) \\ &= \frac{\alpha\alpha_d}{\alpha + \alpha_d} s(x) \end{aligned}$$

D'où dans les deux cas $s(x') \leq \frac{\alpha\alpha_d}{\alpha+\alpha_d} s(x)$. Or $(\alpha - 1)(\alpha_d - 1) < 1 \Leftrightarrow \frac{\alpha\alpha_d}{\alpha+\alpha_d} < 1$.
s étant une fonction entière, on a $s(x') < s(x)$. Ainsi, en appliquant récursivement R au plus $s(x)$ fois, on peut décider si $x \in Q$ ou non. D'où on peut résoudre Q en un temps polynômial, et $P = NP$.

□

Exemple

La technique présentée ici peut sembler plus "forte" que les précédentes, du fait que les contraintes obtenues concernant la taille des noyaux soient valables sous l'hypothèse $P \neq NP$, et non plus $PH \neq \Sigma_3^p$. Cependant, pour pouvoir appliquer ces résultats, il faut pouvoir disposer de problèmes admettant d'une part un noyau polynômial, et d'autre part une version duale pertinente. Ainsi, les résultats utilisant cette technique ont été jusqu'ici moins nombreux que pour les précédentes, et à notre connaissance, seul [10] présente des applications sur des problèmes connus.
On a par exemple le résultat suivant, en supposant $P \neq NP$:

Proposition 8. *[10] Pour tout $\epsilon > 0$, il n'existe pas de noyau de taille $(\frac{4}{3} - \epsilon)k$ pour le problème* K-PLANAR VERTEX COVER.

Preuve. En effet, le théorème des 4 couleurs implique un noyau de taille $4k$ pour K-PLANAR INDEPENDENT SET, qui est le problème dual de K-PLANAR VERTEX COVER. □

2.2.2 OU linéaire

Méthode

Dans [12], Chen et al. raffinent les résultats de Bodlaender et al. [1] sur la OU-composition afin de proposer des bornes inférieures plus fortes pour une généralisation de la notion de noyau. Ils définissent pour cela les ϵ-réductions :

Définition 19. *Soit $\epsilon > 0$. Un problème paramétré (Q, κ) admet une ϵ-réduction s'il existe une fonction calculable $\mathcal{A} : \Sigma^* \to \Sigma^*$ telle que pour tout $x \in \Sigma^*$ on a*
- *$\mathcal{A}(x)$ est calculé en temps polynômial*
- *$|\mathcal{A}(x)| \leq \kappa(x)^{O(1)}.|x|^{1-\epsilon}$*

Remarquons que si (Q, κ) admet un noyau polynômial, alors il admet une ϵ-réduction pour tout $\epsilon \in]0, 1]$.

On a également la relaxation de la notion de OU-composition, représentée par la notion de OU linéaire :

Définition 20. *Soit (Q, κ) un problème paramétré. Un OU linéaire pour le problème (Q, κ) est un algorithme polynômial $\mathcal{O}$ qui, étant donné une séquence finie d'instances $\bar{x} = (x_1, ..., x_t) \in (\Sigma^*)^t$ retourne une instance $\mathcal{O}(\bar{x})$ telle que :*
- *$|\mathcal{O}(\bar{x})| \leq t.(\max_{i=1..t} |x_i|)^{O(1)}$*
- *$\kappa(\mathcal{O}(\bar{x})) \leq (\max_{i=1..t} |x_i|)^{O(1)}$*
- *$\mathcal{O}(\bar{x}) \in Q \Leftrightarrow \exists i \in \{1, .., t\} : x_i \in Q$*

Un OU linéaire est en fait presque identique à une OU-composition, hormis le fait que les instances en entrée n'ont pas forcément la même valeur de paramètre, et que l'instance retournée est de taille bornée par $t.(\max_{i=1..t} |x_i|)^{O(1)}$.
Le résultat est alors du même type que pour la OU-composition :

Théorème 8. *[12] (admis) Soit $\epsilon > 0$. Soit (Q, κ) un problème paramétré admettant un OU linéaire et tel que $\tilde{Q}$ est NP-complet. Sauf si $PH = \Sigma_3^p$, le problème (Q, κ) n'admet pas de ϵ-réduction*

Exemple

Les auteurs montrent notamment que le problème k-path, qui n'admet pas de noyau polynômial sous l'hypothèse $PH \neq \Sigma_3^p$ [1], n'admet pas non plus d'ϵ-réduction.

2.2.3 Non existence de noyaux forts

Méthode

La méthode qui suit, introduite par Chen, Flum et Müller [12] permet de montrer la non-existence de noyaux polynômiaux dits "forts", sous une hypothèse plus forte que dans les résultats précédents : $P \neq NP$:

Définition 21. *Soit (Q, κ) un problème paramétré. Une fonction $\mathcal{N} : \Sigma^* \to \Sigma^*$ est un noyau fort pour (Q, κ) si et seulement si :*
- *$\mathcal{N}$ est un noyau pour (Q, κ)*
- *pour tout $x \in \Sigma^*$ on a $\kappa(\mathcal{N}(x)) \leq \kappa(x)$*

On utilise pour cela le résultat suivant :

Théorème 9. *[12] Soit (Q, κ) un problème FPT dont la version non paramétrée est NP-complète, et soit $\mathcal{F} : \Sigma^* \to \Sigma^*$ un noyau polynômial pour (Q, κ) telle que pour tout $x \in \Sigma^*$ on a $\kappa(\mathcal{F}(x)) < \kappa(x)$. Alors $P = NP$.*

Preuve. Soit $\mathcal{F}$ un tel algorithme et $x \in \Sigma^*$. On définit l'algorithme $\mathcal{A}$ consistant à calculer itérativement $\mathcal{F}(x)$, $\mathcal{F}(\mathcal{F}(x))$ ... jusqu'à ce que le paramètre soit nul. Par définition de $\mathcal{F}$, au plus $\kappa(x)$ étapes sont nécessaires. Comme (Q, κ) est FPT, l'instance obtenue x' est décidable en temps polynômial en $|x'|$ (car $\kappa(x') = 0$). Comme $|\mathcal{F}(x)| \leq p(\kappa(x))$ pour un polynôme p et que $\kappa(\mathcal{F}(x)) < \kappa(x)$, on a $|x'| < \kappa(x)$, et comme κ est calculée en temps polynômial, on a $|x'| \leq |x|^{O(1)}$. Ainsi, $\mathcal{A}$ est un algorithme polynômial en $|x|$ permettant de décider si $x \in Q$. Comme la version non paramétrée de (Q, κ) est NP-complète, on a bien $P = NP$. □

Remarquons que la fonction $\mathcal{F}$ décrite dans le théorème précédent n'est pas un noyau fort : en effet, l'inégalité est stricte pour $\mathcal{F}$ alors qu'elle doit être large. Ce théorème joue en quelque sorte le même rôle que le théorème 2 pour les algorithmes de OU-composition : on va maintenant montrer que l'existence à la fois d'un noyau fort polynômial et d'un autre algorithme que l'on va définir permet de construire un algorithme comme celui utilisé dans le théorème précédent, ce qui sera impossible sous l'hypothèse $P \neq NP$.

Définition 22. *Soit (Q, κ) un problème paramétré. Une fonction calculable $\mathcal{R} : \Sigma^* \to \Sigma^*$ est une réduction polynômiale diminuant le paramètre si et seulement si pour tout $x \in \Sigma^*$:*
- *$\mathcal{R}(x)$ est calculé en temps polynômial*
- *$\kappa(\mathcal{R}(x)) < \kappa(x)$*

On a alors le résultat suivant :

Théorème 10. *[12] Soit (Q, κ) un problème paramétré FPT dont la version non paramétrée est NP-complète. Si (Q, κ) admet une réduction polynômiale diminuant le paramètre, alors (Q, κ) n'admet pas de noyau fort polynômial, sous l'hypothèse $P \neq NP$.*

Preuve. En effet, si (Q, κ) admettait une telle réduction $\mathcal{R}$ ainsi qu'un noyau fort polynômial $\mathcal{N}$, il est facile de voir que la composition $\mathcal{N} \circ \mathcal{R}$ est un algorithme comme celui defini dans le théorème 9. □

Exemple

On utilise le résultat précédent sur le problème POINTED PATH :

> POINTED PATH
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$, un sommet $v \in V$
> **question :** Existe-t-il dans G un chemin de taille k commençant par v ?
> **paramètre :** k

Proposition 9. *Le problème* POINTED PATH *n'admet pas de noyau fort polynômial s'il est paramétré par la taille du chemin à trouver.*

Preuve. On admet l'appartenance à FPT pour le problème et la NP-complétude de sa version non paramétrée. On construit une réduction polynômiale diminuant le paramètre : soit (G, k, v) une instance du problème, en supposant $k > 2$. Pour un chemin de longueur 2 $P = (v_1, v_2, v_3)$, on définit le graphe G_P à partir de G en supprimant les deux premiers sommets de P (v_1, etv_2). On construit le graphe G' constitué de l'union disjointe de tous les graphes G_P pour tous les

chemins de longueur 2 commençant par V. On ajoute à G' un sommet w relié à chaque G_P par le troisième sommet v_3 du chemin dont on avait enlevé les deux premiers sommets. Il est facile de vérifier que l'instance ainsi retournée est construite en temps polynômial, et G' a un chemin de taille $k-1$ commençant par w si et seulement si G a un chemin de taille k commençant par v. □

Chapitre 3

Applications

Cette partie traite de l'application des méthodes vues précédemment afin de montrer de nouvelles bornes inférieures, ou bien d'en généraliser d'autres.

3.1 Ensemble dominant

3.1.1 Introduction et travaux existants

Cette section traite du problème de l'ensemble dominant défini par :

> DOMINATING SET
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$.
> **question :** Existe-t-il $D \subseteq V$ avec $|D| \leq k$ et tel que pour tout $x \in V$, soit $x \in D$, soit x est adjacent à un sommet $y \in D$?

La figure 3.1 montre un exemple d'ensemble dominant de taille 3.

