Construction et analyse des algorithmes exacts et exponentiels : énumération input-sensitive

Mohamed Yosri Sayadi

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Construction et analyse des algorithmes exacts et exponentiels : énumération input-sensitive

THÈSE

présentée et soutenue publiquement le 04 novembre 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Mohamed Yosri Sayadi

Composition du jury

Présidents :	Hoai An LE THI	
Rapporteurs :	Bruno ESCOFFIER	Professeur des Universités, Sorbonne Université
	Lhouari NOURINE	Professeur des Universités, Université Clermont Auvergne
Examinateurs :	Hoai An LE THI	Professeur des Universités, Université de Lorraine
	Michaël RAO	Chargé de Recherche CNRS, ENS de Lyon
Directeur de Thèse :	Loïc COLSON	Professeur des Universités, Université de Lorraine

Laboratoire de Génie Informatique, de Production et de Maintenance - EA 3096

لا نقدر على التمام يا سيدي

ناقصون..

لا نستطيع ان نكون أكثر

" محمد التركي "

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction

« *For every polynomial-time algorithm you have, there is an exponential algorithm that I would rather run* »

Alan Perlis, the first Turing Award winner

Sommaire

1.1 Graphes

1.1.1 Notions de graphes

Soit V un ensemble fini de sommets. Soit P(V) l'ensemble des parties de l'ensemble V, c'est-à-dire l'ensemble des sous-ensembles de V, et soit $P_2(V)$ l'ensemble des parties de cardinalité 2 de V.

Le graphe G = (V, E) est un graphe simple non orienté si $E \subseteq P_2(V)$. On appelle E l'ensemble des arêtes de G. On désigne par n le nombre de sommets |V|, et par m le nombre des arêtes |E|. Un graphe G est non trivial s'il a au moins une arête.

Le degré d'un sommet v, noté $deg(v)$, est le nombre d'arêtes reliant v. Un sommet de degré 0 est dit isolé, alors qu'un sommet de degré 1 est dit pendant. On désigne par $\delta(G)$ et $\Delta(G)$, respectivement le degré minimum et maximum d'un graphe G.

Dans le cas d'un graphe coloré, on désigne par $col(v)$ la couleur du sommet $v \in V$.

Deux sommets d'un graphe sont adjacents ou voisins s'il existe une arête qui les relie. On désigne par $N(v)$ l'ensemble des sommets adjacents à v, et par $N[v]$ l'ensemble $N(v) \cup \{v\}$.

Soit S un ensemble de sommets. On désigne par N(S), l'ensemble $\cup_{v \in S} N(v)$ et $N[S] = S \cup N(S)$. Un sommet u est un voisin privé du sommet v, par rapport à S, si $u \in N[v]$ mais $u \notin N[S \setminus \{v\}]$.

Une séquence de sommets $(v_0, v_1, \ldots, v_l)$ est un chemin de v_0 à v_l, noté $v_0 - v_l$, de longueur l dans G si $v_{i-1}v_i \in E$ pour $i = 1, 2 \ldots, l$. Un chemin $(v_0, v_1, \ldots, v_l)$ est un cycle si $v_0 v_l \in E$. Une corde d'un chemin (cycle) est une arête entre deux sommets non consécutifs du chemin (cycle). La distance entre deux sommets u et v est la longueur minimale d'un chemin $u - v$.

Un graphe $G' = (V', E')$ est un sous-graphe de G si $V' \subseteq V$ et $E' \subseteq E$. Un sous-graphe $G' = (V', E')$ est un sous-graphe induit de G si $E' = \{uv : uv \in E\}$ et $u, v \in V'$. On désigne par G[S] le sous-graphe induit de G par S.

S'il existe un chemin pour toute paire de sommets, on dit que le graphe est connexe. Une composante connexe est un sous-graphe connexe maximal. on note $c(G)$ le nombre de composantes connexes dans G. Un sommet v est un point d'articulation de G si $c(G \setminus v) > c(G)$. Une arête e d'un graphe G est un pont si $c(G \setminus e) > c(G)$.

Un graphe de n sommets est dit complet, noté K_n, si chaque paire de sommets distincts est reliée par une arête. On appelle un sous-graphe complet de G une clique. Réciproquement, un ensemble stable ou « independent set » est un sous-ensemble de V où les sommets sont deux à deux non adjacents. On désigne par $w(G)$ la taille de la clique de cardinalité maximale du graphe G.

Noter bien qu'on désigne, tout au long de cette thèse, par $\subseteq$ l'inclusion au sens large et par $\subset$ l'inclusion au sens strict.

1.1.2 Sous-ensembles de sommets

Dans cette section, on cite quelques définitions de sous-ensembles de sommets particuliers.

Définition 1. *Stable maximal*

Soit G = (V, E) *un graphe et* $S \subseteq V$ *un stable de* G. *L'ensemble* S *est un stable maximal s'il n'existe aucun autre stable* $S' \subset V$ *tel que* $S \subset S'$.

Définition 2. *Ensemble dominant minimal*

Soit $G = (V, E)$ *un graphe. Un ensemble de sommets* $D \subseteq V$ *est un ensemble dominant de* G *si pour tout* $v \in V$, $v \in N[D]$. *On dit que* D *est minimal s'il n'existe aucun autre ensemble dominant* $D' \subset V$ *tel que* $D' \subset D$.

Définition 3. *Ensemble dominant connexe minimal*

Soit $G = (V, E)$ *un graphe. Un ensemble de sommets* $D \subseteq V$ *est un ensemble dominant connexe de* G *si* D *est un ensemble dominant et* G[D] *est connexe. L'ensemble* D *est minimal s'il n'existe aucun autre ensemble dominant connexe* $D' \subset V$ *tel que* $D' \subset D$.

Définition 4. *Ensemble dominant total minimal*

Soit $G = (V, E)$ *un graphe. Un ensemble de sommets* $D \subseteq V$ *est un ensemble dominant total de* G *si pour tout* $v \in V$, $v \in N(D)$. *L'ensemble* D *est minimal s'il n'existe aucun autre ensemble dominant total* $D' \subset V$ *tel que* $D' \subset D$.

Définition 5. *Ensemble irredondant maximal*

Soit $G = (V, E)$ *un graphe. Un ensemble de sommets* $S \subseteq V$ *est un ensemble irredondant si pour tout* $v \in S$, v *a un voisin privé dans* G. *L'ensemble* S *est maximal s'il n'existe aucun autre ensemble irredondant* S' *tel que* $S \subset S'$.

Définition 6 ([20])**.** *Ensemble tropical connexe minimal*

Soit $G = (V, E)$ *un graphe où ses sommets sont colorés. Un ensemble* $S \subseteq V$ *est dit tropical s'il contient toutes les couleurs du graphe. Il est dit tropical connexe minimal si l'ensemble* S *est de plus connexe et minimal.*

1.1.3 Les classes de graphes

Plusieurs classes de graphes sont distinguées dans la littérature par leurs structures et leurs propriétés caractéristiques qui peuvent nous aider à améliorer certains algorithmes. Dans cette section, on cite quelques classes de graphes et on donne un diagramme qui permet d'illustrer les différentes relations entre eux.

Les chemins P_n et les cycles C_n

On désigne par P_n (C_n) un chemin (cycle) sans corde avec n sommets.

Les graphes sans triangles

Un graphe sans triangle est un graphe qui ne contient pas de C_3.

Bipartis

Un graphe $G = (V, E)$ est un graphe biparti s'il existe deux ensembles de sommets $U, W \subset V$ tel que $V = U \cup W$ et U, V sont deux stables. On désigne un tel graphe par $G = (U, W, E)$.

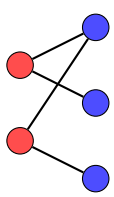

FIGURE 1.1 – Un graphe biparti

Bipartis Complets

Un graphe biparti G = (U, W, E) est dit complet si chaque sommet de U est adjacent à tous les sommets de W. Un graphe biparti complet est noté $K_{|U|,|W|}$.

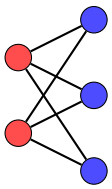

FIGURE 1.2 – Un graphe biparti complet

Sans griffes

Un graphe G = (V, E) est dit sans griffe s'il n'a pas un $K_{1,3}$ comme un sous graphe induit.

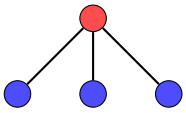

FIGURE 1.3 – Un griffe

Cobipartis

Un graphe G = (K_1, K_2, E) est dit cobiparti si l'ensemble de ses sommets V peut être partitionné en deux cliques K_1 et k_2.

Bipartis cordaux

Un graphe biparti cordal G = (U, W, E) est un graphe biparti où chaque cycle de longueur au moins 6 possède une corde.

Bipartis convexes

Un graphe biparti G = (U, W, E) est convexe s'il existe un ordre sur les sommets W tel que pour tout sommet $u \in U$, ses adjacents $N(u)$ sont consécutifs dans W. Les adjacents d'un sommet $u \in U$ peuvent être répresentés par un intervalle I_u, appelé l'intervalle adjacent de u. Par conséquent, les adjacents des sommets U peuvent être répresentés par un ensemble des intervalles noté I(U).

Sans P_n

Un graphe G = (V, E) est dit sans P_n s'il ne contient pas le chemin induit P_n.

Cographes

Un graphe sans P_4 est dit cographe.

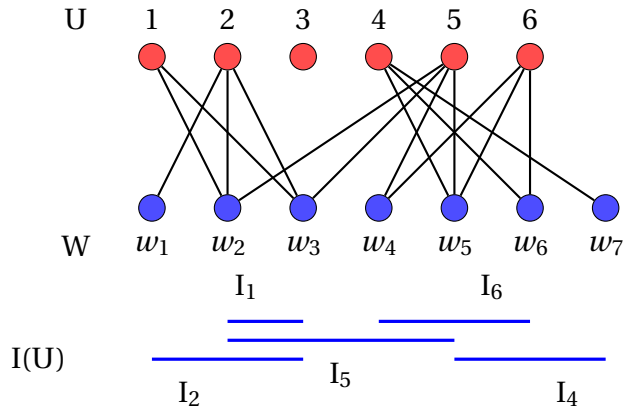

FIGURE 1.4 – Un graphe biparti convexe

Cordaux

Dans la littérature, les graphes cordaux sont appelés aussi « triangulated », « rigid-circuit », « perfect elimination graphs »... Un graphe G est cordal si chaque cycle dans G de longueur au moins 4 possède au moins une corde [36]. On dit qu'un sommet x est simplicial si $N[x]$ forme une clique. Chaque graphe cordal non complet possède au moins deux simpliciaux non adjacents (DIRAC [21]). On peut associer à chaque graphe cordal, un arbre de cliques.

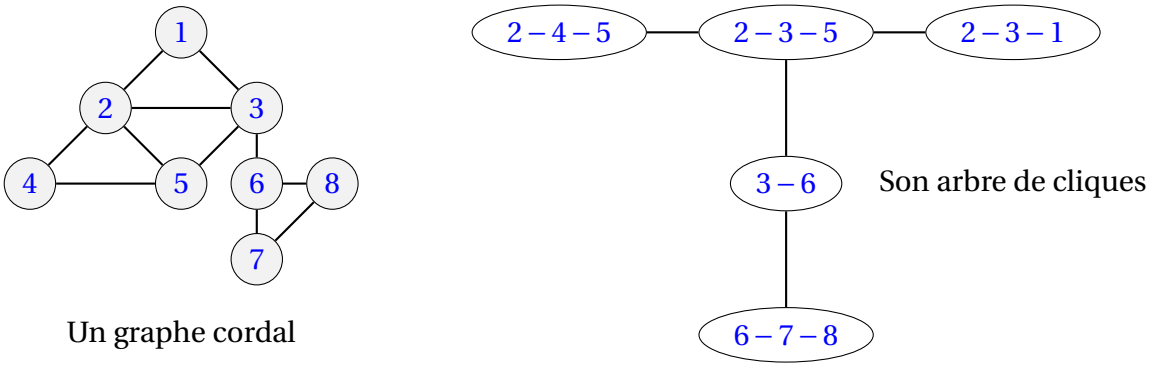

FIGURE 1.5 – Un graphe cordal et son arbre de cliques

Récemment ABU-KHZAM et HEGGERNES [1] ont prouvé des nouvelles propriétés importantes sur les simpliciaux d'un graphe cordal.

Définition 7. *Si u et v sont deux simpliciaux adjacents alors* $N[u] = N[v]$. *On dit, dans ce cas, que u et v sont deux simpliciaux jumeaux.*

Définition 8 ([1])**.** *Soit v un sommet non simplicial dans un graphe cordal* G *qui a exactement un unique adjacent simplicial u. Si v est simplicial dans le graphe* $G - u$, *alors on dit que v est un semi-simplicial dans* G.

Dans la figure 1.5, le sommet « 5 » est un sommet non simplicial dans G et le sommet « 4 » est son unique adjacent simplicial. Comme le sommet « 5 » est simplicial dans $G - \{« 4 »\}$, alors il est un semi-simplicial dans G.

Observation 1 ([1])**.** *Soit v un sommet semi-simplicial dans un graphe cordal* G *et u son adjacent simplicial de degré au moins 2, alors* $N[V] \subseteq N[W]$ *pour tout sommet* $w \in N(u) \cap N(v)$.