Lorsque paramétré par la taille de la solution, DOMINATING SET est connu pour être W[2]-difficile, ce qui motive l'étude de la complexité du problème muni de paramétrisations différentes. Les résultats de Courcelle [16] montrent d'ailleurs que lorsque paramétré par la treewidth du graphe d'entrée, le problème est FPT. Cependant, dans [1], les auteurs montrent que le problème DOMINATING SET n'admet pas de noyau polynômial pour cette paramétrisation. Il est alors intéressant d'étudier l'existence d'un noyau polynômial pour d'autres paramétrisations. Dom et al. [19] "ferment" la hiérarchie énoncée partie 1.3.3 en montrant que ce même problème n'admet pas de noyau polynômial s'il est paramétré par le couple (taille d'un vertex cover, taille de la solution), ce qui implique en particulier qu'il ne peut pas en admettre un s'il est uniquement paramétré par la taille d'un vertex cover.

Leur démonstration utilise une OU-composition du problème COLORED SMALL UNIVERSE HITTING SET, puis une transformation paramétrée polynômiale vers DOMINATING SET. Nous proposons dans la partie suivante une démonstration plus simple utilisant une cross-composition, puis nous montrons la non existence de noyau polynômial pour ce même problème, cette fois paramétré par la distance à une clique, avec les conséquences que cela implique.

3.1.2 Résultats

Dans la suite, pour tout $n \in \mathbb{N}$, $[n]$ désigne l'ensemble $\{1, 2, ..., n\}$.

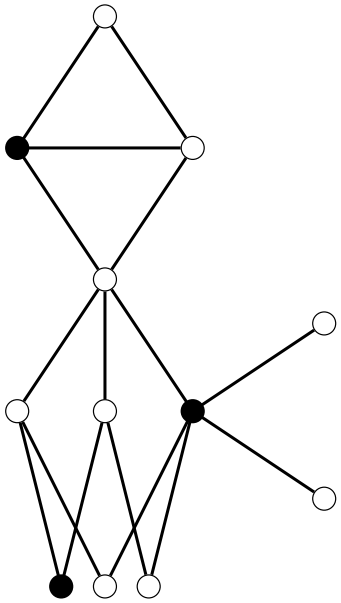


FIGURE 3.1 – Exemple d'ensemble dominant de taille 3. Les nœuds de l'ensemble dominant sont en noirs, les autres en blancs.

Paramétrisation par vertex cover

Théorème 11. DOMINATING SET *paramétré par la taille d'un vertex cover du graphe d'entrée n'admet pas de noyau polynômial, sauf si* $PH = \Sigma_p^3$

Preuve. Montrons que VERTEX COVER se cross-compose en DOMINATING SET paramétré par vertex cover. Comme le problème VERTEX COVER est NP-complet, et que la version non paramétrée de DOMINATING SET paramétré par vertex cover est également NP-complète (réduction à partir de DOMINATING SET en générant polynômialement un vertex cover trivial), cela suffit, d'après [2], à obtenir le résultat.
Soit $(G_1, l_1), ..., (G_t, l_t)$ une séquence d'instances de VERTEX COVER, avec G_i un graphe, $l_i \in \mathbb{N}$ et où l'on demande s'il existe un vertex cover de taille l_i dans le graphe G_i, pour tout $i \in [t]$. Nous définissons notre relation d'équivalence $\mathcal{R}$ telle que toutes les chaînes n'encodant pas une instance valide sont équivalentes, et $(G_1, l_1), (G_2, l_2)$ sont équivalentes si et seulement si $|V(G_1)| = |V(G_2)|$ et $l_1 = l_2$. On pose ainsi par la suite $V(G_i) = [n]$ et $l_i = l$ pour tout $i \in [t]$. Nous allons construire une instance de DOMINATING SET paramétré par vertex cover (G', l', Z') où G' est un graphe, $l' \in \mathbb{N}$ et $Z' \subseteq V(G')$ est un vertex cover de G', telle que G' admette un ensemble dominant de taille l' si et seulement s'il existe $i \in [t]$ tel qu'il existe un vertex cover de taille l dans G_i. Les étapes suivantes permettent de construire une telle instance :

i. On créé un ensemble de sommets indépendants $I := \{u_1, ..., u_t\}$ représentant le sélectionneur d'instances, chacun relié à trois sommets α, β, γ. En outre, on relie γ à un ensemble indépendant S de taille $l + 3$.

ii. Pour tout couple $(p, q) \in [n] \times [n]$ avec $p \neq q$, on créé un sommet $w_{p,q}$. Cet ensemble de sommets est noté B. On relie un sommet $w_{p,q}$ à un sommet u_k si et seulement si $\{p, q\} \notin E(G_k)$.

iii. Pour tout $j \in [l]$, on créé une clique sur n sommets $A_j := \{v_1^j, ..., v_n^j\}$. Chacun des sommets de A_j est ensuite relié à un ensemble indépendant S_j de taille $l+3$. Enfin, pour tout $j \in [l]$, on relie v_i^j à $w_{p,q}$ si et seulement si $i = p$ ou $i = q$. On note $A := A_1 \cup ... \cup A_l$

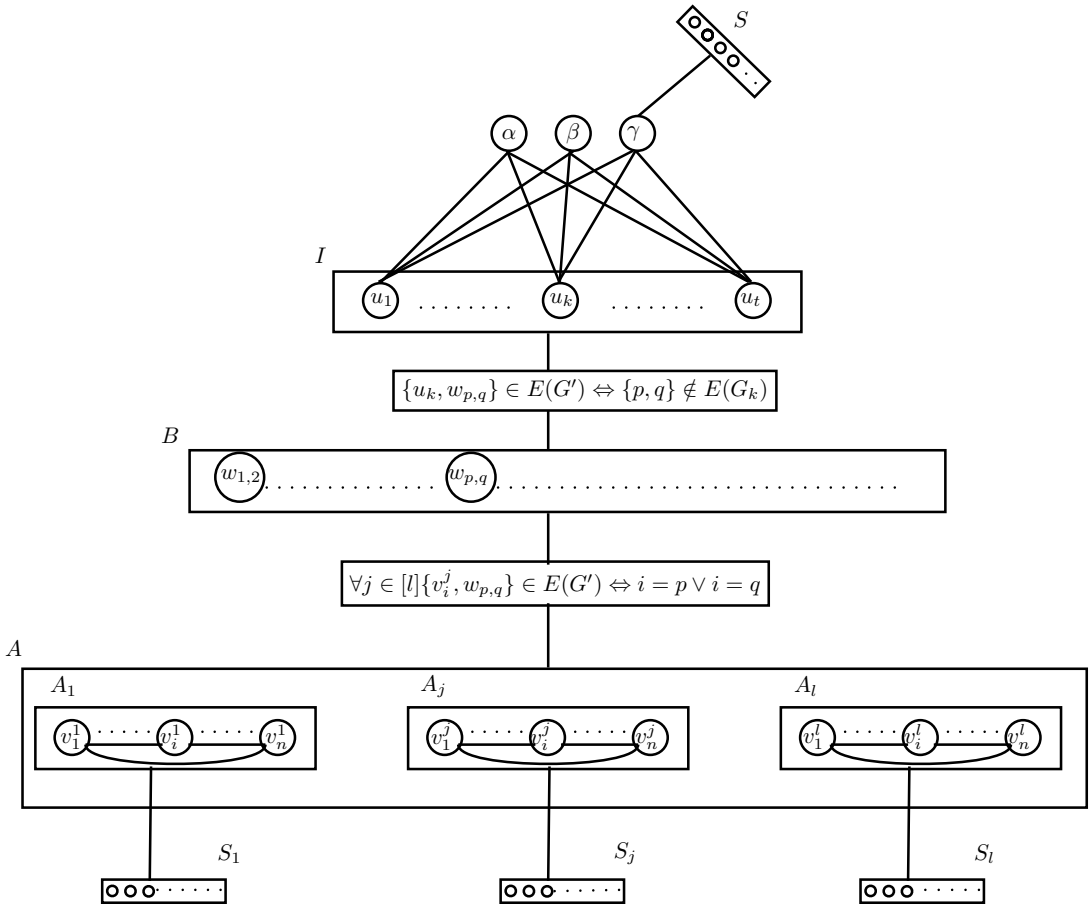


FIGURE 3.2 – Graphe obtenu pour la cross-composition de VERTEX COVER vers DOMINATING SET paramétré par vertex cover

La figure 3.2 résume la construction réalisée.
On définit $l' := l + 2$, et $Z' := S \cup \{\alpha, \beta, \gamma\} \cup B \cup A \cup S_1 \cup ... \cup S_l$ est un vertex cover de taille $(l+3) + 3 + \binom{n}{2} + ln + (l+3)l$, car I est un ensemble indépendant. Montrons que cette instance est positive si et seulement si il existe une instance positive dans la séquence reçue en entrée :

($\Leftarrow$) Supposons qu'il existe un vertex cover $C := \{c_1, ..., c_l\} \subseteq [n]$ de taille l pour le graphe G_k, et construisons un ensemble dominant D de taille l' dans G'. Posons $D := \{\gamma, u_k\} \cup \{v_{c_i}^i : i \in [l]\}$. On a alors :

- S est dominé par γ.
- le sélectionneur d'instances I est dominé par γ.

- α et β sont dominés par u_k.
- pour tout $j \in [l]$, A_j est dominé par $v^j_{c_j}$ (car A_j est une clique).
- pour tout $j \in [l]$, S_j est dominé par $v^j_{c_j}$.
- soit $(p, q) \in [n] \times [n]$.
  - Si $\{p, q\} \notin E(G_k)$, alors $w_{p,q}$ est dominé par u_k.
  - Si $\{p, q\} \in E(G_k)$, alors soit $p \in C$, soit $q \in C$. Sans perdre de généralité on suppose $p \in C$, c'est à dire il existe $j \in [l]$ tel que $p = c_j$. Mais alors $w_{p,q}$ est dominé par $v^j_{c_j}$.

D'où D est bien un ensemble dominant de taille $l + 2$ de G'.

($\Rightarrow$) Supposons maintenant qu'il existe un ensemble dominant D de taille $l + 2$ dans G', et montrons que l'on peut alors construire un vertex cover de taille l pour un certain graphe G_{i^*}. Remarquons d'abord que pour que chaque S_j soit dominé, il est nécessaire que D contienne au moins un sommet parmi chaque A_j, autrement dit, au moins l sommets parmi A. De plus, pour que S soit dominé, il est nécessaire que D contienne γ, et pour que α et β soient dominés, il est nécessaire que D contienne au moins un sommet parmi le sélectionneur d'instances I, disons u_{i^*}. Comme $|D| = l + 2$, on a forcément, pour tout $j \in [l]$, un sommet de A_j, disons $v^j_{c_j}$ qui appartient à D. Montrons alors que $C := \{c_1, ..., c_l\}$ est un vertex cover de G_{i^*}.
Soit $\{p, q\} \in E(G_{i^*})$. Comme $w_{p,q}$ n'est pas dominé par u_{i^*}, il est forcément dominé par au moins un sommet de A, disons par un sommet de A_j, c'est à dire soit par v^j_p, soit par v^j_q. Sans perdre de généralité, on suppose que v^j_p appartient à D. Mais alors $p \in C$, d'où toutes les arêtes sont couvertes par C.

Enfin, afin de compléter la définition d'une cross-composition, on a clairement que la construction proposée ci-dessus se réalise en temps polynômial, et que le paramètre généré en sortie (le vertex cover Z'), est de taille polynômiale en n (remarquons que $l < n$, car sinon les instances sont triviales).

□

Paramétrisation par distance à clique

Théorème 12. DOMINATING SET *paramétré par Clique+kv n'admet pas de noyau polynômial, sauf si* $PH = \Sigma^3_p$.

Preuve. La preuve consiste en une cross-composition presque identique à la précédente. Étant donnée une séquence $(G_1, l), ..., (G_t, l)$ d'instances de VERTEX COVER, avec $V(G_i) = [n]$, on construit en temps polynômial une instance (G', l', Z') où Z' est un sous-ensemble de $V(G')$ tel que $|Z'| \leq (n + \log t)^{O(1)}$ et $G \backslash Z'$ est une clique, et où l'on cherche dans G' un ensemble dominant de taille l'.
Les étapes suivantes permettent de construire une telle instance :

i. On créé une clique $I := \{u_1, ..., u_t\}$ représentant le sélectionneur d'instances. On créé en outre un ensemble indépendant S de taille $l + 2$, et on relie chaque u_i à tous les sommets de S.

ii. Pour tout couple $(p, q) \in [n] \times [n]$ avec $p \neq q$, on créé un sommet $w_{p,q}$. Cet ensemble de sommets est noté B. On relie un sommet $w_{p,q}$ à un sommet u_k si et seulement si $\{p, q\} \notin E(G_k)$.

iii. Pour tout $j \in [l]$, on créé une clique sur n sommets $A_j := \{v^j_1, ..., v^j_n\}$. Chacun des sommets de A_j est ensuite relié à un ensemble indépendant S_j de taille $l + 2$. Enfin, pour tout $j \in [l]$, on relie v^j_i à $w_{p,q}$ si et seulement si $i = p$ ou $i = q$. On note $A := A_1 \cup ... \cup A_l$

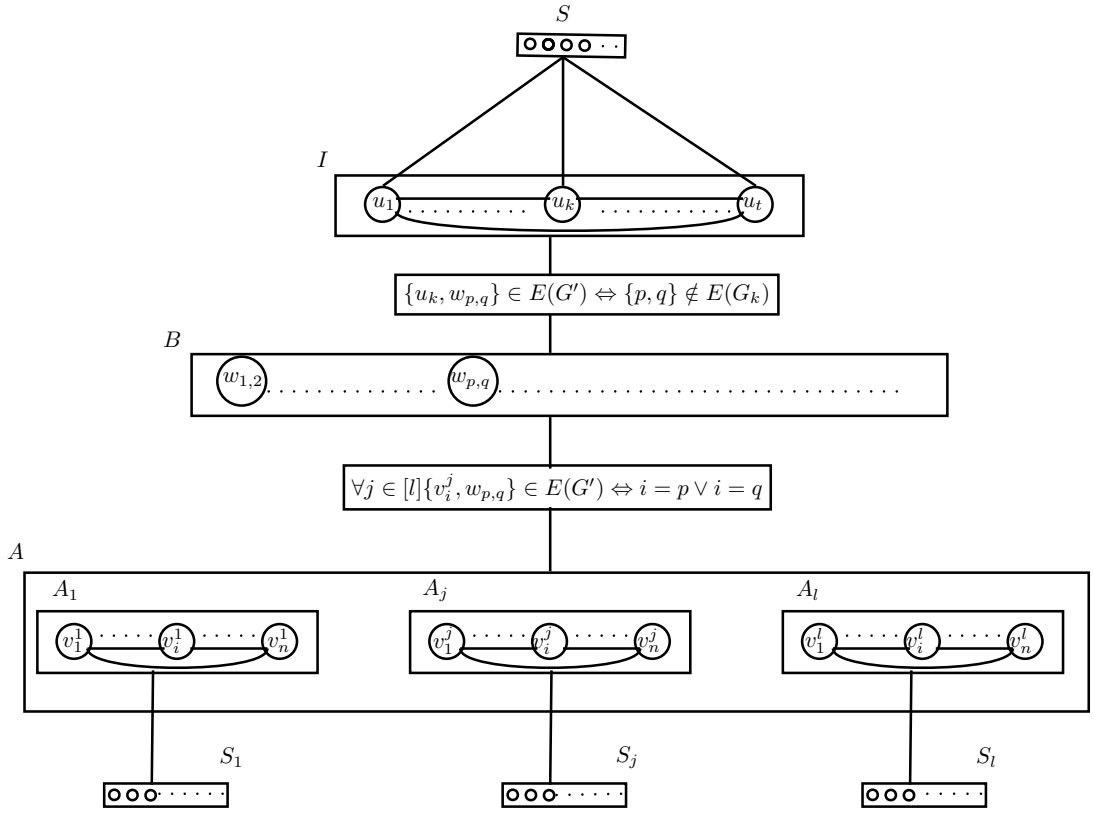


FIGURE 3.3 – Graphe obtenu pour la cross-composition de VERTEX COVER vers DOMINATING SET paramétré par le nombre de sommets à supprimer pour transformer le graphe d'entrée en une clique

La figure 3.3 résume la construction réalisée.
On définit $l' := l+1$, et $Z' := S \cup B \cup A \cup S_1 \cup ... \cup S_l$ est un ensemble de taille $(l+2)+\binom{n}{2}+ln+(l+2)l$ tel que $G' \backslash Z' = I$ est une clique. Montrons que cette instance est positive si et seulement si il existe une instance positive dans la séquence reçue en entrée :

($\Leftarrow$) Supposons qu'il existe un vertex cover $C := \{c_1, ..., c_l\} \subseteq [n]$ de taille l pour le graphe G_k, et construisons un ensemble dominant D de taille l' dans G'. Posons $D := \{u_k\} \cup \{v_{c_i}^i : i \in [l]\}$. On a alors :

- S est dominé par u_k.
- les autres u_i sont dominés par u_k (car I est une clique).
- pour tout $j \in [l]$, A_j est dominé par $v_{c_j}^j$ (car A_j est une clique).
- pour tout $j \in [l]$, S_j est dominé par $v_{c_j}^j$.
- soit $(p,q) \in [n] \times [n]$.
  - Si $\{p,q\} \notin E(G_k)$, alors $w_{p,q}$ est dominé par u_k.
  - Si $\{p,q\} \in E(G_k)$, alors soit $p \in C$, soit $q \in C$. Sans perdre de généralité on suppose $p \in C$, c'est à dire il existe $j \in [l]$ tel que $p = c_j$. Mais alors $w_{p,q}$ est dominé par $v_{c_j}^j$.

D'où D est bien un ensemble dominant de taille $l+1$ de G'.

($\Rightarrow$) Supposons maintenant qu'il existe un ensemble dominant D de taille $l+1$ dans G', et montrons que l'on peut alors construire un vertex cover de taille l pour un certain graphe G_{i^*}. Remarquons d'abord que pour que chaque S_j soit dominé, il est nécessaire que D contienne au moins un sommet parmi chaque A_j, autrement dit, au moins l sommets parmi A. De plus, pour que S soit dominé, il est nécessaire que D contienne au moins un sommet parmi le sélectionneur d'instances I, disons u_{i^*}. Comme $|D| = l+1$, on a forcément, pour tout $j \in [l]$, un sommet de A_j, disons $v^j_{c_j}$ qui appartient à D. Montrons alors que $C := \{c_1, ..., c_l\}$ est un vertex cover de G_{i^*}.
Soit $\{p, q\} \in E(G_{i^*})$. Comme $w_{p,q}$ n'est pas dominé par u_{i^*}, il est forcément dominé par au moins un sommet de A, disons par un sommet de A_j, c'est à dire soit par v^j_p, soit par v^j_q. Sans perdre de généralité, on suppose que v^j_p appartient à D. Mais alors $p \in C$, d'où toutes les arêtes sont couvertes par C.

Enfin, afin de compléter la définition d'une cross-composition, on a clairement que la construction proposée ci-dessus se réalise en temps polynômial, et que le paramètre généré en sortie (l'ensemble Z'), est de taille polynômiale en n (remarquons que $l < n$, car sinon les instances sont triviales). □

Conséquence

Le dernier résultat a des conséquences intéressantes :

Corollaire 4. *Soit $\mathcal{F}$ une classe de graphes contenant toutes les cliques. Le problème* DOMINATING SET *paramétré par le nombre de sommets à supprimer pour que le graphe d'entrée soit dans $\mathcal{F}$ n'admet pas de noyau polynômial.*

Preuve. Le nombre minimum de sommets à supprimer pour transformer un graphe G en un élément de $\mathcal{F}$ est au plus le nombre de sommets à supprimer pour transformer G en une clique. Donc le paramètre "nombre de sommets à supprimer pour que le graphe d'entrée soit dans $\mathcal{F}$" est inférieur au paramètre "nombre de sommets à supprimer pour que le graphe d'entrée soit une clique". Comme le problème DOMINATING SET n'admet pas de noyau polynômial lorsqu'il est paramétré par ce second paramètre, le résultat suit. □

Corollaire 5. *Le problème* DOMINATING SET *n'admet pas de noyau polynômial lorsqu'il est paramétré par la clique-width du graphe d'entrée.*

Preuve. Montrons que si un graphe est à distance k d'une clique (i.e. la suppression de k sommets donne une clique), alors la cliquewidth de ce graphe est au plus $k+2$. D'après la proposition 1, cela suffit à montrer le résultat.