Théorème 1 ([1]). *Soit* G *un graphe cordal non trivial, alors il existe une paire de sommets u et v satisfaisant au moins l'une de ces trois assertions suivantes :*

1. *Les deux sommets u et v sont deux simpliciaux tel que* $N(u) \cap N(v) \neq \emptyset$.
2. *Les deux sommets u et v sont deux simpliciaux jumeaux.*
3. *Le sommet u est simplicial, v son adjacent semi-simplicial et le degré de v dans* G *est au moins 2.*

Corollaire 1 ([1]). *Soit* G *un graphe cordal non trivial sans simpliciaux jumeaux, tous les simpliciaux sont de degré 1, et tous deux sommets de degré 1 n'ont pas un voisin commun. Alors au moins l'une de ces deux assertions suivantes est satisfaisante :*

1. *Il existe deux simpliciaux* u, u' *et respectivement* v, v' *sont ses deux distincts semi-simpliciaux tel que* $N[v] \cap N[v'] \neq \emptyset$.
2. *Il existe trois sommets* u, v *et* w *tel que u est un simplicial dans* G, *v son semi-simplicial adajcent, et* $w \in N(v)$ *est un simplicial dans* $G - v$ *et de degré au moins 2 dans* $G - u$.

Ces propriétés précédentes proposés par Faisal N.Abu-Khzam et Pinar Heggernes sont très importantes du point de vue algorithmique, en plus ils concernent une classe de graphes importante, les graphes cordaux. Durant cette thèse on va proposer un théorème légèrement différent du théorème 1 et du corollaire 1 mais qui est plus compact et plus fort.

Intervalles

Un graphe d'intervalles G = (V, E) est présenté par un ensemble d'intervalles dans lequel chaque sommet correspond à un intervalle de la droite réelle. Deux sommets sont adjacents dans le graphe G si et seulement si leurs intervalles correspondants ont une intersection non vide.

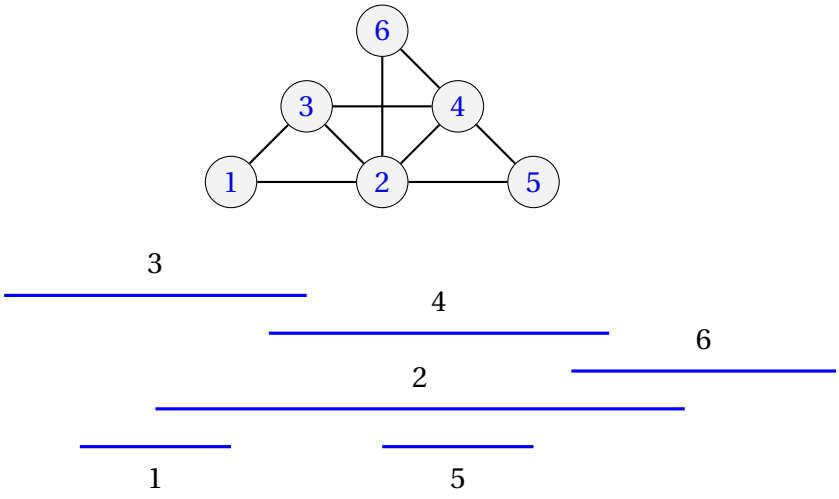

FIGURE 1.6 – Un graphe d'intervalles

Les graphes scindés

Un graphe G = (K, I, E) est dit scindé ou « split » si l'ensemble de ses sommets V peut être partitionné en une clique K et un stable I.

Biconnexes

Un graphe est dit biconnexe s'il est connexe et s'il ne contient pas des points d'articulation.

Blocs [38]

Un bloc d'un graphe est soit un sous-graphe biconnexe maximal, soit un pont, soit un sommet isolé [50]. Un graphe est dit bloc si et seulement si chaque bloc est une clique. Deux blocs distincts dans un graphe bloc ont au plus un sommet en commun. Les graphes blocs sont des graphes cordaux.

Forêts

Une forêt est un graphe sans cycles. Remarquons qu'une forêt est un graphe cordal.

Arbres

Un arbre est une forêt connexe. Un sommet dans un arbre est dit une feuille s'il est de degré 1. Un arbre a au moins deux feuilles (conséquence immédiate du théorème de DIRAC [21]).

Les graphes de permutation

On dit que G = (V, E) est un graphe de permutation s'il existe deux permutations σ et π de l'ensemble de sommets V telles que $xy \in E(G)$ si et seulement si $\sigma(x) < \sigma(y)$ et $\pi(x) > \pi(y)$.

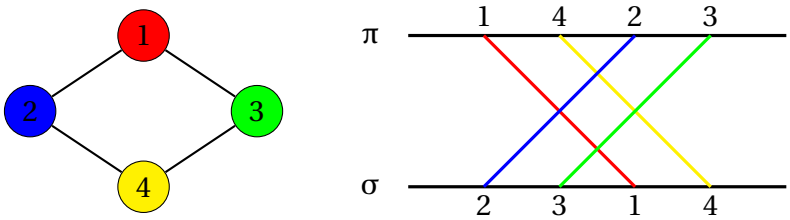

FIGURE 1.7 – Un graphe de permutation

Les seuils

On dit que G = (V, E) est un graphe à seuil (threshold graph) s'il existe un sommet $v \in V$ tel que :

— Le sommet v est soit isolé soit universel (un sommet adjacent à tous les autres sommets du graphe G).

— Le sous-graphe de G induit par $V \setminus \{v\}$ est un graphe à seuil.

Les seuils ont été introduits pour la première fois par CHVÁTAL et HAMMER [14].

Les trivialement parfaits

Ce sont les cographes qui sont aussi des graphes d'intervalles [45]. Donc ils sont une sous-classe des graphes cordaux. Alors un graphe trivialement parfait est un graphe sans P_4 et sans C_4.

Les chaines

Un graphe biparti G = (U, W, E) est dit chaine s'il existe un ordre $\sigma_U =< u_1, u_2, \ldots, u_k >$ de ses sommets U tel que $N(u_i) \subseteq N(u_{i+1})$ pour tout $i \in \{1, \ldots, k-1\}$, ainsi qu'un ordre $\sigma_W =< w_1, w_2, \ldots, w_l >$ de ses sommets W tel que $N(w_j) \supseteq N(w_{j+1})$ pour tout $j \in \{1, \ldots, l-1\}$.

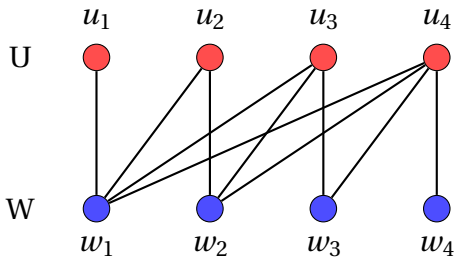

FIGURE 1.8 – Un graphe chaine

Les fortement cordaux

Un graphe cordal est dit fortement cordal si chaque cycle C de longueur paire supérieure ou égale à 6 possède une corde xy telle que la distance entre x et y est impaire dans le cycle C.

Les distance-héréditaires

Un graphe G est distance-héréditaire si la distance entre deux sommets quelconques d'un sous-graphe de G induit connexe est égale à la distance entre ces deux sommets dans G [42].

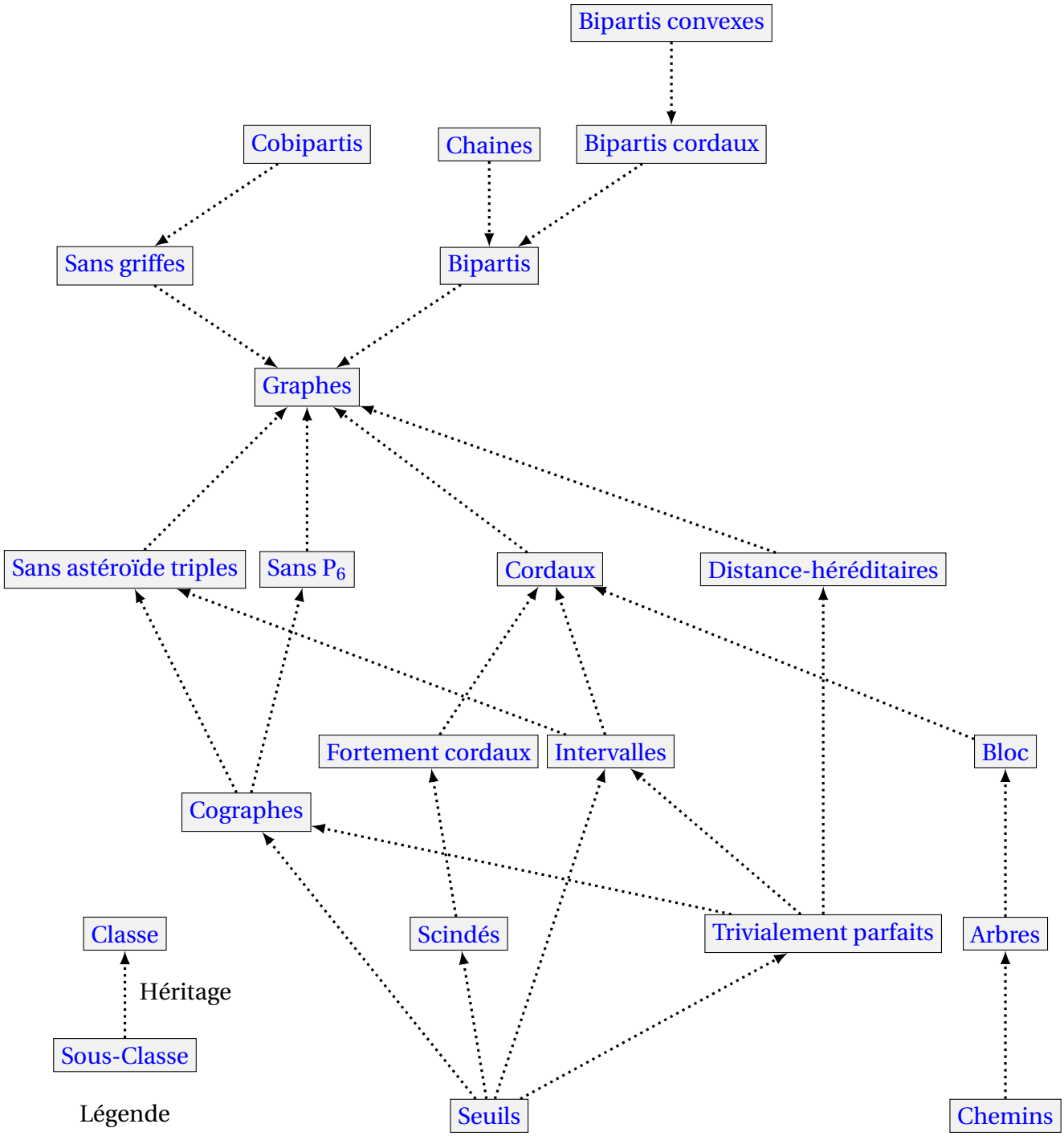

FIGURE 1.9 – Un diagramme de classes de graphes

1.2 Quelques problèmes

Une solution qui satisfait certains critères est dite réalisable. Un stable maximal (Définition :1) est un sous-ensemble de sommets mais qui satisfait les conditions de stabilité et de maximalité. Le nombre de solutions réalisables est parfois exponentiel, on dit dans ce cas que notre problème est combinatoire. En fait l'ensemble de stables maximaux, par exemple, peut atteindre $3^{n/3}$, ce qui est démontré par MOON et MOSER [55].

Démonstration. Soit $G = (V, E)$ un graphe constitué par union de k C_3. Chaque $i$ème composante est désigné par T_i pour tout $i \in \{1, \dots k\}$. Le graphe G a $n = 3k$ sommets. Soit $S \subseteq V$ est un stable maximal de G si et seulement si $|S \cap T_i| = 1$ pour tout $i \in \{1, \dots k\}$. Donc le nombre de stables maximaux dans G est $3^k = 3^{n/3}$. □

Ce nombre exponentiel des solutions réalisables est l'origine de la diversité de la nature des problèmes combinatoires (décision, optimisation, dénombrement, énumération).

1.2.1 Décision

Un problème de décision est un problème qui nécessite une réponse soit par « oui » soit par « non ».

EXEMPLE : ENSEMBLE STABLE

entrée:	Un graphe $G = (V, E)$ et un entier $k \in \mathbb{N}$.
question:	Existe-t-il un ensemble $S \subseteq V$ tel que S est un ensemble stable de taille k ?

1.2.2 Optimisation

Un problème d'optimisation consiste à trouver la meilleure solution réalisable satisfaisant certains critères. La meilleure solution est appelée la solution optimale. La solution optimale est la solution qui réalise le maximum du profit ou qui minimise au maximum le coût.

EXEMPLE : ENSEMBLE STABLE MAXIMUM

entrée:	Un graphe $G = (V, E)$.
question:	Chercher un ensemble $S \subseteq V$ de taille maximum tel que S est un ensemble stable ?

1.2.3 Enumération

Un problème d'énumération consiste à énumérer explicitement tous les solutions réalisables possibles.

EXEMPLE : ENUMÉRATION DES ENSEMBLES STABLES MAXIMAUX

entrée:	Un graphe $G = (V, E)$.
question:	Lister tous les ensembles stables maximaux de G

1.2.4 Dénombrement

L'objectif d'un problème de dénombrement est de compter le nombre de solutions possibles satisfaisant certains critères sans les énumérer.

EXEMPLE : DÉNOMBREMENT DES ENSEMBLES STABLES MAXIMAUX

entrée: Un graphe $G = (V, E)$.
question: Quel est le nombre exact des ensembles stables maximaux dans G?

L'énumération des tous les ensembles réalisables d'un problème donne immédiatement la solution optimale et le nombre exact de tous les solutions. Donc on peut réduire les problèmes d'optimisation et de dénombrement aux problèmes d'énumération. Cela prouve que les problèmes d'énumération sont plus difficiles. Donc il y a des niveaux hiérarchiques des problèmes selon leur difficultés. C'est ce qui nous conduit à parler, dans la section suivante, de la notion de complexité.