Soit $G = (V, E)$ un graphe, et $X \subseteq V$ tel que $G[V \backslash X]$ est la clique K_s pour $s \leq n$. On note $k := |X|$. La procédure suivante permet de constuire le graphe G à l'aide des opérations autorisées pour calculer la cliquewidth : on commence par ajouter tous les sommets de X coloriés de 1 à k. Ensuite, pour chaque sommet de la clique, on le créé en le coloriant $k+1$ et on le relie à tous ses voisins dans G (chacun de ses voisins a une couleur différente, et c'est le seul sommet de couleur $k+1$) ainsi qu'aux sommets de couleur $k+2$ (les autres sommets de la clique). On change enfin sa couleur en $k+2$. On utilise bien $k+2$ couleurs.

□

3.2 Suppression de nœuds pour obtenir une propriété Π

3.2.1 Introduction et travaux existants

Dans [1], Bodlaender et al. montrent entre autres que le problème VERTEX COVER paramétré par la treewidth du graphe en entrée n'admet pas de noyau polynômial, et font remarquer que leur technique de démonstration peut s'appliquer à beaucoup d'autres problèmes. Le résultat qui suit généralise leur preuve en exhibant notamment une classe de problèmes pour laquelle leur idée fonctionne.

Dans ce qui suit, Π désigne une propriété de graphes *non triviale*, *héréditaire sur les sous-graphes induits* et *stable par union disjointe*. *Non triviale* signifie qu'il existe une infinité de graphes satisfaisant la propriété Π, ainsi qu'une infinité de graphes ne satisfaisant pas la propriété Π. *Héréditaire sur les sous-graphes induits* signifie qu'un graphe G satisfait la propriété Π si et seulement si chacun de ses sous-graphes induits satisfait la propriété Π. *Stable par union disjointe* signifie que l'union disjointe de deux graphes satisfaisant la propriété Π satisfait également la propriété Π. Un exemple d'une telle propriété pour un graphe est "ne comporte pas de cycle".

On rappelle que pour un graphe $G = (V, E)$, si $\mathcal{T}$ est une décomposition arborescente de G, $w(\mathcal{T})$ est la largeur de $\mathcal{T}$.

On définit les trois problèmes paramétrés suivants :

> W-NODE DELETION FOR PROPERTY Π
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$, $\mathcal{T}$ une décomposition arborescente de G.
> **question :** Existe-t-il $X \subseteq V$ tel que $|X| \leq k$ et $G[V \backslash X]$ satisfait la propriété Π ?
> **paramètre :** $w(\mathcal{T})$
>
> W-SUBGRAPH FOR PROPERTY Π
> **entrée :** Un graphe $G = (V, E)$, $k \leq |V|$, $\mathcal{T}$ une décomposition arborescente de G.
> **question :** Existe-t-il $X \subseteq V$ tel que $|X| > k$ et $G[X]$ satisfait la propriété Π ?
> **paramètre :** $w(\mathcal{T})$
>
> W-SUBGRAPH FOR PROPERTY Π COMPRESSION
> **entrée :** Un graphe $G = (V, E)$, $X \subseteq V$ tel que $G[X]$ satisfait la propriété Π, $\mathcal{T}$ une décomposition arborescente de G.
> **question :** Existe-t-il $X' \subseteq V$ tel que $|X'| > |X|$ et $G[X']$ satisfait la propriété Π ?
> **paramètre :** $w(\mathcal{T})$

On notera par la suite respectivement ces trois problèmes W-ND-Π, W-S-Π et W-S-Π-COMPRESSION.
Par exemple, pour la propriété "ne comporte pas de cycle", le problème W-ND-Π correspond au problème FEEDBACK VERTEX SET paramétré la treewidth du graphe d'entrée (on cherche un ensemble de sommet dont la suppression laisse un graphe sans cycle).

Enfin, les version classique de ces trois problèmes, obtenues en supprimant le paramètre, seront notées respectivement ND-Π-WITH-TREEWIDTH, S-Π-WITH-TREEWIDTH et S-Π-COMPRESSION-WITH-TREEWIDTH.

3.2.2 Résultats

Le principal résultat est la non existence de noyau pour W-NODE DELETION FOR PROPERTY Π. On utilise tout d'abord les lemmes suivants :

Lemme 1. *Les problèmes* ND-Π-WITH-TREEWIDTH*,* S-Π-WITH-TREEWIDTH *et* S-Π-COMPRESSION-WITH-TREEWIDTH *sont NP-complets.*

Preuve. D'après [35], le problème NODE DELETION FOR PROPERTY Π (ND-Π en abrégé) est NP-complet car Π est non triviale et héréditaire. En calculant en temps polynômial une décomposition arborescente triviale d'un graphe G, on réduit une instance de ND-Π en une instance de ND-Π-WITH-TREEWIDTH, donc ce dernier est NP-complet.

Ensuite, remarquons que S-Π-WITH-TREEWIDTH est le problème dual de ND-Π-WITH-TREEWIDTH. Ainsi, une solution pour l'un est une solution pour l'autre et inversement. D'où S-Π-WITH-TREEWIDTH est NP-complet.

Enfin, supposons que S-Π-COMPRESSION-WITH-TREEWIDTH est polynômial, i.e. on dispose d'une fonction calculable en temps polynômial *compress* qui, étant donné un graphe $G = (V, E)$, un ensemble $X \subseteq V$ tel que $G[X]$ satisfait la propriété Π et une décomposition arborescente de G, retourne un ensemble X' tel que $|X'| > |X|$ et $G[X']$ satisfait la propriété Π s'il en existe un, et -1 sinon. On est alors capable de résoudre le problème S-Π-WITH-TREEWIDTH, comme le montre l'algorithme 1. Ce dernier étant NP-complet, on a bien que le problème S-Π-COMPRESSION-WITH-TREEWIDTH est NP-complet.

Algorithme 1 algorithme de compression itérative pour le problème S-Π-WITH-TREEWIDTH

ENTRÉES: $G = (V, E)$ un graphe, $k \in \mathbb{N}$, $\mathcal{T}$ une décomposition arborescente de G.
SORTIE: YES s'il existe un ensemble $X \subset V$ de taille k tel que $G[X]$ satisfait la propriété Π, NO sinon

```
1: V' ← ∅
2: X ← ∅
3: pour i ← 1 à |V| faire
4:    V' ← V' ∪ {v_i}
5:    X ← X ∪ {v_i}
6:    si |X| > k alors
7:       T' ← restriction de T à V'
8:       A ← compress(G[V'], X, T')
9:       si A = −1 alors
10:         retourner NO
11:      sinon
12:         X ← A
13:      finsi
14:   finsi
15: fin pour
16: retourner YES
```

□

Lemme 2. *Le problème* W-S-Π-COMPRESSION *est OU-composable.*

Preuve. Soit $((G_1, X_1, \mathcal{T}_1), ..., (G_s, X_s, \mathcal{T}_s))$ une séquence de s instances du problème. On construit l'instance $(G', X', \mathcal{T}')$ de la manière suivante :

– $G' = (V', E')$ est l'union disjointe des G_i.
– X' est l'union disjointe des X_i.
– $\mathcal{T}$ est construit en reliant chacun des $\mathcal{T}_i$ de manière quelconque.

Remarquons que $\mathcal{T}$ est bien une décomposition arborescente de G', et que X' est bien une solution pour G', car chaque X_i est une solution pour G_i, et que Π est héréditaire sur sous-graphes induits.

Supposons qu'il existe X'' de taille strictement supérieure à $|X'|$ qui satisfait la propriété Π. Comme Π est une propriété héréditaire sur les sous-graphes induits, on a pour tout $i \in \{1, .., s\}$ que $G_i[X'' \cap V_i]$ satisfait la propriété Π. Par un argument de comptage, il existe forcément $i \in \{1, .., s\}$ tel que $|X'' \cap V_i| > |X_i|$.
Inversement, si pour un certain $i \in \{1, .., s\}$ il existe X'_i tel que $|X'_i| > |X|$ et $G_i[X'_i]$ satisfait la propriété Π, comme Π est stable par union disjointe, $X'' := X'_i \cup (\bigcup_{k \neq i'} X_k)$ est un ensemble de taille strictement supérieure à $|X'|$ tel que $G'[X'']$ satisfait la propriété Π.
Remarquons enfin que l'instance $(G', X', \mathcal{T})$ est construite en temps polynômial.

□

Lemme 3. *Le problème* W-S-Π-COMPRESSION *n'admet pas de noyau polynômial, sauf si* $PH = \Sigma_3^p$.

Preuve. D'après [1], les lemmes 1 et 2 permettent d'obtenir le résultat. □

Lemme 4. *Il existe une transformation paramétrée polynômiale de* W-S-Π-COMPRESSION *vers* W-S-Π.

Preuve. Soit $(G, X, \mathcal{T})$. Il suffit de retourner l'instance $(G, |X| + 1, \mathcal{T})$. □

Lemme 5. *Le problème* W-S-Π *n'admet pas de noyau polynômial, sauf si* $PH = \Sigma_3^p$.

Preuve. D'après [6], les lemmes 1 et 4 permettent d'obtenir le résultat. □

Lemme 6. *Il existe une transformation polynômiale paramétrée du problème* W-S-Π *vers* W-ND-Π.

Preuve. Soit $(G, k, \mathcal{T})$. Il suffit de retourner $(G, |V| - k, \mathcal{T})$.
En effet, s'il existe $X \subseteq V$ avec $|X| \geq k$ et tel que $G[X]$ satisfait Π, alors $X' := V \backslash X$ est un ensemble tel que $|X'| \leq |V| - k$ et tel que $G[V \backslash X'] = G[X]$ satisfait la propriété Π. La même idée convient pour la réciproque.

□

Théorème 13. *Le problème* W-ND-Π *n'admet pas de noyau polynômial, sauf si* $PH = \Sigma_3^p$.

Preuve. D'après [6], les lemmes 1 et 6 permettent d'obtenir le résultat. □

3.3 Partition domatique

3.3.1 Introduction et travaux existants

Introduction Dans cette section, on s'intéresse au problème de la partition domatique d'un graphe :

Définition 23. *Soit $G = (V, E)$ un graphe et $k \in \mathbb{N}$. Une k-partition domatique de G est une fonction $p : V \to \{1, .., k\}$ telle que pour tout $i = 1..k$, $p^{-1}(i)$ est un ensemble dominant de G. Autrement dit, c'est une coloration de G où chaque couleur induit un ensemble dominant. Le nombre domatique d'un graphe est le plus grand entier k tel que ce graphe admette une k-partition domatique.*

On définit le problème suivant pour tout $k \in \mathbb{N}$:

> k-DOMATIC PARTITION
> **entrée :** Un graphe $G = (V, E)$.
> **question :** Existe-t-il une k-partition domatique de G ?

La figure 3.4 montre un exemple de 3-partition domatique : chaque nœud est bien dominé par sa propre couleur et les 2 autres.

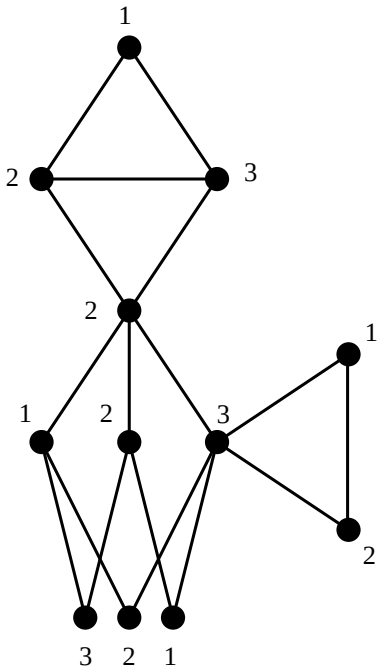


FIGURE 3.4 – Exemple de 3-partition domatique

Travaux existants Ce problème a été introduit pour ses applications dans les réseaux, notamment pour la recherche de groupes de transmission disjoints [14] ou des économies d'énergie [39]. Les cas $k = 1$ et $k = 2$ sont triviaux : en effet, tout graphe admet une partition en 2 ensembles dominants disjoints : il suffit pour cela de chercher un arbre couvrant tous les sommets du graphe, et de colorer alternativement les sommets en blanc et noir à partir de la racine, chaque sommet aura ainsi un voisin d'une couleur différente. Sa NP-complétude a été établie pour $k \geq 3$ par réduction à partir de SAT [26].

A partir de là, plusieurs techniques ont été utilisées pour tenter d'attaquer le problème. Une des manières est la conception d'algorithmes exponentiels de meilleure complexité que l'algorithme exponentiel trivial, en cherchant des algorithmes de complexité $O^*(c^n)$ pour une valeur

de c la plus petite possible. Koivisto et al. [33] ont notamment élaboré un algorithme de complexité $O^*(2^n)$. Concernant le problème de maximisation associé, où l'on cherche à trouver le plus grand entier k tel qu'un graphe G admette une k-partition domatique, un algorithme polynômial développé par Feige et al. [21] donne une solution approchée à facteur logarithmique.

Comme le problème est NP-complet pour $k \geq 3$, il est vain de chercher un algorithme FPT pour celui-ci lorsque paramétré par la taille de la partition. On s'intéresse alors à d'autres paramétrisations. Remarquons que pour k fixé et pour treewidth fixée, le problème peut être résolu en temps linéaire à l'aide des outils de Courcelle [16], car clairement exprimable à l'aide d'une formule MSO. Il est donc pertinent de considérer les paramétrisations de la hiérarchie $Treewidth \leq t+kv$ pour k fixé concernant la recherche de noyaux polynômiaux. A notre connaissance, la première utilisation de la complexité paramétrée pour notre problème est le résultat suivant :

Théorème 14. *[1] Le problème* 3-DOMATIC PARTITION *n'admet pas de noyau polynômial s'il est paramétré par la largeur d'une décomposition arborescente du graphe d'entrée, en supposant que les problèmes coNP-complets n'admettent pas d'algorithmes de OU-distillation.*

Preuve. L'union disjointe d'une séquence de graphes constitue un algorithme de AND-composition trivial. □

Nous proposons un résultat plus fort, dans le sens où il implique ce dernier, sous l'hypothèse cette fois-ci où $PH \neq \Sigma_p^3$, qui est une supposition considérée comme moins probable par la communauté que celle utilisée dans le théorème précédent. En outre, en utilisant les notations de la section 1.3.3, notre résultat implique la non existence de noyau polynômial lorsque paramétré par $Path+kv$ et donc en particulier $Forest+kv$ (taille d'un feedback vertex set).

Nous utilisons un récent résultat de Bodlaender et al. sur le problème de la 3-coloration :

Théorème 15. *[3, Corollaire 2] Le problème* 3-COLORATION *n'admet pas de noyau polynômial s'il est paramétré par Path+kv, dans l'hypothèse où $PH \neq \Sigma_p^3$.*

3.3.2 Résultats

Remarquons tout d'abord que les résultats négatifs concernant la non-existence de noyau pour 3-DOMATIC PARTITION sont valables pour tout $k \geq 3$ fixé, pour la plupart des paramétrisation structurelles. En effet, on a la proposition suivante :

Proposition 10. *Pour tout $k \in \mathbb{N}$, Il existe une transformation paramétrée polynômiale de* K-DOMATIC PARTITION *vers* (K+1)-DOMATIC PARTITION*, tous deux paramétrés par $\mathcal{C}+kv$ quelque soit la classe de graphes $\mathcal{C}$.*

Preuve. Soit $G = (V, E)$ un graphe et $X \subseteq V$ tel que $G[V \backslash X] \in \mathcal{C}$. On construit G' à partir de G en ajoutant un sommet universel ω (relié à tous les autres sommets). Si G admet une k-partition domatique, alors il suffit de donner à ω la couleur $k+1$, on a ainsi une $(k+1)$-partition domatique de G'. Inversement, si G' admet une $(k+1)$-partition domatique, alors G a une k-partition domatique partielle (i.e. ne couvrant pas tous les sommets), en enlevant la couleur des sommets colorés comme ω. Cette dernière peut enfin être étendu en une k-partition domatique de tout le graphe en remarquant qu'un ensemble dominant est stable par ajout de sommets. Concernant le paramètre, on retourne l'ensemble $X' := X \cup \{\omega\}$ de taille $|X| + 1$. On a alors $G'[V' \backslash X'] = G[V \backslash X] \in \mathcal{C}$ et la transformation est clairement réalisée en temps polynômial, ce qui termine la démonstration. □

Par soucis de simplicité, on donnera seulement les résultats négatifs pour $k = 3$, ceux-ci étant ainsi vrais quelque soit $k \geq 3$ d'après la proposition précédente, par récurrence.

Rappelons que la classe *LinearForest* est la classe des graphes étant l'union disjointe de chemins. On a alors le résultat suivant :

Théorème 16. *Le problème* 3-DOMATIC PARTITION *n'admet pas de noyau polynômial s'il est paramétré par LinearForest+kv.*

Preuve. La preuve qui suit s'inspire de celle montrant la NP-complétude de 3-DOMATIC PARTITION dans les graphes bipartis [32].

On construit une transformation paramétrée polynômiale de 3-COLORATION paramétré par Path+kv vers 3-DOMATIC PARTITION paramétré par *LinearForest*+kv.
Soit $G = (V, E)$ un graphe et $X \subseteq V$ un ensemble tel que $G[V \backslash X]$ est un chemin. On construit le graphe $G' = (V' \cup V'', E')$ et un modulateur X' de G' pour la classe des forêts linéaires de la manière suivante :

- $V' := V$.
- pour tout $e = \{u, v\} \in E$ on créé un sommet $\alpha_{u,v}$ dans V'' que l'on relie à u et v dans V'.
- on ajoute à V'' trois sommets w_1, w_2, w_3 reliés à tous les sommets de V'.
- $X' := X \cup \{w_1, w_2, w_3\}$.

La figure 3.5 résume la construction réalisée. Il est facile de vérifier que G' est construit en temps polynômial. Montrons maintenant que G' admet une 3-partition domatique si et seulement si G admet une 3-coloration.

$\Leftarrow$ Soit $c : V \to \{1, 2, 3\}$ une 3-coloration de V. Pour tout sommet $\alpha_{u,v}$ de V'', celui-ci n'est relié qu'à u et v qui sont adjacents dans G. Ainsi $c(u)$ et $c(v)$ sont différents, et on affecte à $\alpha_{u,v}$ la troisième couleur, il est ainsi dominé par les 3 couleurs. On affecte enfin la couleur i à w_i pour $i = 1, 2, 3$. Chaque sommet de V' est ainsi dominé par les trois couleurs grâce à w_1, w_2 et w_3. De même chaque w_i est dominé par les trois couleurs car c est une 3-coloration de V.

$\Rightarrow$ Réciproquement, montrons qu'une 3-partition domatique $p : V' \cup V'' \to \{1, 2, 3\}$ est une 3-coloration pour G (en prenant la restriction à V'). En effet, pour tous sommets u et v de V' tels que $\{u, v\}$ est une arête de G, le sommet $\alpha_{u,v}$ n'est relié qu'à u et v, donc pour être dominé par les 3 couleurs, u et v doivent avoir des couleurs différentes, ce qui termine la preuve de l'équivalence des solutions.