1.3 Complexité

Afin d'étudier la complexité de certains problèmes, plusieurs classes sont définies :

Définition 9. *On dit que le problème* Π *est au moins aussi difficile qu'un problème* Π' *s'il existe un algorithme en temps polynomial pour transformer les entrées du problème* Π' *en entrées du problème* Π *de sorte que ce dernier produit la même sortie que le problème d'origine.*

Définition 10 ([11])**.** *La classe* P *est l'ensemble des problèmes de décision pour lesquels, pour toute instance de tout problème, on peut déterminer par un algorithme polynomial si la réponse est « oui » ou « non ».*

Définition 11 ([11])**.** *La classe* NP *est l'ensemble des problèmes de décision pour lesquels, pour toute instance de tout problème, on peut vérifier par un algorithme polynomial appliqué à un « certificat succint » (i.e. un certificat de taille polynomiale par rapport à la taille de l'instance) que la réponse est « oui » si et seulement si l'instance est positive.*

Définition 12 ([11])**.** *Un problème de décision est dit* NP*-complet s'il est dans la classe* NP *et s'il est au moins aussi difficile que tout problème de la classe* NP*.*

En 1971, COOK [15] a montré le premier problème NP-complet qui a été le problème SAT. Puis, KARP [49] a montré, par réduction polynomiale (voir définition 9), 21 problèmes NP-complets. Et depuis le nombre de problèmes NP-complets n'a cessé de croitre.

Définition 13 ([11])**.** *Un problème est dit* NP*-difficile s'il est au moins aussi difficile qu'un problème* NP*-complet.*

La résolution de ces problèmes se fait par des algorithmes plus ou moins « efficaces », selon leur complexité.

1.4 Algorithmes

Un algorithme est un ensemble d'instructions qui prend en entrée une instance des données et permet de produire une ou plusieurs solutions du problème correspondant en sortie. Un seul problème peut avoir plusieurs algorithmes qui le résolvent mais qui se diffèrent par leur temps d'exécution. Et l'objectif toujours est de concevoir des algorithmes avec un temps d'exécution aussi court que possible. Notons bien que certains problèmes n'admettent aucun algorithme qui les résout, on dit dans ce cas que le problème (de décision) est non calculable (non décidable). Afin d'analyser le temps d'exécution d'un algorithme, on rappelle quelques notations.

1.4.1 Les notations asymptotiques : $\mathcal{O}, \Omega, \Theta$ et $\mathcal{O}^*$

Les notations asymptotiques permettent d'évaluer le temps d'éxécution d'un algorithme. On désigne par $T(n)$ le temps d'exécution maximal d'un algorithme pour une instance de donnée de taille n, et par $f(n)$ une fonction donnée.

La notation $\mathcal{O}$

Soit c et n_0 deux constantes positives. Si pour tout $n \geq n_0$ on a $T(n) \leq c.f(n)$, on dit alors que $T(n)$ est en $\mathcal{O}(f(n))$.

La notation Ω

Soit c et n_0 deux constantes positives. Si pour tout $n \geq n_0$ on a $T(n) \geq c.f(n)$, on dit alors que $T(n)$ est en $\Omega(f(n))$.

La notation Θ

Si une fonction $T(n)$ est à la fois en $\mathcal{O}(f(n))$ et en $\Omega(f(n))$, on dit que $T(n)$ est en $\Theta(f(n))$.

A ces notations popularisés par Donald Knuth, on ajoute la définition de $\mathcal{O}^*$.

La notation $\mathcal{O}^*$

On dit qu'un temps d'exécution $T(n)$ est en $\mathcal{O}^*(f(n))$, si $T(n)$ est $\mathcal{O}(f(n).poly(n))$.

Le temps d'exécution au pire cas, exprimé par un $\mathcal{O}$, permet de classifier les algorithmes.

1.4.2 Classification suivant le temps d'exécution

Soit $T(n)$ un temps d'exécution pour une instance de donnée de taille n. On peut dire que l'algorithme est logarithmique, linéaire, polynomial ou exponentiel si $T(n)$ est en $\mathcal{O}(log(n))$, $\mathcal{O}(n)$, $\mathcal{O}(n^d)$ pour un certain entier positif d, ou $\mathcal{O}(c^n)$ pour un certain réel $c > 1$ respectivement.

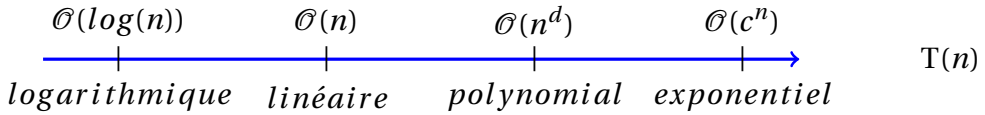

FIGURE 1.10 – Des différents temps d'exécution

Tout au long de ce manuscrit, on va adopter 4 chiffres après la virgule comme une précision pour le réel c des algorithmes exponentiels en $\mathcal{O}(c^n)$.

1.4.3 P = NP?

Le problème si P = NP ou P ≠ NP reste « non résolu » jusqu'à nos jours. Et si P ≠ NP, qui est très probable, un problème qui est dans NP et non dans P n'admet pas un algorithme polynomial pour le résoudre. La non existence des algorithmes polynomiaux pour résoudre les problèmes NP-difficile, n'empêche pas d'avoir plusieurs approches pour les attaquer.

1.4.4 Méthodes de résolution

Il existe plusieurs méthodes et approches pour attaquer les problèmes NP-difficile :

Les algorithmes d'approximation

Un algorithme approximatif est un algorithme qui permet de trouver une solution approchée du problème mais avec une garantie sur la qualité de la solution fournie par rapport à la solution optimale.

D'une façon formelle, soit S^* une solution optimale d'une instance d'un problème de minimisation, soit S une solution fournie par l'algorithme approximatif, on dit que l'algorithme est d'approximation de facteur $f > 1$ si pour toute instance de donnée la solution fournie $S \leq f.S^*$.

Un schéma d'approximation est un algorithme qui prend en arguments une instance du problème et un paramètre $\epsilon > 0$, et qui retourne une solution approximative de facteur $(1+\epsilon)$.

Si le temps d'exécution de l'algorithme se fait, en plus, en temps polynomial, on dit que l'algorithme est un PTAS « Polynomial-Time Approximation Scheme ».

Les heuristiques

Ces sont des algorithmes qui permettent de trouver rapidement des solutions approchées mais en revanche, sans garantie sur la solution fournie. La qualité d'une heuristique est estimée par des méthodes purement expérimentales, au contraire d'un algorithme d'approximation dont la qualité est estimée par son facteur d'approximation.

Les algorithmes paramétrés

Ces sont des algorithmes exacts dont l'analyse de temps d'exécution dépend non pas seulement de la taille d'entrée mais en plus d'un paramètre k.

L'objectif des algorithmes paramétrés d'une part est d'étudier les différents paramètres qui mettent en cause qu'un problème est NP-complet. Et d'autre part de trouver un « bon paramètre » pour attaquer le problème. Et la « cerise sur le gâteau » de trouver un algorithme FPT.

Définition 14. *Un problème paramétré est dit* FPT *si on peut le résoudre en temps* $f(k).n^c$ *tel que n est la taille d'instance du problème,* $f(k)$ *dépend seulement de k et c ne dépend pas de k.*

Afin de citer un algorithme FPT, on définit tout d'abord le problème couverture par sommets.

COUVERTURE PAR SOMMETS

entrée:	Un graphe G = (V, E) et un entier $k \leq \|V\|$.
question:	Existe-t-il un ensemble $S \subseteq V$ de taille au plus k tel que pour toute arête $uv \in E$ au moins u ou v est dans S ?

On donne maintenant l'algorithme FPT qui résout le problème de couverture par

sommets.

Algorithme 1 : couverture(G,K)

```
Données : Un graphe G = (V, E) et un entier k ≤ |V|
Résultat : « oui » ou « non »
Sélectionner un sommet v de degré maximum
si deg(v) ≤ 1 alors
    si k ≥ |E| alors
        retourner « oui »
    sinon
        retourner « non »
    fin
fin
si deg(v) ≥ 2 alors
    retourner couverture(G \ {v}, k − 1)
    retourner couverture(G \ N[v], k − |N(v)|)
fin
```

Soit S ⊆ V une couverture par sommets. L'idée de l'algorithme est de brancher sur le sommet v de degré maximum. Si Δ(G) est au plus 1 alors toutes les couvertures par sommets contiennent |E| sommets. Sinon on a deux cas : soit $v \notin S$, soit $v \in S$.

— Le sommet $v \notin S$:
Dans ce cas afin de couvrir tous les arêtes qui relient v, on a $N(v) \subseteq S$ d'où l'appel récursif COUVERTURE$(G \setminus N[v], k - |N(v)|)$.

— Le sommet $v \in S$:
Dans ce cas tous les arêtes qui relient v sont couverts, donc on peut supprimer le sommet v et décrémenter k par 1. D'où l'appel récursif COUVERTURE$(G \setminus \{v\}, k-1)$.

On peut simuler l'exécution de cet algorithme par un arbre d'un hauteur borné par k. Et chaque noeud non feuille a au plus 2 fils. Alors le nombre maximum des noeuds dans cet arbre est 2^{k+1}, et pour chaque noeud on fait un calcul en temps polynomial en fonction de n. D'où le temps d'exécution de l'algorithme est $\mathcal{O}(2^k.poly(n))$. C'est un algorithme FPT puisque $f(k) = 2^k$ dans ce cas et qui dépend seulement de k.

Restriction des instances d'entrée

On restreint les instances d'entrée à certaines classes particulières. Dans les algorithmes de graphes, par exemple, on restreint l'entrée sur des graphes qui ont des bonnes propriétés algorithmiques comme les graphes cordaux, les graphes planaires, les graphes de permutation, les graphes d'intervalles .... Et on exploite dans l'algorithme les différents propriétés du graphe en espérant d'améliorer le temps d'exécution.

Les méthodes précédentes sont basées sur des algorithmes déterministes. On dit qu'un algorithme est déterministe s'il produit toujours le même résultat pour une même entrée.

On définit maintenant un algorithme non déterministe qu'on appelle un algorithme randomisé ou probabiliste.

Les algorithmes randomisés

Un algorithme randomisé, ou probabiliste, est un algorithme non déterministe qui utilise un générateur de nombres aléatoires dans ses instructions afin de trouver rapidement une solution. Cette solution est exacte dans les algorithmes randomisés de type

« Las Vegas » et approchée dans les algorithmes randomisés de type « Monte Carlo ».

Durant cette thèse on va s'intéreser uniquement aux algorithmes exponentiels exacts.

1.5 Les algorithmes exponentiels exacts

Malgré leurs temps exponentiel, les algorithmes exponentiels exacts sont inévitables vu que P = NP est improbable. Industriellement parlant, les solutions exactes de problèmes NP-difficile sont parfois demandées. C'est qui est bien possible pour les instances d'entrée de taille modérée et même préférable dans certaines situations.

> « *For every polynomial time algorithm you have, there is an exponential algorithm that I would rather run.* »
>
> *– Alan Perlis –*

Et du côté théorique, la construction et l'analyse de ces algorithmes permettent de mieux comprendre la complexite de cette classe de problèmes. Ces techniques ont reçu des grands intérêts qui sont couronnés par la publication de l'ouvrage « Exact Exponential Algorithms » par Fedor V. Fomin et Dieter Kratsch [25]. Cet ouvrage de référence a consacré un chapitre entier pour chaque paradigme des algorithmes exponentiels exacts :

— Programmation dynamique.

— Brancher et réduire.

— Mesurer pour Conquérir.

— Inclusion-exclusion.

— etc . . .

On va détailler par la suite les deux paradigmes « Brancher et réduire » et « Mesurer pour Conquérir » qui sont utilisés dans cette thèse.

1.5.1 Brancher et réduire

« Brancher et réduire » est une technique principale et fondamentale pour la conception des algorithmes exponentiels exacts. Elle se cache souvent dans les publications sous différents noms comme « branching », « DPLL », « search tree », « backtracking ». . .

Cette popularité est due à sa facilité d'implémentation et l'espace mémoire requis pour son exécution qui est seulement polynomial.

Les algorithmes « brancher et réduire » sont des algorithmes récursifs constitués par deux types d'instructions qu'on appelle « règle de réduction » et « règle de branchement ».

Règle de réduction

Une régle de réduction est utilisée pour simplifier l'instance du problème en une autre, ou pour arrêter l'exécution de l'algorithme, et/ou pour afficher une solution.

Règle de branchement

Une règle de branchement permet de résoudre une instance du problème par des appels récursifs à deux ou plusieurs instances de plus petites tailles. Typiquement les règles de branchement et réduction ne nécessitent qu'un temps polynomial.

Grâce à cette récursivité, les algorithmes « brancher et réduire » sont faciles à les prouver par récurrence. Mais ce n'est pas toujours le cas pour l'analyse du temps d'exécution.

1.5.2 Arbre de recherche et Temps d'exécution

On peut simuler l'exécution d'un algorithme « brancher et réduire » par un arbre de recherche enraciné où la racine correspond à l'instance d'entrée initiale du problème. Et chaque fils d'un noeud correspond à un appel récursif par une règle de branchement de l'instance de ce noeud. Noter bien, qu'on n'assigne pas un fils à un noeud si on applique une régle de réduction à l'instance correspond à ce noeud.

Le temps d'exécution d'un algorithme « brancher et réduire » est calculé à partir du nombre de noeuds de son arbre de recherche. Mais comme le nombre de noeuds d'un arbre de recherche avec l feuilles est au plus $2l-1$ alors il suffit de borner le nombre maximum de feuilles. Afin de compter ce dernier nombre, on associe une mesure $\mu(I)$ à chaque intance I du problème.

Pour chaque règle de branchement b de l'instance $\mu(I)$ qui fait appel à $r \geq 2$ instances $I_1, I_2, \ldots, I_r$ de tailles respectivement $\mu(I) - t_1, \mu(I) - t_2, \ldots, \mu(I) - t_r$ tel que $\mu(I_i) \leq \mu(I) - t_i$ pour toute $i \in \{1..r\}$, on appelle $b = (t_1, t_2, \ldots, t_r)$ un vecteur de branchement de cette règle.

Soit $T(s)$ le nombre maximum de feuilles d'un arbre de recherche associé à une instance du problème I de taille $\mu(I) = s$ où on applique une unique règle de branchement b.

On a :

$$T(s) \leq T(s-t_1) + T(s-t_2) + \ldots + T(s-t_r) \tag{1.1}$$

Il suffit de résoudre l'équation :

$$T(s) = T(s-t_1) + T(s-t_2) + \ldots + T(s-t_r) \tag{1.2}$$

Comme la solution est de la forme α^s alors :

$$x^s = x^{s-t_1} + x^{s-t_2} + \ldots + x^{s-t_r} \tag{1.3}$$

On appelle l'unique solution réelle positive α de l'équation, le facteur de branchement de la règle $b = (t_1, t_2, \ldots, t_r)$ et on le désigne par $\tau(t_1, t_2, \ldots, t_r)$. Dans la théorie il faut résoudre le polynôme par l'une de méthodes de l'analyse numérique comme la méthode de Newton-Raphson. Mais dans la pratique, ce n'est pas difficile de calculer $\tau(t_1, t_2, \ldots, t_r)$ si on utilise un logiciel comme Matlab ou un site comme WolframAlpha.

Dans un algorithme « brancher et réduire » il existe en général plusieurs règles de branchement, dans ce cas on prend le pire cas, c'est-à-dire le facteur de branchement α le plus grand. Et on déduit que le temps d'exécution de l'algorithme est $\mathcal{O}^*(\alpha^s)$.

Remarques :

— Si $\mu(I) = s < n$, on peut déduire directement que le temps d'exécution de l'algorithme est $\mathcal{O}^*(\alpha^n)$.

— Si α est un réel arrondi, alors on peut éliminer le polynôme et déduire que le temps d'exécution est $\mathcal{O}(\alpha^n)$.

1.5.3 Exemple

On donne dans cette section un algorithme de type « brancher et réduire » pour résoudre le problème du stable maximum.

Algorithme 2 : mis(G)

Données : Un graphe G = (V, E)
Résultat : La taille du stable maximum
si $\exists x \in V : deg(x) = 0$ **alors**
  retourner 1 + mis(G − x)
fin
si $\exists x \in V : deg(x) = 1$ **alors**
  retourner 1 + mis(G − N[x])
fin
si $\Delta(G) \geq 3$ **alors**
  Sélectionner un sommet v de degré maximum
  retourner max {1 + mis(G − N[v]), mis(G − v) }
fin
si $\Delta(G) \leq 2$ **alors**
  retourner |V| ÷ 2
fin

L'idée de l'algorithme est de traiter les sommets de degré inférieur à 1 avec des règles de réduction. Puis de sélectionner un sommet v de degré maximum. Si $\Delta(G) \geq 3$ alors on branche :

— Soit v dans le stable de taille maximum : Dans ce cas, on incremente la taille du stable par 1 et on fait un appel récursif à MIS(G − N[v]).

— Soit v n'appartient pas au stable de taille maximum : Dans ce cas, on fait un appel récursif à MIS(G − v).

Sinon, puisque les deux premières règles de réduction ne sont pas appliquées alors il n'existe aucun sommet de degré inférieur ou égal à 1. Et comme $\Delta(G) = 2$ alors tous les sommets sont de degré 2 et G est un C_n. Alors il suffit de retourner $n \div 2$. Tous les règles de cet algorithme sont des règles de réduction sauf la troisième lorsque $\Delta(G) \geq 3$ qui est une régle de branchement. Dans ce cas on a un vecteur de branchement $(\Delta(G)+1, 1)$ et au pire cas $\Delta(G) = 3$ qui correspond à un vecteur de branchement $(4, 1)$. Son facteur de branchement est inférieur strictement à 1.3803. Donc le temps d'éxécution de l'algorithme est $\mathcal{O}(1.3803)$.

1.5.4 Mesurer pour Conquérir

Les algorithmes « Mesurer pour Conquérir » sont similaires dans leurs constructions aux algorithmes « Brancher et réduire ». Mais la différence est au niveau de l'analyse du temps d'exécution. En général dans un algorithme « Brancher et réduire », en associant à chaque sommet un poids de « 0 » ou « 1 », μ(I) est un entier. Alors que dans un algorithme « mesurer pour conquérir », on associe à chaque sommet un réel postif comme un poids, et dans ce cas $\mu(I) \in \mathbb{R}+$. Typiquement la distinction des poids se base sur les degrés de sommets.

Analysons maintenant plus soigneusement l'algorithme précédent. La suppression d'un sommet par une régle de branchement peut générer des sommets de degré inférieur strictement à 2. Ces sommets seront traités immédiatement par les deux premières règles

de réduction mais jamais comptés dans l'analyse du temps d'exécution. D'où la limite de l'analyse standard.

En plus, les sommets de degré inférieur ou égal à 2 sont traitables facilement par des simples règles de réduction alors que les sommets de degré 3 ou plus ne sont traitables que par des règles de branchement. Et vu que l'arbre de recherche se construit uniquement par les règles de branchement, alors on peut dire que les sommets de degré 3 ou plus sont plus coûteux que les autres sommets.

De même les sommets de degré 2, qui sont aussi faciles à traiter par des règles de réduction, sont différents aux sommets de degré 0 ou 1. Ces derniers sont traitables immédiatement dès leur apparition alors que certains sommets de degré 2 persistent dans le graphe jusqu'à la fin. D'où l'idée de donner des poids différents aux sommets en fonction de leur degré.

Soit n_2, n_3 et $n_{\geq 4}$ le nombre de sommets de degré respectivement 2, 3 et au moins 4.

Soit w_2, w_3 et w_4 des poids qui appartiennent à [0..1].

Soit d_2, d_3, d_4 et $d_{\geq 5}$ le nombre de voisins du sommet v de degré respectivement 2, 3, 4 et au moins 5.

Et on définit alors la mesure suivante :

$$\mu(G) = w_2 n_2 + w_3 n_3 + w_4 n_{\geq 4} \tag{1.4}$$

Maintenant regardons la règle de branchement lorsque le degré de v est égal à 3.

— Soit v dans le stable de taille maximum : Dans ce cas, on supprime N[v] et la mesure diminue par $w_3 + w_2 d_2 + w_3 d_3$.

— Soit v n'appartient pas au stable de taille maximum : Dans ce cas, on supprime le sommet v et la mesure diminue par un w_3. En plus les adjacents de v de degré 2 deviennent de degré 1 et seront supprimés par la deuxième règle de réduction. Alors la mesure diminue aussi par $w_2 d_2$. Par contre les adjacents de degré 3 se baissent à degré 2 et la mesure diminue par $d_3(w_3 - w_2)$.

D'où l'équation :

$$\begin{aligned} T(\mu) \leq T(\mu - w_3 - w_2 d_2 - w_3 d_3) + T(\mu - w_3 - w_2 d_2 - d_3(w_3 - w_2)), \\ 0 \leq d_2, d_3 \leq 3 \text{ et } d_2 + d_3 = 3 \end{aligned} \tag{1.5}$$

De même si le sommet v est de degré 4 alors :

$$\begin{aligned} T(\mu) \leq T(\mu - w_4 - w_2 d_2 - w_3 d_3 - w_4 d_4) + T(\mu - w_4 - w_2 d_2 - d_3(w_3 - w_2) - d_4(w_4 - w_3)), \\ 0 \leq d_2, d_3, d_4 \leq 4 \text{ et } d_2 + d_3 + d_4 = 4 \end{aligned} \tag{1.6}$$

Si le sommet v est de degré au moins 5 alors :

$$\begin{aligned} T(\mu) \leq & T(\mu - w_4 - w_2 d_2 - w_3 d_3 - w_4 d_4 - w_4 d_{\geq 5}) \\ & + T(\mu - w_4 - w_2 d_2 - d_3(w_3 - w_2) - d_4(w_4 - w_3)), \\ 0 \leq & d_2, d_3, d_4, d_{\geq 5} \leq deg(v) \text{ et } d_2 + d_3 + d_4 + d_{\geq 5} = deg(v) \end{aligned} \tag{1.7}$$

Calculons enfin les poids avec la méthode de Newton ou la descente de gradient, on trouve que les valeurs optimales de w_2, w_3 et w_4 sont respectivement 0.596602, 0.9228645 et 1. Alors le temps d'exécution de l'algorithme est $\mathcal{O}(1.2905^n)$ au lieu de $\mathcal{O}(1.3803^n)$. La puissance de « Mesurer pour Conquérir », comme l'illustre cet exemple, est la capacité d'améliorer l'analyse du temps d'exécution d'un algorithme sans la nécessité de le modifier.

1.6 Enumération

Un problème d'énumération consiste à énumérer explicitement tous les solutions. Récemment l'énumération a suscité un grand intérêt et un établissement de plusieurs nouveaux résultats. Cet intérêt a touché plusieurs domaines de l'informatique comme la bio-informatique [29] où l'énumération des protéines satisfaisant certains phénotypes permet d'identifier des nouveaux gènes ainsi que leur caractéristiques, où même révéler des nouvelles fonctionnalités d'un gène déjà connu. Ceci est devenu de plus en plus disponible et popularisé grâce aux bases de données biologiques accessibles en ligne. L'énumération a suscité un intérêt aussi dans d'autres domaines de l'informatique comme l'intelligence artificielle, l'exploration de données, les bases de données...

L'énumération est indispensable. Parfois, l'algorithme d'optimisation ou de dénombrement le plus rapide connu est celui d'énumération. En outre, parfois une seule solution est insuffisante et la solution optimale n'a pas de sens dans un domaine comme la bio-informatique où le mécanisme est complexe et plusieurs facteurs entrant en jeu sont non connus en avance.

Il existe deux approches d'énumération : l'une est appelée « output-sensitive » et l'autre « input-sensitive ».

1.6.1 Output-sensitive

La mesure du temps d'exécution des algorithmes d'énumération de type « Output-sensitive » se base sur la taille d'entrée, mais surtout sur la taille de sortie, c'est-à-dire le temps d'exécution nécessaire pour produire deux solutions consécutives, appelé délai, ainsi que le temps nécessaire pour trouver la première solution.

L'objectif des chercheurs travaillant sur l'énumération « output-sensitive » est de concevoir des algorithmes en délai polynomial où en délai incrémental.

Un algorithme d'énumération est dit en délai polynomial si la production de deux solutions consécutives se fait en temps polynomial par rapport à la taille d'entrée du problème.

Un algorithme est dit en délai incrémental si le délai est polynomial par rapport à la taille d'entrée et le nombre des solutions déjà produites. D'une façon formelle le délai entre le i-ème solution et $(i+1)$-solution est borné par un $poly(i+n)$ où n est la taille d'entrée.

Il est clair que les algorithmes en délai polynomial sont considérés plus efficaces que les algorithmes en délai incrémental vu l'inclusion de deux classes des problèmes (SCHMIDT [60]) et vu que le délai incrémental peut augmenter après chaque solution produite.

1.6.2 Input-sensitive

Contrairement aux algorithmes « output-sensitive », l'analyse du temps d'exécution des algorithmes d'énumération « input-sensitive » se base uniquement à la taille d'entrée. Les techniques les plus utilisées dans cette approche sont principalement « Brancher et réduire » et « Mesurer pour Conquérir ». L'objectif n'est pas seulement d'énumérer des objets mais aussi de donner une « borne supérieure » pour le nombre maximum de ces objets. Le temps d'exécution d'un algorithme d'énumération « input-sensitive » donne systématiquement cette borne. Et la question qui reste à poser si on peut concevoir un algorithme plus rapide, autrement dit, si la « borne supérieure » est bien stricte. Pour cela,

la trouvaille d'une famille de graphes particulière, appelée « borne inférieure », avec le nombre le plus élevé possible de ces objets énumérés est nécessaire pour s'assurer que l'algorithme est optimal et que la « borne supérieure » est bien stricte.

Parmi les travaux les plus connus dans cette direction, il y a celles de MOON et MOSER [55] qui ont montré que le nombre maximum des ensembles stables maximaux dans un graphe de n sommets est $3^{n/3}$.

Dans la section suivante, on détaille ces résultats.

1.7 Enumération des stables maximaux

Dans cette section, on cite ENUMMIS, l'algorithme 3, qui énumère tous les ensembles stables maximaux dans un graphe G de n sommets. L'algorithme ENUMMIS prend comme arguments un graphe et une solution potentielle S d'un ensemble stable maximum initialement vide. Alors, il suffit d'appeler ENUMMIS(G, $\emptyset$).