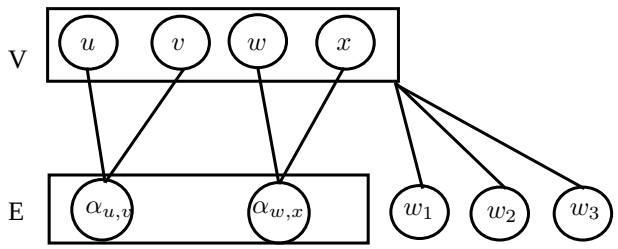


FIGURE 3.5 – Graphe G' obtenu après la transformation paramétrée polynômiale de 3-COLORATION paramétré par Path+kv vers 3-PARTITION DOMATIQUE paramétré par LinearForest+kv

Montrons maintenant que X' est bien un modulateur de G' pour la classe des forêts linéaires. Après avoir enlevé de G' les sommets de X', il ne reste dans V' que des sommets représentant un chemin $(v_1, v_2, ..., v_k)$ dans G, et il ne reste dans V'' que les sommets $\{\alpha_{v_1,v_2}, ..., \alpha_{v_{k-1},v_k}\}$, chaque $\alpha_{v_i,v_{i+1}}$ n'étant relié qu'à v_i et v_{i+1}, ainsi que d'autres sommets isolés. Les sommets restants forment ainsi un chemin et des sommets isolés. Enfin, on a $|X'| = |X| + 3$ qui reste bien polynômial en $|X|$, ce qui termine la démonstration.

□

Etant donné qu'une forêt de chemins est une forêt d'arbres particulière, on a le corollaire immédiat suivant :

Corollaire 6. *Le problème* 3-DOMATIC PARTITION *n'admet pas de noyau polynômial si paramétré par la taille d'un feedback vertex set.*

Ce dernier résultat nous amène ainsi à étudier la taille des noyaux lorsque l'on paramètre par la taille d'un vertex cover :

Problème ouvert 3. *Le problème* 3-DOMATIC PARTITION *admet-il un noyau polynômial s'il est paramétré par la taille d'un vertex cover ?*

Conclusion

Dans ce mémoire, nous nous sommes intéressés à l'analyse théorique des algorithmes de kernelization via la théorie de la complexité paramétrée. Cette dernière nous permet de garantir une performance des méthodes que l'on peut mettre en œuvre, et en particulier d'établir des bornes inférieures sur la taille des noyaux que l'on peut espérer obtenir. Ces bornes inférieures sont d'une grande importance, car le principal challenge en kernelization est de trouver des noyaux de taille la plus petite possible. On a vu en particulier que l'existence d'un algorithme de OU-composition ou de cross-composition et celle d'un noyau polynômial pour un problème et une paramétrisation donnés impliqueraient l'existence d'un algorithme de OU-distillation pour tous les problèmes NP-complets, qui à son tour impliquerait l'effondrement de la hiérarchie polynômiale au troisième niveau, chose considérée comme improbable par la communauté. Un important problème ouvert dans ce domaine concerne les algorithmes de ET-distillation : l'existence d'une ET-distillation pour un problème NP-complet implique-t-elle également une contradiction dans la théorie de la complexité ? Si oui, alors les algorithmes de ET-composition pourraient également être utilisés pour montrer la non-existence de noyaux polynomiaux pour des problèmes paramétrés. Nous avons également vu comment un problème pouvait transmettre sa non-existence de noyau polynômial grâce aux transformations paramétrées polynômiales.

Une fois ces outils définis, il est intéressant d'observer pour quels problèmes on peut espérer trouver ou non un noyau polynômial, et surtout pour quelles paramétrisations. La notion de hiérarchie de paramètres permet notamment de regarder les limites des algorithmes de kernelization pour un problème donné. On s'est intéressé en particulier à la hiérarchie de paramètres "distance à treewidth bornée", pour des bornes croissantes. Ces paramètres sont intéressants car ils rentrent dans le cadre de paramétrisation par distance à trivialité : si un problème est facile dans les graphes à treewidth bornée (ce qui est le cas de nombreux problèmes), on considère comme paramètre le nombre de sommets à enlever pour qu'il soit de treewidth bornée.
On observe par exemple que lorsque la borne est 0 (paramétrisation par la taille d'un vertex cover), le problème 3-COLORATION admet un noyau polynomial, mais n'en admet pas pour des bornes supérieures. Concernant le fameux problème du VERTEX COVER, beaucoup étudié en complexité paramétrée, un noyau polynomial est connu pour des valeurs de treewidth 0 (taille de la solution) et 1 (paramétrisation par taille d'un feedback vertex set), et le problème est ouvert pour les autres valeurs, sachant qu'il n'en existe pas lorsque paramétré par la treewidth du graphe d'entrée, paramètre situé à la fin de cette hiérarchie. Pour ce dernier paramètre, nous avons montré qu'une grande famille de problèmes n'admettait pas de noyau polynômial : les problèmes NODE DELETION FOR PROPERTY Π pour toute propriété de graphes Π héréditaire, non triviale et stable par union disjointe. Enfin, nous nous sommes intéressés au problème K-DOMATIC PARTITION, peu étudié en complexité paramétrée, en établissant la non existence de noyau polynômial pour le paramètre précédent "distance à treewidth bornée", pour toutes les valeurs différentes de 0. Le problème reste donc ouvert pour la paramétrisation par la taille d'un vertex cover, et

une réponse positive consituerait une séparation intéressante concernant la difficulté d'établir des algorithmes de kernelization efficaces pour ce problème.
Il serait également intéressant pour le futur d'étudier d'autres hiérarchies de paramètres, telles que la "distance à cliquewidth bornée", ou bien des distances à base de suppressions ou de modifications d'arêtes à la place de la suppression de sommets classique.

Annexe A

Compendium de bornes inférieures

Les problèmes marqués ★ sont des nouveaux résultats, leur preuve se trouve donc précédemment.
Sauf si $PH = \Sigma_p^3$, les problèmes suivants n'admettent pas de noyau polynômial.

3-COLORING [3]
input : a graph $G = (V, E)$, a set $X \subseteq V$ such that $G[V \setminus X]$ is a path.
question : Is G 3-colorable ?
parameter : $|X|$

Method : Polynomial parameter tranformation from CNF-SAT parameterized by the number of variables.

3-COLORING [3]
input : a graph $G = (V, E)$, $X \subseteq V$ a dominating set.
question : Is G 3-colorable ?
parameter : $|X|$

Method : Polynomial parameter tranformation from CNF-SAT parameterized by the number of variables.

3-SAT [41]
input : a 3-CNF formula F, a Horn-backdoor or 2CNF-backdoor $\mathcal{B}$ of F
question : is F satisfiable ?
parameter : size of $\mathcal{B}$

Method : OR-composition.

$\{0,1\}$**-CSP** [41]
input : a constraint network $I = (V, U, C)$ with binary domains, a tree decomposition $\mathcal{T}$ of the constraint graph of I
question : Is I satisfiable ?
parameter : width of $\mathcal{T}$

Method : OR-composition

BIPARTITE REGULAR PERFECT CODE [19]
input : a bipartite graph $G = (T \cup N, E)$ with every vertex in N having the same degree, $k \in \mathbb{N}$.
question : Is there a vertex subset $N' \subseteq N$ of size at most k such that every vertex in T has exactly one neighbor in N' ?
parameter : $k + |T|$

Method : OR-composition for the colored version of the problem, then polynomial parameter transformation (variant of RED-BLUE DOMINATING SET).

BOUNDED TREEWIDTH SUBGRAPH TEST [1]
input : a graph $G = (V, E)$, a graph H of treewidth at most k.
question : is H a subgraph of G ?
parameter : k

Method : OR-composition.

CAPACITED VERTEX COVER [19]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $cap : V \to \mathbb{N}$
question : does G contains a vertex cover $C \subseteq V$ of size at most k and a mapping from E to C such that for every $v \in C$, at most $cap(v)$ edges are mapped by v ?
parameter : k

Method : polynomial parameter transformation from RED-BLUE DOMINATING SET

CHROMATIC NUMBER [2]
input : a graph $G = (V, E)$, $l \in \mathbb{N}$, C a vertex cover of G
question : is there a proper l-coloration of V ?
parameter : $|C|$

Method : cross-composition from 3-COLORATION WITH TRIANGLE SPLIT DECOMPOSITION

C_l-FREE EDGE DELETION PROBLEM FOR $l \geq 12$ [27]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$
question : is there a subset $V' \subseteq V$ of size at most k such that $G \backslash V'$ is C_l-free ?
parameter : k.

Method : polynomial parameter transformation from the restriction of TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE WITHOUT 0-EDGES to ANNOTATED C_l-FREE EDGE-DELETION, then polynomial parameter transformation to C_l-FREE EDGE DELETION PROBLEM.

CLIQUE [1]
input : a graph $G = (V, E)$, $l \in \mathbb{N}$, $\mathcal{T}$ a tree decomposition of G
question : does G contains a clique of size l ?
parameter : width of $\mathcal{T}$

Method : OR-composition of the refinement version of the problem, then polynomial parameter transformation.

CLIQUE [2]
input : a graph $G = (V, E)$, $l \in \mathbb{N}$, C a vertex cover of G
question : does G contains a clique of size l ?
parameter : $|C|$

Method : cross-composition from the classical problem CLIQUE.

COLORFUL GRAPH MOTIF [18]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, a function $f : V \to \{1, ..., k\}$
question : Does there exist a connected set $S \subseteq V$ of size at most k such that $f|_S$ is bijective ?
parameter : k

Method : OR-composition (in forests of maximum degree 3).

CONNECTED DOMINATING SET IN D-DEGENERATE GRAPHS $(d \geq 2)$ [18]
input : a d-degenerate graph $G = (V, E)$, $k \in \mathbb{N}$
question : is there a connected dominating set of size at most k in G ?
parameter : k

Method : polynomial parameter transformation from COLORFUL GRAPH MOTIF.

CONNECTED DOMINATING SET IN GRAPHS OF GIRTH 5 OR 6 [36]
input : a graph $G = (V, E)$ of girth 5 or 6, $k \in \mathbb{N}$
question : is there a connected dominating set of size at most k in G ?
parameter : k

Method : polynomial parameter transformation from FAIR CONNECTED COLORS.

CONNECTED FEEDBACK VERTEX SET [37]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $C \subseteq V$ such that $G \backslash C$ is a cluster graph.
question : is there a set $F \subseteq V$ such that $|F| \leq k$, $G[F]$ is connected and $(V \backslash F, E)$ is acyclic
parameter : $|C|$.

Method : polynomial parameter transformation from CONNECTED VERTEX COVER.

CONNECTED FEEDBACK VERTEX SET IN D-DEGENERATE GRAPH $(d \geq 2)$ [18]
input : a d-degenerate graph $G = (V, E)$, $k \in \mathbb{N}$
question : is there a connected feedback vertex set of size at most k in G ?
parameter : k

Method : polynomial parameter transformation from CONNECTED VERTEX COVER.

CONNECTED ODD CYCLE TRANSVERSAL IN D-DEGENERATE GRAPH $(d \geq 2)$ [18]
input : a d-degenerate graph $G = (V, E)$, $k \in \mathbb{N}$
question : is there a set $S \subseteq V$ of size at most k in G such that $G[V \backslash S]$ is bipartite and $G[S]$ is connected ?
parameter : k

Method : polynomial parameter transformation from CONNECTED VERTEX COVER.

CONNECTED VERTEX COVER [19]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$
question : does G contains a vertex cover $C \subseteq V$ of size at most k which induces a connected graph ?
parameter : k

Method : polynomial parameter transformation from RED-BLUE DOMINATING SET

DIRECTED HAMILTONIAN CYCLE [4]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $X \subseteq V$ such that $G[V \backslash X] \in BiPaths$
question : does G contains a hamiltonian cycle of size k ?
parameter : $|X|$

Method : cross-composition from HAMILTONIAN S-T PATH IN DIRECTED BIPARTITE GRAPHS.

DISJOINT-CONSISTENCE [41]
input : two of variables X, Y over domains D (from a CSP)
question : is the Disjoint constraint consistent ?
parameter : size of the intersection of the domains of X and Y.

Method : polynomial parameter transformation from SAT parameterized by the number of variables.

DISJOINT CYCLE [7]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$
question : does G contains k vertex-disjoint cycles ?
parameter : k

Method : polynomial parameter transformation from DISJOINT FACTORS.

DISJOINT PATH [7]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$
question : does G contains k vertex-disjoint paths ?
parameter : k

Method : then polynomial parameter transformation from DISJOINT FACTORS.

DISJOINT SETS [19]
input : a set family $\mathcal{F}$ over a universe U with every set $S \in \mathcal{F}$ having size at most d, $k \in \mathbb{N}$.
question : Is there a subfamily $\mathcal{F}'$ of $\mathcal{F}$ of size at least k such that for every pair of sets S_1, S_2 in $\mathcal{F}'$ we have $S_1 \cap S_2 \neq \emptyset$
parameter : $k + d$

Method : polynomial parameter transformation from BIPARTITE REGULAR PERFECT CODE.

DODGSON SCORE [22]
input : a set C of candidates, a distinguished candidate $x \in C$, a multiset V of votes (a vote is a strict total order over C), $k \in \mathbb{N}$
question : Can x be made a Condorcet winner by making at most k swaps between adjacent candidates ?
parameter : k

Method : polynomial parameter transformation from SET COVER parameterized by $k + |U|$.

K-DOMATIC PARTITION ★
input : a graph $G = (V, E)$, a set $X \subseteq V$ such that $G[V \setminus X]$ is a disjoint union of paths
question : Is there a k-domatic partition of G ?
parameter : $|X|$

Method : Polynomial parameter tranformation from 3-COLORING parameterized by the size of a modulator for the Path class.

DOMINATING SET [19]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $C \subseteq V$ a vertex cover of G.
question : does G contains a dominating set $D \subseteq V$ of size at most k ?
parameter : $(|C|, k)$

Method : OR-composition for the colored version of the problem, then polynomial parameter transformation.

DOMINATING SET ★
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, a set $X \subseteq V$ such that $G[V \setminus X]$ is a cluster graph.
question : does G contains a dominating set $D \subseteq V$ of size at most k ?
parameter : $|X|$

Method : Cross-composition from VERTEX COVER

DOMINATING SET [1]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $\mathcal{T}$ a tree decomposition of G.
question : does G contains a dominating set $D \subseteq V$ of size at most k ?
parameter : width of $\mathcal{T}$.

Method : polynomial parameter transformation from COLORED HITTING SET.

DOMINATING SET IN D-DEGENERATED GRAPHS [1]
input : a d-degenerated graph $G = (V, E)$, $k \in \mathbb{N}$.
question : does G contains a dominating set $D \subseteq V$ of size at most k ?
parameter : (k, d).

Method : consequence of the non existence of polynomial kernel for the same problem parameterized by vertex cover.

DOMINATING SET IN H-MINOR FREE GRAPHS [1]
input : a graph H, and a H-minor free graph $G = (V, E)$.
question : does G contains a dominating set $D \subseteq V$ of size at most k ?
parameter : $(k, |H|)$.

Method : consequence of the non existence of polynomial kernel for the same problem parameterized by vertex cover.

FAIR CONNECTED COLORS [36]
input : A graph $G = (V, E)$, where the vertices V are properly colored with k colors in such a way that all neighbors of each vertex have distinct colors.
question : Does G contain a tree T on k vertices as a subgraph, where each vertex of T has a distinct color ?
parameter : $|k|$.

Method : OR-composition (disjoint union).

FEEDBACK VERTEX SET [2]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $C \subseteq V$ such that $G \backslash C$ is a co-cluster graph.
question : is there a set $F \subseteq V$ such that $|F| \leq k$ and $(V \backslash F, E)$ is acyclic
parameter : $|C|$.

Method : cross-composition from FEEDBACK VERTEX SET ON BIPARTITE GRAPHS OF GIRTH ≥ 6

FEEDBACK VERTEX SET [2]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $C \subseteq V$ such that $G \backslash C$ is a cluster graph.
question : is there a set $F \subseteq V$ such that $|F| \leq k$ and $(V \backslash F, E)$ is acyclic
parameter : $|C|$.

Method : cross-composition from INDEPENDENT SET.

HAMILTONIAN CYCLE [4]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $X \subseteq V$ such that $G[V \backslash X] \in Outerplanar$
question : does G contains a hamiltonian cycle of size k ?
parameter : $|X|$

Method : cross-composition from HAMILTONIAN S-T PATH IN BIPARTITE GRAPHS.

INDEPENDENT SET [1]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, a tree decomposition $\mathcal{T}$ of G.
question : is there an independent set of G of size at least k ?
parameter : width of $\mathcal{T}$

Method : OR-composition of the refinement version of the problem, then polynomial parameter transformation.

INDEPENDENT SET [2]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $X \subseteq V$ such that $G[V \setminus X]$ belongs to a graph class which contains all cliques.
question : is there an independent set of G of size at least k ?
parameter : $|X|$

Method : Consequence of the non existence of polynomial kernel for CLIQUE parameterized by the size of a vertex cover.

K-LEAF OUT BRANCHING [23]
input : a digraph $G = (V, E)$, $k \in \mathbb{N}$.
question : Does G contain a rooted oriented spanning tree with at least k leaves ?
parameter : k

Method : polynomial parameter transformation from K-LEAF OUT TREE.

K-LEAF OUT TREE [23]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$.
question : Does G contain a spanning tree with at least k leaves ?
parameter : k

Method : OR-composition.

LONGEST CYCLE [1]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$
question : is there a cycle of size k in G ?
parameter : k

Method : OR-composition (disjoint union).

LONGEST PATH [1]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$
question : is there a path of size k in G ?
parameter : k

Method : OR-composition (disjoint union).

MINOR ORDER TEST [1]
input : two graphs G, H.
question : is H a minor of G ?
parameter : numeric encoding of H (position of H in some canonical ordering of simple graphs).

Method : OR-composition.

NODE DELETION FOR HEREDITARY AND NON TRIVIAL PROPERTY Π ★
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $\mathcal{T}$ a tree decomposition of G.
question : is there a set $X \subset V$ of size at most k such that $G[V \setminus X]$ has the property Π ?

parameter : width of $\mathcal{T}$

Method : OR-composition for a refinement version of the problem, then polynomial parameter transformation.

NON-DETERMINISTIC TURING MACHINE COMPUTATION [1]
input : a non-deterministic turing machine M with alphabet size σ, and $k \in \mathbb{N}$.
question : Does M have a computation path halting on the empty input in at most k steps ?
parameter : (k, σ).

Method : OR-composition.

NOT-1-IN-3 SAT [34]
input : a conjunction F of not-1-in-3 constraints, $k \in \mathbb{N}$.
question : is there an feasible assignment for F with at most k ones ?
parameter : k.

Method : OR-composition.

NVALUE-CONSISTENCE [41]
input : a set of variables V over domains D (from a CSP)
question : is the NValue constraint consistent ?
parameter : size of the union of all domains.

Method : polynomial parameter transformation from SAT parameterized by the number of variables.

PLANAR GRAPH INDUCED SUBGRAPH TEST [1]
input : a graph H, a planar graph G
question : is H an induced subgraph of G ?
parameter : $|H|$

Method : OR-composition.

PLANAR GRAPH SUBGRAPH TEST [1]
input : a graph H, a planar graph G
question : is H a subgraph of G ?
parameter : $|H|$

Method : OR-composition.

POSITIVE BAYESIAN NETWORK INFERENCE [41]
input : a boolean bayesian netword (G, T), a variable v, a loop cutset L
question : $Pr(v = true) > 0$?
parameter : $|L|$.

Method : polynomial parameter transformation from SAT parameterized by the number of variables.

RED-BLUE DOMINATING SET [19]
input : a graph $G = (T \cup N, E)$, $k \in \mathbb{N}$
question : is there a set $N' \subseteq N$ such that $|N'| \leq k$ and N' dominates T ?
parameter : $(|T|, k)$

Method : OR-composition for the colored version of the problem, then polynomial parameter transformation.

SAT [25]
input : a CNF formula F
question : is F satisfiable ?
parameter : number of variables

Method : would imply a distillation algorithme for SAT.

SAT [41]
input : a CNF formula F, an $\mathcal{A}$-backdoor $\mathcal{B}$ of F
question : is F satisfiable ?
parameter : size of $\mathcal{B}$

Method : polynomial parameter transformation from SAT parameterized by the number of variables.