Algorithme 3 : EnumMIS(G,S)

Données : Un graphe G = (V, E), et un ensemble S
Résultat : Tous les ensembles stables maximaux dans G
si V(G) = $\emptyset$ **alors**
  retourner S
fin
Sélectionner un sommet v de degré minimum
EnumMIS(G − N[v], S ∪ {v})
pour chaque $x \in N(v)$ **faire**
  EnumMIS(G − N[x], S ∪ {x})
fin

L'idée de l'algorithme ressemble beaucoup à sa version d'optimisation (l'algorithme 2 : MIS(G)) et la seule différence est l'ajout d'une boucle pour lister tous les ensembles stables maximaux. Du coup, il suffit d'analyser le temps d'exécution de l'algorithme sans donner une preuve de correction. L'algorithme ENUMMIS est un algorithme de type « Brancher et réduire » qui contient une unique règle de branchement. Le sommet v est de degré minimum alors on a pour tout $x \in N(v)$, $|N[x]| \geq |N[v]| = \delta(G) + 1$. Donc le vecteur de branchement dans ce cas est $(\underbrace{\delta(G)+1, \ldots, \delta(G)+1}_{\delta(G)+1})$ et son facteur atteint le maximum lorsque $\delta(G) = 2$. Comme le facteur du vecteur de branchement $(3,3,3)$ est égal à $3^{1/3}$, alors le temps d'exécution de l'algorithme est $\mathcal{O}^*(3^{n/3})$. Cet algorithme permet d'établir une borne supérieure pour le nombre maximum des ensembles stables maximaux.

MOON et MOSER ont prouvé autrement les mêmes résultats par une méthode purement combinatoire, c'est-à-dire sans la conception d'un algorithme d'énumération.

Théorème 2 ([55]). *Le nombre maximum des ensembles stables maximaux dans un graphe de n sommets est au plus* $3^{n/3}$.

Cette borne supérieure est stricte, vu l'existence d'une borne inférieure : une famille de graphes, composée par l'union des triangles, qui a exactement $3^{n/3}$ ensembles stables maximaux (cette borne est décrite dans l'introduction de la section 1.2).

1.8 Problèmes étudiés

Au contraire de l'énumération des ensembles stables maximaux, avoir deux bornes qui se confondent n'est pas évident du tout. Et c'est assez courant dans l'énumération « input-sensitive » d'avoir un grand écart. Ce problème concerne même les ensembles les plus classiques et les plus étudiés comme les ensembles dominants minimaux où le meilleur algorithme connu pour énumérer ces ensembles est en $\mathcal{O}(1.7159^n)$ [24] et la meilleure borne inférieure, établie par Dieter Kratsch, connue est seulement 1.5704^n [24].

Durant cette thèse, on propose dans le chapitre 2, un nouvel algorithme pour énumérer tous les ensembles dominants minimaux dans les graphes cordaux en $\mathcal{O}(1.5048^n)$. On a étudié aussi l'énumération des ensembles dominants connexes minimaux et les ensembles irredondants maximaux qui sont très proches aux ensembles dominants minimaux. Dans le chapitre 4, on propose un algorithme d'énumération des ensembles dominants connexes minimaux dans les graphes bipartis convexes en $\mathcal{O}(1.7254^n)$. D'une part c'est déjà un progrès par rapport à la borne précédente, celle d'un graphe général, qui est en $\mathcal{O}(2^{(1-\epsilon)n})$ où $\epsilon > 10^{-50}$ [54]. D'autre part, c'est le premier algorithme d'énumération « input-sensitive » établi pour une sous-classe non triviale de graphes bipartis.

On a conçu, dans le chapitre 3, des algorithmes d'énumération des ensembles irredondants maximaux dans les graphes cordaux, les graphes d'intervalles et les forêts en $\mathcal{O}(1.7549^n)$, $\mathcal{O}(1.6957^n)$ et $\mathcal{O}(1.6181^n)$ respectivement au lieu de l'algorithme trivial en $\mathcal{O}^*(2^n)$. On a proposé aussi comme une borne inférieure un graphe de 1.5292^n ensembles irredondants maximaux.

Finalement, afin de varier, on a étudié, dans le chapitre 4, un nouvel ensemble défini récemment : L'ensemble tropical connexe minimal. On a proposé une borne inférieure de 1.4961^n, mais sans réussir à améliorer la borne supérieure de 2^n. On a proposé des algorithmes d'énumération des ensembles tropicaux connexes minimaux dans les graphes cobipartis, les graphes d'intervalles, et les graphes blocs en $\mathcal{O}^*(3^{n/3})$, $\mathcal{O}(1.8613^n)$ et $\mathcal{O}^*(3^{n/3})$ respectivement. On a établi une borne inférieure de 1.4766^n pour les graphes scindés et de $3^{n/3}$ pour les graphes cobipartis, les graphes d'intervalles et les graphes blocs.

Chapitre 2

Enumération des ensembles dominants minimaux

Sommaire

2.1 Etat de l'art

Un ensemble de sommets $D \subseteq V$ est un ensemble dominant de G si pour tout $v \in V$, $v \in N[D]$.

Le problème de dominance, c'est à dire l'ensemble dominant de taille minimum, est un problème connu « NP-complet » dépuis longtemps. Ce n'est pas seulement l'un des principaux problèmes théoriques et fondamentaux mais aussi un problème qui a des nombreuses applications industrielles dans la pratique [39, 40]. Le meilleur algorithme qui le résout est de temps d'execution $\mathcal{O}(1.4689^n)$ [43].

On dit qu'un ensemble dominant D est minimal s'il n'existe aucun autre ensemble dominant $D' \subset V$ tel que $D' \subset D$. Un ensemble dominant D est minimal si pour tout sommet $v \in D$, v a un privé.

L'énumération des ensembles dominants minimaux dans un graphe général se fait en $\mathcal{O}(1.7159^n)$ [24]. C'est qui implique que le nombre maximum des ensembles dominants minimaux dans un graphe de n sommets est au plus 1.7159^n. La meilleure borne inférieure connue est 1.5704^n [24].

Concernant les classes de graphes, la plupart des classes ont deux bornes strictes. Par exemple, les graphes scindés [18], les graphes d'intervalles [18], les trivialement parfaits [17] ont deux bornes strictes de $3^{n/3}$.

Les cographes, les seuils et les chaines ont deux bornes strictes de $15^{n/6}$, $w(G)$ et $\lfloor n/2 \rfloor + m$ respectivement [17].

Le problème d'énumération des ensembles dominants minimaux dans les arbres est resté non résolu longtemps. Au début, KRZYWKOWSKI [52] a proposé un algorithme en $\mathcal{O}(1.4656^n)$. Puis GOLOVACH et collab. [31], ont amélioré l'algorithme en $\mathcal{O}(1.4422^n)$. Récemment et finalement, ROTE [59] a réussi à prouver que 1.4195^n est la borne stricte pour les arbres.

Pour les graphes cordaux, COUTURIER et collab. [17] ont prouvé que le nombre maximum des ensembles dominants minimaux est au plus 1.6181^n. Puis ABU-KHZAM et HEGGERNES [1] ont amélioré cette borne à 1.5214^n. Mais la meilleure borne inférieure connue est 1.4422^n [17].

Concernant les algorithmes « output-sensitive », GOLOVACH et collab. [30] ont montré que tous les ensembles dominants minimaux d'un graphe biparti cordal peuvent être générés en délai incrémental polynomial.

De plus, il existe des algorithmes d'énumération en délai polynomial pour énumérer tous les ensembles dominants minimaux dans les graphes cordaux [48] et les graphes sans triangles [7].

De même, il existe des algorithmes en délai linéaire pour les graphes scindés et les graphes sans P_6 cordaux [46], ainsi pour les graphes d'intervalles et les graphes de permutation après un prétraitement en temps polynomial [47].

2.1.1 Les ensembles dominants minimaux dans les graphes quelconques

Borne Inférieure

La borne inférieure est construite par l'union des octaèdres. Chaque octaèdre contient 15 ensembles dominants minimaux. Par exemple dans l'octaèdre illustré à la figure 2.1, on a :

— Soit les deux sommets 1 et 2 à la fois dans l'ensemble. Dans ce cas, on a l'unique ensemble dominant minimal $\{1,2\}$.

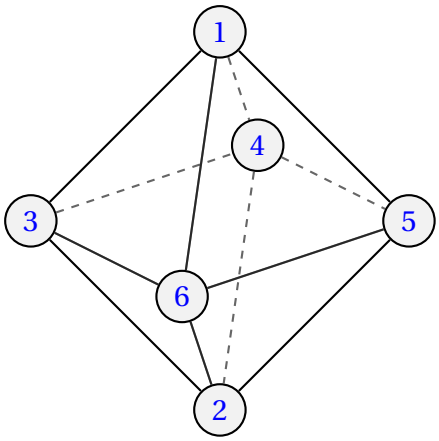

FIGURE 2.1 – La borne inférieure des ensembles dominants minimaux dans un graphe quelconque

— Soit on a l'un de deux. Et dans ce cas, il faut sélectionner l'un des quatre sommets 3,4,5 et 6. Dans ce cas, on a 8 ensembles dominants minimaux.

— Soit ni le sommet 1, ni le sommet 2 dans l'ensemble et il faut sélectionner 2 parmi les quatre sommets 3,4,5 et 6. Dans ce cas, on $C_4^2 = 6$ ensembles dominants minimaux.

La borne est composée par $n/6$ octaèdres. Dans chaque octaèdre il y a 15 ensembles dominants minimaux. Donc on a $15^{n/6} \simeq 1.5704^n$ ensembles dominants minimaux.

Algorithme de FOMIN, GRANDONI, PYATKIN et STEPANOV

L'idée de l'algorithme est de réduire dans un graphe G le problème d'énumération des ensembles dominants minimaux DOM(G) à un problème d'énumération des ensembles minimaux de couverture par ensembles COV(U,S) où U est un ensemble appelé l'univers et S une famille des sous-ensembles de U. La technique utilisée pour résoudre le problème d'énumération des ensembles minimaux de couverture par ensembles est « Mesurer pour Conquérir ».

Les règles de branchement et de réduction s'appliquent principalemet selon la fréquence des éléments de l'univers dans S d'une part et d'autre part sur le nombre des éléments d'un sous ensemble dans S.

Les auteurs ont donné alors à chaque ensemble $s \in S$ un poids $\alpha_{|s|}$ et à chaque $u \in U$ un poids $\beta_{|u|}$ où $|u|$ dénote le nombre d'apparition de u dans S.

Puis ils ont défini la mesure suivante :

$$k(U,S) = \sum_{s \in S} \alpha_{|s|} + \sum_{u \in U} \beta_{|u|} \tag{2.1}$$

Tel que :

— $1 \leq \alpha_1 < \alpha_2 < \alpha_3 < \alpha_4 = \alpha_i$, pour tout $i \geq 5$.

— $0 = \beta_1 < \beta_2 < \beta_3 < \beta_4 < \beta_5 = 1 = \beta_j$, pour tout $j \geq 6$.

Résolvant ce problème d'optimisation, les auteurs ont obtenu le théorème suivant :

Théorème 3 ([24]). *Une instance de couverture par ensembles* (U,S) *a au plus* $\lambda^{|U|+\alpha.|S|}$ *ensembles minimaux de couverture par ensembles et qui peuvent être énumérés en* $\mathcal{O}^*(\lambda^{|U|+\alpha.|S|})$, *où* $\lambda = 1.156154$ *et* $\alpha = 2.720886$.

Afin d'énumérer tous les ensembles dominants minimaux, il suffit d'énumérer tous les ensembles minimaux de couverture par ensembles de l'instance (U,S) où l'univers U

est l'ensemble de sommets du graphe et la famille des sous-ensembles $S = \{\cup_{u \in U} N[u]\}$. Comme $|U| = |S| = n$, alors un graphe de n sommets a au plus $\lambda^{(1+\alpha).n}$ ensembles dominants minimaux.

Théorème 4 ([24])**.** *Un graphe de n sommets a au plus* 1.7159^n *ensembles dominants minimaux qu'on peut énumérer en* $\mathcal{O}(1.7159^n)$.

2.1.2 Les ensembles dominants minimaux dans les graphes cordaux

Le problème de dominance est « NP-complet » dans les graphes cordaux [8]. Le premier algorithme d'énumération des ensembles dominants minimaux dans les graphes cordaux est proposé par COUTURIER, HEGGERNES, VAN 'T HOF et KRATSCH [17] et qui s'exécute en $\mathcal{O}(1.6181^n)$. Ils ont proposé dans le même article une borne inférieure de 1.4422^n.

Borne inférieure

La borne est composée par $n/3$ triangles (C_3). Dans chaque triangle il y a 3 ensembles dominants minimaux. Donc on a $3^{n/3} \simeq 1.4422^n$ ensembles dominants minimaux.

Algorithme de COUTURIER, HEGGERNES, VAN 'T HOF et KRATSCH

Algorithme 4 : EnumMDS(G,S)

Données : Un graphe G = (V, E), un ensemble S
Résultat : Tous les ensembles dominants minimaux
si *Il existe un sommet isolé v* **alors**
  si *v est dominé* **alors**
    EnumMDS(G \ {v}, S)
  sinon
    EnumMDS(G \ {v}, S ∪ {v})
  fin
fin
Sélectionner un sommet simplicial x
si *x est dominé* **alors**
  EnumMDS(G \ N[x], S ∪ {x})
  EnumMDS(G \ {x}, S)
fin
si *x n'est pas dominé* **alors**
  Soit un sommet $y \in N(x)$
  EnumMDS(G \ {x, y}, S ∪ {y})
  EnumMDS(G \ {y}, S)
fin

Soit D un ensemble dominant minimal et soit l'ensemble $S \subseteq D$. Chaque graphe cordal non complet possède au moins deux simpliciaux non adjacents (DIRAC [21]). D'où la possibilité de brancher sur un simplicial x. Si le simplicial x est dominé alors il y a deux possiblités :

— Le simplicial $x \in D$: Dans ce cas $N(x) \cap D = \emptyset$ sinon x ne peut pas avoir un voisin privé et D ne satisfait pas la condition de minimalité. D'où l'appel récursif de ENUMMDS(G \ N[x], S ∪ {x}).

— Le simplicial $x \notin D$: Et comme x est dominé et il ne peut pas être un voisin privé d'un autre sommet alors c'est possible de supprimer x du graphe. D'où l'appel récursif EnumMDS$(G \setminus \{x\}, S)$.

Sinon le simplicial x n'est pas dominé. Mais comme le sommet x a au moins un adjacent, sinon dans ce cas le sommet x est isolé et la règle de réduction est applicable, alors on branche sur son adjacent y :

— Le sommet $y \in D$: Dans ce cas le simplicial x ne peut pas avoir un privé et donc il ne peut pas être dans l'ensemble D. D'où l'appel récursif EnumMDS$(G \setminus \{x, y\}, S \cup \{y\})$.

— Le sommet $y \notin D$. Comme le simplicial x n'est pas dominé alors n'importe quel sommet appartient à $N[x] \setminus \{y\}$ et qui va dominer x dans les prochaines étapes de l'algorithme, va dominer aussi à l'occasion le sommet y. D'où la possibilité de supprimer y du graphe et d'appeler récursivement EnumMDS$(G \setminus \{y\}, S)$.

Les deux règles de branchement de cet algorithme de type « Brancher et réduire » sont d'un vecteur de branchement $(2,1)$ qui a un facteur arrondi de 1.6181. D'où le théorème suivant :

Théorème 5 ([17])**.** *Un graphe cordal de n sommets a au plus* 1.6181^n *ensembles dominants minimaux qu'on peut énumérer en* $\mathcal{O}(1.6181^n)$.

Algorithme de Abu-Khzam et Heggernes

Abu-Khzam et Heggernes [1] ont proposé aussi un algorithme de branchement en s'appuyant non pas seulement sur le fait que chaque graphe cordal a un sommet simplicial mais sur le développement d'une propriété plus puissante :

Théorème 6 ([1])**.** *Soit* G *un graphe cordal non trivial, alors il existe une paire de sommets u et v satisfaisant au moins l'une de ces trois assertions suivantes :*

1. *Les deux sommets u et v sont deux simpliciaux tel que* $N(u) \cap N(v) \neq \emptyset$.
2. *Les deux sommets u et v sont deux simpliciaux jumeaux.*
3. *Le sommet u est simplicial, v son adjacent semi-simplicial et le degré de v dans* G *est au moins* 2.

Abu-Khzam et Heggernes [1] sont parvenus à prouver en s'appuyant sur cette propriété le théorème suivant :

Théorème 7 ([1])**.** *Un graphe cordal de n sommets a au plus* 1.5214^n *ensembles dominants minimaux qu'on peut énumérer en* $\mathcal{O}(1.5214^n)$.

On propose dans la section suivante un algorithme « Mesurer pour Conquérir » d'énumération des ensembles dominants minimaux en $\mathcal{O}(1.5048^n)$ qui s'appuie également sur cette nouvelle propriété sur les sommets simpliciaux d'un graphe cordal décrite dans le théorème 6.

2.2 Notre contribution

Le lemme suivant se déduit immédiatement du théorème 6.

Lemme 1. *Soit* G *un graphe cordal non trivial et connexe alors il existe un sommet simplicial* $x \in V(G)$ *qui a un voisin* $y \in N_G(x)$ *tel que :*

— *Le sommet y est un simplicial dans* $G - x$.

— *Sinon, il existe un sommet simplicial* $z \neq x$ *dans* G *tel que* $xz \notin E(G)$ *et* $y \in N_G(x) \cap N_G(z)$.

En basant sur le lemme 1, on construit un algorithme d'énumération avec sa preuve de correction dans les sections 2.2.1 et 2.2.2, et dans la section 2.2.3 on évalue son temps d'exécution afin d'obtenir une borne supérieure des ensembles dominants minimaux dans les graphes cordaux.

2.2.1 L'algorithme d'énumération

Soit G un graphe cordal. Soit EnumMDS(S,X,F) un algorithme récursif de type « Brancher et réduire », où $S, X, F \subseteq V(G)$ sont des sous-ensembles disjoints. L'algorithme EnumMDS(S,X,F) énumère tous les ensembles dominants minimaux D dans G tels que $S \subseteq D$ et $D \setminus S \subseteq X$.

Soit $H = G[X \cup F]$ un graphe induit de G par $X \cup F$ tel que $F \cap D = \emptyset$. On dit qu'un sommet est *interdit* et il appartient à F s'il ne peut pas être inclus dans un ensemble dominant minimal D. Par contre, on dit qu'un sommet est *libre* s'il appartient à X. On dit aussi qu'un sommet de H est *dominé* s'il est dominé dans G par un sommet de S.

À chaque étape de l'algorithme, on réduit l'instance par une règle de réduction ou on branche. On suppose toujours qu'une étape de l'algorithme est appliquée si aucune étape précédente n'est applicable.

On dit qu'un ensemble dominant minimal D est *compatible* avec le triplet (S,X,F) si les conditions suivantes sont satisfaisantes :

(i) $S \subseteq D$,

(ii) $D \setminus S \subseteq X$,

(iii) S domine $V(G) \setminus (X \cup F)$,

(iv) pour tout $v \in D \setminus S$, v a un privé dans $F \cup X$.

L'idée principale de la preuve de correction de EnumMDS(S,X,F) est de prouver que l'algorithme prendre en compte chaque ensemble dominant minimal compatible avec (S,X,F) et il le produit.

Pour le rendre plus lisible, on a séparé l'algorithme en trois phases : Dans la première phase, soit on réduit l'instance d'entrée par des règles de réduction sans brancher, soit on arrête l'algorithme. Dans la deuxième phase, on branche sur des sommets simpliciaux et leurs voisins. Dans la troisième phase, on branche sur les sommets pendants du graphe H.

On a illustré, au travers des figures explicatives, quelques étapes de l'algorithme en représentant, si possible de le déterminer, un sommet dominé en bleu et un sommet interdit en rouge.

Phase 1. Prétraitement.

1. Si $X = \emptyset$, alors vérifiez si S est un ensemble dominant minimal dans G et retournez le si le cas ; arrêtez.
2. Si H a un sommet interdit et dominé x, alors appelez EnumMDS$(S, X, F \setminus \{x\})$.
3. S'il existe $x \in X$ tel que tous les sommets $N_H[x]$ sont dominés, alors appelez EnumMDS$(S, X \setminus \{x\}, F)$.
4. S'il existe $x \in F$ tel que tous les sommets $N_H[x]$ sont interdits, alors rejetez S et arrêtez.

5. S'il existe un sommet $x \in F$ non dominé tel que x a un seul voisin libre y, alors appelez ENUMMDS$(S \cup \{y\}, X \setminus \{y\}, F \setminus \{x\})$.
6. S'il existe un sommet $x \in X$ non dominé tel que tous les sommets $N_H(x)$ sont interdits, alors appelez ENUMMDS$(S \cup \{x\}, X \setminus \{x\}, F)$.
7. S'il existe un sommet simplicial non dominé $x \in X$ dans H avec $d_H(x) \geq 2$ tel qu'il existe un sommet $y \in N_H(x) \cap F$, alors appelez ENUMMDS$(S, X, F \setminus \{y\})$.

La preuve de correction de la phase 1 est basée sur le lemme suivant.

Lemme 2. *Soit* D *un ensemble dominant minimal du graphe* G *compatible avec un triplet* (S, X, F) *des sous-ensembles disjoints de* V(G). *Si l'une des conditions des étapes 1–7 de la phase 1 est satisfaite, alors soit* $D = S$ *et l'algorithme retourne* D*, soit l'algorithme ne s'arrête pas et il y a un appel récursif de* ENUMMDS(S', X', F') *dans l'appel de* ENUMMDS(S, X, F) *tel que* D *est compatible avec* (S', X', F').

Démonstration.

On considère 7 cas correspondant aux étapes 1–7 de la phase 1 de l'algorithme.

Cas 1. L'ensemble $X = \emptyset$.

On considère la première étape de la phase 1 qui peut être appliquée. Si $X = \emptyset$, alors $S = D$, car $D \setminus S \subseteq X$, et l'algorithme retourne D.

Cas 2. Il existe un sommet x interdit et dominé dans H.

Si H a un sommet interdit et dominé x, alors $x \notin D$. De même, x ne peut pas être un privé pour aucun $v \in D \cap X$, car x est dominé par un sommet de S. Par conséquent, D est compatible avec $(S, X, F \setminus \{x\})$.

Cas 3. Il existe $x \in X$ tel que tous les sommets $N_H[x]$ sont dominés.

S'il existe $x \in X$ tel que tous les sommets $N_H[x]$ sont dominés, alors $x \notin D$, car x ne peut pas avoir un privé dans H. De même, x ne peut pas être un privé pour aucun $v \in D \cap X$, car x est dominé par un sommet de S. Par conséquent, D est compatible avec $(S, X \setminus \{x\}, F)$.

Cas 4. Il existe un sommet $x \in F$ tel que tous les sommets $N_H[x]$ sont interdits.

On suppose qu'il existe un sommet $x \in F$ tel que tous les sommets $N_H[x]$ sont interdits. Alors $x \notin D$ comme x est interdit et il n'est pas dominé car l'étape 2 n'est pas appliquée. Contradiction; car D est un ensemble dominant minimal compatible avec (S,F,X). Par conséquent, on rejete S.

Cas 5. Il existe un sommet $x \in F$ non dominé tel que x a un seul voisin libre y.

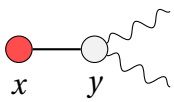

FIGURE 2.2 – Cas 5 de la phase 1

S'il existe un sommet $x \in F$ non dominé tel que x a un seul voisin libre y, alors $y \in D$, car D est un ensemble dominant. De plus, x est un privé de y. Par conséquent, D est compatible avec $(S \cup \{y\}, X \setminus \{y\}, F \setminus \{x\})$.

Cas 6. Il existe un sommet $x \in X$ non dominé tel que tous les sommets $N_H(x)$ sont interdits.

On suppose qu'il existe un sommet $x \in X$ non dominé tel que tous les sommets $N_H(x)$ sont interdits. Puisque le sommet x doit être dominé, alors $x \in D$ et x est un privé pour lui-même. Par conséquent, D est compatible avec $(S \cup \{x\}, X \setminus \{x\}, F)$.

Cas 7. Il existe un sommet simplicial non dominé $x \in X$ dans H avec $d_H(x) \geq 2$ tel qu'il existe un sommet $y \in N_H(x) \cap F$.

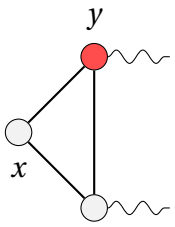

FIGURE 2.3 – Cas 7 de la phase 1

Supposons qu'il y a un sommet simplicial non dominé $x \in X$ dans H avec $d_H(x) \geq 2$ tel qu'il existe un sommet $y \in N_H(x) \cap F$. Etant donné que les étapes 4 et 6 ne sont pas appliquées, $d_H(x) \geq 2$ et $N_H(x)$ contiennent des sommets libres. Puisque x doit être dominé, alors il y a un sommet $z \in N_G[x] \setminus F$ tel que $z \in D$. On note que le sommet z domine x. Supposons que y est un privé pour z alors z est un sommet unique de D dans $N_H[x]$, car x est un sommet simplicial. Par conséquent, x est un privé pour z et z a son privé dans $V(H) \setminus \{y\}$. Par conséquent, D est compatible avec $(S, X, F \setminus \{y\})$.

□

On considère maintenant la deuxième phase de l'algorithme.

Phase 2. Premier niveau de branchement.

1. Si H a une composante connexe H' qui est un graphe complet, alors pour chaque $x \in V(H') \cap X$, appelez EnumMDS$(S \cup \{x\}, X \setminus V(H'), F \setminus V(H'))$.
2. S'il y a un sommet pendant non dominé x de H, alors notez y le voisin de x et branchez :
   (i) appelez EnumMDS$(S \cup \{y\}, X \setminus \{x, y\}, F)$,
   (ii) appelez EnumMDS$(S \cup \{x\}, X \setminus \{x, y\}, F)$.
3. S'il y a un sommet simplicial non dominé x de H tel qu'il existe un sommet simplicial $z \neq x$ de H et un sommet $y \in N_H(x) \cap N_H(z)$, alors branchez :
   (i) appelez EnumMDS$(S \cup \{y\}, X \setminus \{x, y, z\}, F \setminus \{x, z\})$,
   (ii) appelez EnumMDS$(S, X \setminus \{y\}, F)$.
4. S'il y a un sommet simplicial non dominé x de H tel qu'il existe $y \in N_H(x)$ qui est un sommet simplicial de $H - x$, alors branchez :
   (i) appelez EnumMDS$(S \cup \{y\}, X \setminus N_H[x], F \setminus \{x\})$,
   (ii) appelez EnumMDS$(S, X \setminus \{y\}, F)$.
5. S'il existe un sommet simplicial dominé x de H avec $d_H(x) \geq 2$, alors branchez :
   (i) appelez EnumMDS$(S \cup \{x\}, X \setminus N_H[x], F \setminus N_H(x))$,
   (ii) appelez EnumMDS$(S, X \setminus \{x\}, F)$.
6. S'il existe un sommet pendant dominé x de H tel que son voisin y est adjacent à un autre sommet pendant dominé z de H, alors branchez :
   (i) appelez EnumMDS$(S \cup \{x\}, X \setminus \{x, y, z\}, F \setminus \{y\})$,
   (ii) appelez EnumMDS$(S, X \setminus \{x\}, F)$.
7. S'il existe un sommet pendant dominé x de H tel que son voisin y est un sommet pendant dans $H - x$, alors notez z l'adjacent de y distinct de x.