SET COVER [19]
input : a set family $\mathcal{F}$ over an universe U with $|U| \leq d$, $k \in \mathbb{N}$
question : is there a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ such that $|\mathcal{F}'| \leq k$ and $\bigcup_{S \in \mathcal{F}'} S = U$?
parameter : (k, d)

Method : polynomial parameter transformation from RED-BLUE DOMINATING SET

STABLE MODEL EXISTENCE [41]
input : a normal logic program P, a feedback set V of P.
question : has P a stable model ?
parameter : $|V|$

Method : polynomial parameter transformation from SAT parameterized by the number of variables.

STEINER TREE [19]
input : a graph $G = (T \cup N, E)$, $k \in \mathbb{N}$
question : is there a set $N' \subseteq N$ such that $|N'| \leq k$ and $G[T \cup N']$ is connected ?
parameter : $(|T|, k)$

Method : polynomial parameter transformation from RED-BLUE DOMINATING SET

STEINER TREE IN D-DEGENERATE GRAPH $(d \geq 2)$ [18]
input : a graph $G = (V, E)$, a set of terminals $T \subseteq V$, $k \in \mathbb{N}$.
question : Does there exist $S \subset V$ of size at most k such that $G[S \cup T]$ is connected ?
parameter : $(|T|, k)$

Method : polynomial parameter transformation from COLORFUL GRAPH MOTIF.

SUBSET SUM [19]
input : a set S of n integers such that $\max_{x \in S} \leq 2^d$, $t \in \mathbb{N}$, $k \in \mathbb{N}$
question : is there a subset $S' \subseteq S$ such that $|S'| \leq k$ and $\sum_{x \in S'} x = t$?
parameter : (k, d)

Method : polynomial parameter transformation from COLORED RED-BLUE DOMINATING SET.

TOTAL VERTEX COVER [30]
input : a graph $G = (V, E)$, $k, t \in \mathbb{N}$ with $k \geq t$.
question : does there exist a set $S \subseteq V$ of size at most k such that S is a vertex cover of G and each connected component of (V, S) contains at least t edges of S ?
parameter : k

Method : polynomial parameter transformation from RED-BLUE DOMINATING SET.

TREE CONTRACTABILITY [29]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$
question : can we transform G to a tree by contracting at most k edges.
parameter : k

Method : polynomial parameter transformation from RED-BLUE DOMINATING SET.

TREEWIDTH [5]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $X \subseteq V$ such that $G[V \backslash X]$ is a co-cluster graph
question : Does G has treewidth at most k ?
parameter : $|X|$

Method : cross-composition from TREEWIDTH.

UNIQUE COVERAGE [19]
input : a set family $\mathcal{F}$ over an universe U, $k \in \mathbb{N}$
question : is there a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ and a set $S' \subseteq U$ such that $|S| \geq k$ and every element of S' appears in exactly one set in $\mathcal{F}'$.
parameter : k

Method : OR-composition for the colored version of the reduced problem (reduction via a kernelization), then polynomial parameter transformation.

USES-CONSISTENCE [41]
input : two of variables X, Y over domains D (from a CSP)
question : is the Uses constraint consistent ?
parameter : size of the domain of Y.

Method : polynomial parameter transformation from SAT parameterized by the number of variables.

VERTEX COVER [1]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $\mathcal{T}$ a tree decomposition of G.
question : Does G contains a vertex cover of size at most k ?
parameter : width of $\mathcal{T}$

Method : Polynomial parameter transformation from INDEPENDENT SET parameterized by treewidth.

VERTEX COVER [2]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $X \subseteq V$ such that $G[V \backslash X]$ belongs to a graph class which contains all cliques.
question : Does G contains a vertex cover of size at most k ?
parameter : $|X|$

Method : Consequence of the non existence of polynomial kernel for CLIQUE parameterized by vertex cover.

WEIGHTED FEEDBACK VERTEX SET [2]
input : a graph $G = (V, E)$, $k \in \mathbb{N}$, $w : V \to \mathbb{N}$, C a vertex cover of G.
question : is there a set $F \subseteq V$ such that $\sum_{x \in F} w(x) \leq k$ and $(V \backslash F, E)$ is acyclic
parameter : $|C|$

Method : cross-composition from FEEDBACK VERTEX SET ON BIPARTITE GRAPHS.

WEIGHTED INDEPENDENT SET [30]
input : a graph $G = (V, E)$, $w : V \to \mathbb{N}$, $k \in \mathbb{N}$, F a feedback vertex set of G.
question : is there an independent set of weight at most k of G ?
parameter : $|F|$

Method : OR-composition for T-PAIRED VECTOR AGREEMENT, then polynomial parameter transformation.

WEIGHTED TREEWIDTH [5]
input : a graph $G = (V, E)$, $w : V \to \mathbb{N}$, $k \in \mathbb{N}$, X a vertex cover of G.
question : Does G has a weighted treewidth of weight at most k ?
parameter : $|X|$

Method : cross-composition from TREEWIDTH.

WEIGHTED VERTEX COVER [30]
input : a graph $G = (V, E)$, $w : V \to \mathbb{N}$, $k \in \mathbb{N}$, F a feedback vertex set of G.
question : is there a vertex cover of weight at most k in G ?
parameter : $|F|$

Method : OR-composition for T-PAIRED VECTOR AGREEMENT, then polynomial parameter transformation.

Bibliographie

[1] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8) :423–434, 2009.

[2] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Cross-composition : A new technique for kernelization lower bounds. In *STACS*, pages 423–434, 2011.

[3] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Data reduction for graph coloring problems. In *(submitted)*, 2011.

[4] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernel bounds for path and cycle problems. In *(submitted)*, 2011.

[5] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Preprocessing for Treewidth : A Combinatorial Analysis through Kernelization. In *ICALP*, 2011.

[6] H. L. Bodlaender, S. Thomassé, and A. Yeo. Analysis of data reduction : Transformations give evidence for non-existence of polynomial kernels. Technical report, 2008.

[7] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *ESA*, pages 635–646, 2009.

[8] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58 :171–176, 1996.

[9] L. Cai. Parameterized complexity of vertex colouring. *Discrete Appl. Math.*, 127 :415–429, 2003.

[10] J. Chen, H. Fernau, I. Kanj, and G. Xia. Parameteric duality and kernelization : lower bounds and upper bounds on kernel size. In *STACS*, pages 269–280, 2005.

[11] J. Chen, I. A. Kanj, and W. Jia. Vertex cover : Further observations and further improvements. *J. Algorithms*, 41(2) :280–301, 2001.

[12] Y. Chen, J. Flum, and M. Müller. Lower bounds for kernelizations and other preprocessing procedures. In *CiE*, pages 118–128, 2009.

[13] M. Chlebík and J. Chlebíková. Crown reductions for the minimum weighted vertex cover problem. *Discrete Appl. Math.*, 156 :292–312, 2008.

[14] E. J. Cockayne and H. S. T. Optimal domination in graphs. *IEEE Trans. Circuits and Systems*, pages 855–857.

[15] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, 1971.

[16] B. Courcelle. The monadic second-order logic of graphs I. recognizable sets of finite graphs. *Information and Computation*, pages 12–75, 1990.

[17] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Appl. Math.*, 101 :77–114, 2000.

[18] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. Kernelization hardness of connectivity problems in d-degenerate graphs. In *WG*, pages 147–158, 2010.

[19] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *ICALP*, pages 378–389, 2009.

[20] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[21] U. Feige, M. M. Halldórsson, G. Kortsarz, and A. Srinivasan. Approximating the domatic number. *SIAM J. Comput.*, 32 :172–195, January 2003.

[22] M. R. Fellows, B. M. P. Jansen, D. Lokshtanov, F. A. Rosamond, and S. Saurabh. Determining the winner of a dodgson election is hard. In *FSTTCS*, pages 459–468, 2010.

[23] H. Fernau, F. V. Fomin, D. Lokshtanov, D. Raible, S. Saurabh, and Y. Villanger. Kernel(s) for problems with no kernel : On out-trees with many leaves. *CoRR*, abs/0810.4796, 2008.

[24] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[25] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1) :91–106, 2008.

[26] M. R. Garey and D. S. Johnson. *Computers and Intractability ; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[27] S. Guillemot, C. Paul, and A. Perez. On the (non-)existence of polynomial kernels for P_l-free edge modification problems. In *IPEC*, pages 147–157, 2010.

[28] J. Guo, F. Hüffner, and R. Niedermeier. A structural view on parameterizing problems : Distance from triviality. In *IWPEC*, pages 162–173, 2004.

[29] P. Heggernes, P. van 't Hof, B. Lévêque, D. Lokshtanov, and C. Paul. Contracting graphs to paths and trees. *(submitted)*, 2011.

[30] B. M. P. Jansen and H. L. Bodlaender. Vertex cover kernelization revisited : Upper and lower bounds for a refined parameter. In *STACS*, pages 177–188, 2011.

[31] I. A. Kanj, M. J. Pelsmajer, and M. Schaefer. Parameterized algorithms for feedback vertex set. In R. G. Downey, M. R. Fellows, and F. K. H. A. Dehne, editors, *IWPEC*, volume 3162, pages 235–247, 2004.

[32] H. Kaplan and R. Shamir. The domatic number problem on some perfect graph families. *Information Processing Letters*, 49 :51–56, 1995.

[33] M. Koivisto. An $o^*(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion–exclusion. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 583–590, 2006.

[34] S. Kratsch and M. Wahlström. Two edge modification problems without polynomial kernels. In *IWPEC*, pages 264–275, 2009.

[35] J. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. System Sci.*, 20(i2) :219–230, 1980.

[36] N. Misra, G. Philip, V. Raman, and S. Saurabh. The effect of girth on the kernelization complexity of Connected Dominating Set. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8, pages 96–107, 2010.

[37] N. Misra, G. Philip, V. Raman, S. Saurabh, and S. Sikdar. FPT algorithms for connected feedback vertex set. *CoRR*, abs/0909.3180, 2009.

[38] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications 31, 2002.

[39] S. V. Pemmaraju and I. A. Pirwani. Energy conservation via domatic partitions. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 143–154. ACM, 2006.

[40] N. Robertson and P. D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *J. Algorithms*, 7(3) :309–322, 1986.

[41] S. Szeider. Limits of preprocessing. *CoRR*, abs/1104.5566, 2011.

[42] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2) :32 :1–32 :8, 2010.