   Si y est interdit et z est libre, alors branchez :
   (i) appelez EnumMDS$(S \cup \{z\}, X \setminus \{x, z\}, F \setminus \{y\})$,
   (ii) appelez EnumMDS$(S \cup \{x\}, X \setminus \{x, z\}, (F \cup \{z\}) \setminus \{y\})$,

   Si z est interdit et y est libre, alors branchez :
   (i) appelez EnumMDS$(S \cup \{x\}, X \setminus \{x, y\}, F)$,

(ii) appelez ENUMMDS$(S \cup \{y\}, X \setminus \{x, y,\}, F \setminus \{z\})$,

Si y et z sont libres, alors branchez :

(i) appelez ENUMMDS$(S \cup \{x\}, X \setminus \{x, y, z\}, F \cup \{z\})$,

(ii) appelez ENUMMDS$(S \cup \{y\}, X \setminus \{x, y, z\}, F)$,

(iii) appelez ENUMMDS$(S \cup \{z\}, X \setminus \{x, y, z\}, F)$.

La preuve de correction de la phase 2 est basée sur le lemme suivant.

Lemme 3. *Soit* D *un ensemble dominant minimal du graphe* G *compatible avec un triplet* (S, X, F) *des sous-ensembles disjoints de* $V(G)$. *Si la phase 1 de l'algorithme n'est pas appliquée et l'une des conditions des étapes 1–7 de la phase 2 est satisfaite, alors il y a un appel récursif de* ENUMMDS(S', X', F') *dans l'appel de* ENUMMDS(S, X, F) *tel que* D *est compatible avec* (S', X', F').

Démonstration.

On considère 7 cas correspondant aux étapes 1–7 de la phase 2 de l'algorithme.

Cas 1. H a une composante connexe H' qui est un graphe complet.

On note que $|D \cap V(H')| \leq 1$, sinon les sommets de D dans H' ne peuvent pas avoir des voisins privés. De même on note que $V(H') \cap X \neq \emptyset$ et qu'il y a au moins un sommet non dominé dans H', car la phase 1 de l'algorithme n'est pas appliquée. Donc, $|V(H') \cap D| = 1$.

Soit $x \in X$ le sommet unique de $V(H') \cap D$. Dans ce cas, n'importe quel sommet non dominé de H' est un privé pour x, alors que les autres sommets de H' ne peuvent pas être des voisins privés pour les sommets de $D \setminus (S \cup \{x\})$. Par conséquent, D est compatible avec $(S \cup \{x\}, X \setminus V(H'), F \setminus V(H'))$.

Cas 2. Il existe un sommet pendant non dominé x de H.

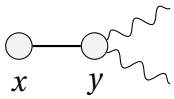

FIGURE 2.4 – Cas 2 de la phase 2

Soit y le voisin de x dans H. Etant donné que les étapes $4-6$ de la phase 1 ne peuvent pas être appliquées, alors $x, y \in X$. Si $y \in D$, alors $x \notin D$ sinon, x n'aurait pas un privé dans H, ce qui contredirait que D est compatible avec (S, X, F). Donc x est un privé pour y et il ne peut être privé pour aucun autre sommet de $D \setminus S$. Par conséquent, D est compatible avec $(S \cup \{y\}, X \setminus \{x, y\}, F)$.

On suppose maintenant que $y \notin D$. Puisque x doit être dominé, alors $x \in D$ et x est un privé pour lui-même. Par conséquent, D est compatible avec $(S \cup \{x\}, X \setminus \{x, y\}, F)$.

Cas 3. Il y a un sommet simplicial non dominé x de H tel qu'il existe un sommet simplicial $z \neq x$ de H et un sommet $y \in N_H(x) \cap N_H(z)$.

Etant donné que les étapes 6 et 7 de la phase 1 ne s'appliquent pas, alors y est un sommet libre.

Si $y \in D$, alors $x, z \notin D$ sinon, x ou z n'a pas un privé dans H, car $N_H[x]$ et $N_H[z]$ sont des cliques. De même x est un privé pour y et x, z ne peuvent pas être privés pour les sommets de $D \setminus (S \cup \{y\})$. Par conséquent, D est compatible avec $(S \cup \{y\}, X \setminus \{x, y, z\}, F \setminus \{x, z\})$.

On suppose maintenant que $y \notin D$. Le sommet x devrait être dominé par un sommet $v \in N_H[x] \cap D$. On note que $v \neq y$ domine y. On note également que si y est un privé pour v, alors v est l'unique sommet de D dans $N_H[x]$, et puisque x

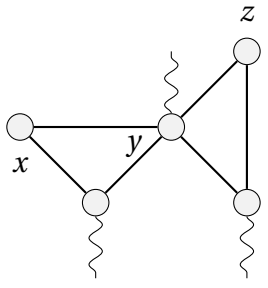

FIGURE 2.5 – Cas 3 de la phase 2

est un sommet simplicial de H, x est un privé de v. Par conséquent, D est compatible avec $(S, X \setminus \{y\}, F)$.

Cas 4. Il y a un sommet simplicial non dominé x de H tel qu'il existe $y \in N_H(x)$ qui est un sommet simplicial de $H - x$.

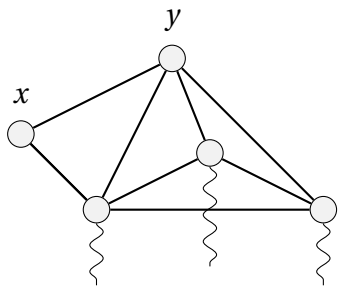

FIGURE 2.6 – Cas 4 de la phase 2

Comme les deux étapes 6 et 7 de la phase 1 ne s'appliquent pas, alors les sommets $N_H(x)$ sont des sommets libres dont le sommet y. Si $y \in D$, alors $N_H[x] \cap D = \{y\}$, sinon, si $x \in D$, alors x n'a pas de privé dans $F \cup X$, et si un sommet de $N_H(x) \setminus \{y\}$ est dans D, alors y n'a pas de privé dans $X \cup F$. Alors, x est un privé pour y et les sommets $N_H[x]$ sont dominés par y et ne peuvent pas être privés pour les sommets de $D \setminus (S \cup \{y\})$. Par conséquent, D est compatible avec $(S \cup \{y\}, X \setminus N_H[x], F \setminus \{x\})$.

On suppose maintenant que $y \notin D$. Alors x est dominé par un sommet $v \in N_H[x] \cap D$. On note que y est dominé par v et que si y est un privé pour v, alors v est l'unique sommet de D dans $N_H[x]$. Puisque x est un sommet simplicial dans H, alors x est un privé de v. Par conséquent, D est compatible avec $(S, X \setminus \{y\}, F)$.

Dans ce qu'il reste des cas, tous les sommets simpliciaux sont dominés libres.

On rappelle que tous les sommets dominés sont libres et que tous les sommets interdits ne sont pas dominés, vu que l'étape 2 de la phase 1 n'est pas appliquée.

Cas 5. Il existe un sommet simplicial dominé x de H avec $d_H(x) \geq 2$.

Si $x \in D$, alors $N_H(x) \cap D = \emptyset$, car x devrait avoir un privé dans $F \cup X$. Puisque les sommets $N_H(x)$ sont dominés par x, alors ils ne peuvent pas être privés pour les sommets de $D \setminus (S \cup \{x\})$. Par conséquent, D est compatible avec $(S \cup \{x\}, X \setminus N_H[x], F \setminus N_H(x))$.

Si $x \notin D$, alors x ne peut pas être un privé pour les sommets de $D \setminus S$, car x est dominé. Par conséquent, D est compatible avec $(S, X \setminus \{x\}, F)$.

Case 6. Il existe un sommet pendant dominé x de H tel que son voisin y est adjacent à un autre sommet pendant dominé z de H.

Si $x \in D$, alors y est son privé et $z \notin D$. Puisque y et z ne peuvent pas être privés pour les sommets de $D \setminus (S \cup \{x\})$, alors D est compatible avec $(S \cup \{x\}, X \setminus \{x, y, z\}, F \setminus \{y\})$.

Si $x \notin D$, alors x ne peut pas être un privé pour les sommets de $D \setminus S$, car x est dominé. Par conséquent, D est compatible avec $(S, X \setminus \{x\}, F)$.

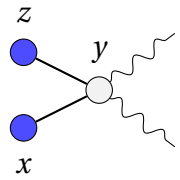

FIGURE 2.7 – Cas 6 de la phase 2

Case 7. Il existe un sommet pendant dominé x de H tel que son voisin y est un sommet pendant dans $H - x$, alors on note z l'adjacent de y distinct de x.

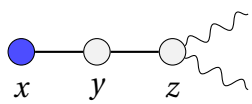

FIGURE 2.8 – Cas 7 de la phase 2

Soit z le voisin de y dans $H - x$. Si y et z sont interdits, alors x est l'unique voisin libre de y et l'étape 5 de phase 1 sera appliquée. Par conséquent, au moins l'un de deux sommets est libre.

Si y est interdit et que z est libre, alors $z \in D$ ou $x \in D$ pour dominer y.

Si $z \in D$ alors $x \notin D$ car il ne peut pas avoir un voisin privé. Comme x est dominé et que y est un sommet interdit dominé par z, alors D est compatible avec $(S \cup \{z\}, X \setminus \{x, z\}, F \setminus \{y\})$.

Si $x \in D$ alors y est son voisin privé et donc $z \notin D$. Par conséquent, D est compatible avec $(S \cup \{x\}, X \setminus \{x,z\}, (F \cup \{z\}) \setminus \{y\})$.

On Suppose que y est libre et que z est interdit. Le sommet y n'est pas dominé, car l'étape 3 de la phase 1 n'est pas appliquée. Alors $y \in D$ ou $x \in D$ pour dominer y.

Si $x \in D$ alors y est son voisin privé. Par conséquent, D est compatible avec $(S \cup \{x\}, X \setminus \{x,y\}, F)$.

Si $y \in D$ alors $x \notin D$. On note également que x, z ne peut pas être privé pour les sommets de $D \setminus (S \cup \{y\})$. Par conséquent, D est compatible avec $(S \cup \{y\}, X \setminus \{x,y\}, F \setminus \{z\})$.

On suppose maintenant que y et z sont libres.

Si $y \in D$, alors $x \notin D$ car x ne peut pas avoir un voisin privé dans $X \cup F$. On a aussi, $z \notin D$, car y devrait avoir un privé dans $X \cup F$. De même x et z ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{y\})$. Par conséquent, D est compatible avec $(S \cup \{y\}, X \setminus \{x,y,z\}, F)$.

Soit $y \notin D$.

Si $z \in D$, alors $x \notin D$, car sinon, x n'a pas un privé dans $X \cup F$. Comme $x, y \notin D$ et ils ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{z\})$ alors, D est compatible avec $(S \cup \{z\}, X \setminus \{x,y,z\}, F)$.

On suppose maintenant que $y, z \notin D$. Puisque y devrait être dominé, alors $x \in D$ et y ne peut pas être un privé pour aucun sommet de $D \setminus (S \cup \{x\})$.

Par conséquent, D est compatible avec $(S \cup \{x\}, X \setminus \{x,y,z\}, F \cup \{z\})$.

□

On résume, à ce niveau-là, les propriétés structurelles de l'instance (S, X, F) pour lesquelles les phases 1 et 2 ne sont pas applicables.

Lemme 4. *On suppose que* S, X, F *sont des sous-ensembles disjoints de* $V(G)$ *tels qu'aucune étape des phases 1 et 2 ne puisse être appliquée par* ENUMMDS(S, X, F).

Soit u un sommet simplicial de H. *Alors on a :*

a) Si u est non dominé, alors $d_H(u) \geq 2$. *De plus, pour tout autre sommet simplicial v,* $N_H(u) \cap N_H(v) = \emptyset$, *et les sommets* $N_H(u)$ *ne sont pas des sommets simpliciaux dans* $H - u$.

b) Si u est dominé, alors u est un pendant. De plus, son voisin unique v n'est pas simplicial et n'est adjacent à aucun autre sommet simplicial sauf u et $d_H(v) \geq 3$.

Démonstration. Soit u un sommet simplicial de H.

Si $d_H(u) = 0$, alors la phase 1 serait appliquée pour u. Donc $d_H(u) \geq 1$.

On suppose que u n'est pas dominé.

Si $d_H(u) = 1$, alors l'étape 2 de la phase 2 serait appliquée. Donc $d_H(u) \geq 2$. S'il y a un sommet simplicial $v \neq u$ de H tel que $N_H(u) \cap N_H(v) \neq \emptyset$, alors l'étape 3 de la phase 2 serait appliquée. Si il y a $v \in N_H(u)$ tel que v est un sommet simplicial de $H - u$, alors l'étape 4 de la phase 2 serait appliquée.

On suppose que u est dominé.

Si $d_H(u) \geq 2$, alors l'étape 5 de la phase 2 serait appliquée. Donc $d_H(u) = 1$.

Soit v le voisin de u. Si v est simplicial, alors l'étape 1 de la phase 2 serait appliquée.

On suppose maintenant que v est adjacent à un sommet simplicial $w \neq u$. Le sommet w est dominé alors, sinon l'étape 2 ou 3 de la phase 2 serait appliquée. Mais cela aussi n'est pas possible à cause des étapes 5 et 6 de la phase 2.

Finalement, si $d_H(v) = 2$, alors l'étape 7 de la phase 2 serait appliquée. Donc $d_H(v) \geq 3$.

□

On considère maintenant la troisième phase de l'algorithme.

Phase 3. Deuxième niveau de branchement.

1. S'il y a un sommet pendant dominé x de H tel que son voisin y dans H est un sommet simplicial de $H - x$ et il a au moins un voisin interdit, alors on branche en fonction de l'état des sommets $N_H(y) \setminus \{x\}$.

   Si tous les sommets de $N_H(y) \setminus \{x\}$ sont interdits, alors on branche :

   (i) appelez EnumMDS$(S \cup \{x\}, X \setminus \{x, y\}, F)$,

   (ii) appelez EnumMDS$(S \cup \{y\}, X \setminus \{x, y\}, F)$.

   Si $(N_H(y) \setminus \{x\}) \cap X \neq \emptyset$, alors on branche :

   (i) appelez EnumMDS$(S \cup \{x\}, X \setminus N_H[y], (F \cup (N_H(y) \setminus \{x\})) \setminus \{y\})$,

   (ii) appelez EnumMDS$(S, X \setminus \{x\}, F \setminus N_H(y))$.

2. S'il y a deux sommets dominés pendants x et u de H tels que le voisin y de x dans H est un sommet simplicial dans $H - x$ et le voisin z de u est dans $N_H(y)$, alors on branche :

   (i) appelez EnumMDS$(S \cup \{u, x\}, X \setminus (N_H[z] \cup \{x\}), (F \cup N_H(z)) \setminus \{y, u\}))$,

   (ii) appelez EnumMDS$(S, X \setminus \{u\}, F)$.

3. S'il y a un sommet pendant dominé x de H et un sommet simplicial non dominé u de H tel que le voisin y de x dans H est un sommet simplicial dans $H - x$ et il y a un sommet $z \in N_H(y) \cap N_H(u)$, alors on branche :

   (i) appelez EnumMDS$(S \cup \{z\}, X \setminus \{x, y, z, u\}, F \setminus \{y, u\})$,

   (ii) appelez EnumMDS$(S, X \setminus \{z\}, F)$.

4. S'il y a deux sommets dominés pendants x et u de H tels que leurs voisins y et v respectivement sont des sommets simpliciaux de $H - \{x, u\}$ et $N_H(v) \setminus \{u\} \subseteq N_H(y) \setminus \{x\}$, alors on branche :

   Si v est libre, alors on branche :

   (i) appelez EnumMDS$(S \cup \{x, u\}, X \setminus (N_H[y] \cup \{u, v\}), (F \cup N_H(y)) \setminus \{x, y\})$,

   (ii) appelez EnumMDS$(S \cup \{x, v\}, X \setminus (N_H[y] \cup \{u, v\}), (F \cup N_H(y)) \setminus (N_H(v) \cup \{x, y\}))$,

   (iii) appelez EnumMDS$(S, X \setminus \{x\}, F)$.

   Si v est interdit, alors on branche :

   (i) appelez EnumMDS$(S \cup \{x, u\}, X \setminus (N_H[y] \cup \{u, v\}), (F \cup N_H(y)) \setminus \{x, y, v\})$,

   (iii) appelez EnumMDS$(S, X \setminus \{x\}, F)$.

5. S'il y a deux sommets dominés pendants x et u de H tels que leurs voisins y et v respectivement sont des sommets simpliciaux de $H - \{x, u\}$ et il y a un sommet $z \in N_H(y) \cap N_H(v)$, alors on branche :

   (i) appelez EnumMDS$(S \cup \{z\}, X \setminus \{z, x, u, y, v\}, F \setminus \{y, v\})$,

   (ii) appelez EnumMDS$(S \cup \{x, u\}, X \setminus (N_H[v] \cup N_H[y]), F \cup (N_H(v) \cup N_H(y)) \setminus \{x, u, y, v\})$,

   (iii) appelez EnumMDS$(S, X \setminus \{z\}, F)$.

6. S'il existe un sommet pendant dominé x de H tel que son voisin y est un sommet simplicial de $H - x$ et il y a $z \in N_H(y) \setminus \{x\}$ tel que z est un sommet simplicial de $H - \{x, y\}$, alors on branche :

   Si z est dominé, alors on branche :

(i) appelez ENUMMDS($S \cup \{z\}, X \setminus N_H[y], (F \cup N_H(y)) \setminus \{x,z\}$),

(ii) appelez ENUMMDS($S, X \setminus \{z\}, F$).

Si z n'est pas dominé, alors on branche :

(i) appelez ENUMMDS($S \cup \{z\}, X \setminus N_H[y], (F \cup N_H(y)) \setminus \{x,z\}$),

(ii) appelez ENUMMDS($S \cup \{x\}, X \setminus N_H[y], (F \cup \{z\}) \setminus \{y\}$),

(iii) appelez ENUMMDS($S, X \setminus \{x,z\}, F$).

Lemme 5. *Soit* D *un ensemble dominant minimal du graphe* G *compatible avec un triplet* (S,X,F) *des sous-ensembles disjoints de* $V(G)$. *Si les phases 1 et 2 de l'algorithme ne sont pas appliquées et l'une des conditions des étapes 1–6 de la phase 3 est satisfaite, alors il y a un appel récursif de* ENUMMDS(S',X',F') *dans l'appel de* ENUMMDS(S,X,F) *tel que* D *est compatible avec* (S',X',F').

Démonstration.

On considère 6 cas correspondant aux étapes 1–6 de la phase 3 de l'algorithme. On suppose à chaque cas que tous les cas précédents ne s'appliquent pas. De même, les sommets pendants sont dominés et ils ont des voisins non dominés, car l'étape 3 de la phase 1 ne s'applique pas.

Cas 1. S'il y a un sommet pendant dominé x de H tel que son voisin y dans H est un sommet simplicial de $H - x$ et il a au moins un voisin interdit.

On note qu'au moins un sommet de $N_H[y] \setminus \{x\}$ est libre, car l'étape 5 de la phase 1 n'est pas appliquée.

On suppose que tous les sommets $N_H(y) \setminus \{x\}$ sont interdits.

Dans ce cas, le sommet y est alors libre. Le sommet y doit être dominé, alors on a $x \in D$ ou $y \in D$.

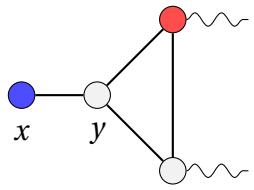

FIGURE 2.9 – Cas 1 de la phase 3

Si $x \in D$ alors y est son voisin privé. Par conséquent, D est compatible avec $(S \cup \{x\}, X \setminus \{x,y\}, F)$.

Soit $x \notin D$ et $y \in D$. Alors y est un privé pour lui-même, et x ne peut pas être un voisin privé pour un sommet de $D \setminus (S \cup \{y\})$. Par conséquent, D est compatible avec $(S \cup \{y\}, X \setminus \{x,y\}, F)$.

Supposons que $(N_H(y) \setminus \{x\}) \cap X \neq \emptyset$.

Si $x \in D$, alors y est son voisin privé et, par conséquent, $(N_H(y) \setminus \{x\}) \cap D = \emptyset$. Cela implique que D est compatible avec $(S \cup \{x\}, X \setminus N_H[y], F \cup (N_H(y) \setminus \{x\}) \setminus \{y\})$.

Si $x \notin D$, alors x ne peut pas être un voisin privé pour aucun sommet de D. Le sommet y doit être dominé par un sommet $v \in D \setminus S$. Puisque y est un sommet simplicial de $H - x$, cela implique que v domine tous les sommets $N_H[y] \setminus \{x\}$ et que y est un privé de v. Par conséquent, D est compatible avec $(S, X \setminus \{x\}, F \setminus N_H(y))$.

Comme l'étape 1 de la phase 3 ne s'applique plus, alors $N_H(y) \subseteq X$, pour tout y voisin d'un sommet pendant dominé x et y est un sommet simplicial dans $H - x$.

Cas 2. S'il y a deux sommets dominés pendants x et u de H tels que le voisin y de x dans H est un sommet simplicial dans $H - x$ et le voisin z de u est dans $N_H(y)$.

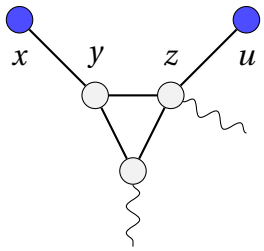

FIGURE 2.10 – Cas 2 de la phase 3

On suppose que $u \in D$. Alors le sommet z est son privé et, par conséquent, $(N_H[z] \setminus \{u\}) \cap D = \emptyset$. Mais le sommet y doit être dominé. Cela implique que $x \in D$. Clairement, y et z ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{x, u\})$. Par conséquent, D est compatible avec $(S \cup \{u, x\}, X \setminus (N_H[z] \cup \{x\}), (F \cup N_H(z)) \setminus \{y, u\})$. On suppose que $u \notin D$. Puisque u est dominé, il ne peut pas être un privé pour les sommets de $D \setminus S$. Par conséquent, D est compatible avec $(S, X \setminus \{u\}, F)$.

Cas 3. S'il y a un sommet pendant dominé x de H et un sommet simplicial non dominé u de H tel que le voisin y de x dans H est un sommet simplicial dans $H - x$ et il y a un sommet $z \in N_H(y) \cap N_H(u)$.

On suppose que $z \in D$. Alors $x, y, u \notin D$, car ils ne peuvent pas avoir un privé dans $X \cup F$. De même x, y, u ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{z\})$. Par conséquent, D est compatible avec $(S \cup \{z\}, X \setminus \{x, y, z, u\}, F \setminus \{y, u\})$.

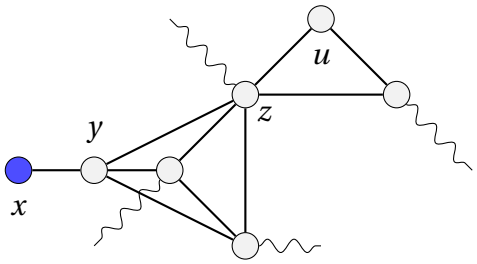

FIGURE 2.11 – Cas 3 de la phase 3

On suppose maintenant que $z \notin D$. Le sommet u est dominé par un sommet $v \in D \setminus S$. Le sommet $v \in N_H[u] \setminus \{z\}$ domine u et z. De plus u est le privé de v. Par conséquent, D est compatible avec $(S, X \setminus \{z\}, F)$.

Cas 4. S'il y a deux sommets dominés pendants x et u de H tels que leurs voisins y et v respectivement sont des sommets simpliciaux de $H - \{x, u\}$ et $N_H(v) \setminus \{u\} \subseteq N_H(y) \setminus \{x\}$.

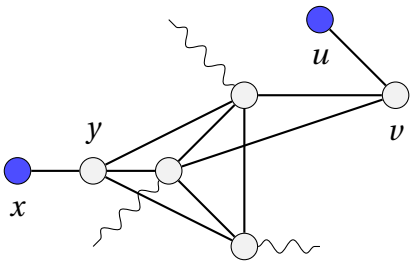

FIGURE 2.12 – Cas 4 de la phase 3

Soit v un sommet libre.

On suppose que $x \in D$. Alors le sommet y est son privé et les sommets $N_H[y] \setminus \{x\}$ ne sont pas dans D. Mais comme le sommet v doit être dominé, alors soit $u \in D$ ou soit $v \in D$. Si $u \in D$, alors v est son privé. Par conséquent, D est compatible avec $(S \cup \{x, u\}, X \setminus (N_H[y] \cup \{u, v\}), (F \cup N_H(y)) \setminus \{x, y\})$. Si $v \in D$, alors v est un privé pour lui-même et les sommets de $N_H(v)$ sont dominés par v et ne peuvent pas être privés pour les autres sommets. Par conséquent, D est compatible avec $(S \cup \{x, v\}, X \setminus (N_H[y] \cup \{u, v\}), (F \cup N_H(y)) \setminus (N_H(v) \cup \{x, y\}))$. Si $x \notin D$, alors D est compatible avec $(S, X \setminus \{x\}, F)$, car x est dominé.

Soit v un sommet interdit.

Si $x \in D$, alors $u \in D$, car v devrait être dominé. Alors D est compatible avec $(S \cup \{x, u\}, X \setminus (N_H[y] \cup \{u, v\}), (F \cup N_H(y)) \setminus \{x, y, v\})$. Si $x \notin D$, alors D est compatible avec $(S, X \setminus \{x\}, F)$.

Cas 5. S'il y a deux sommets dominés pendants x et u de H tels que leurs voisins y et v respectivement sont des sommets simpliciaux de $H - \{x, u\}$ et il y a un sommet

$z \in N_H(y) \cap N_H(v)$.

On suppose que $z \in D$. Dans ce cas $x, y, u, v \notin D$, sinon ces sommets n'ont pas de privés dans $X \cup F$ et ils ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{z\})$. Par conséquent, D est compatible avec $(S \cup \{z\}, X \setminus \{z, x, u, y, v\}, F \setminus \{y, v\})$.

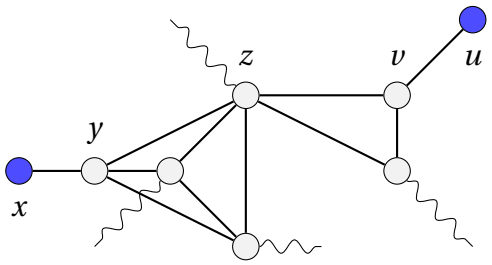

FIGURE 2.13 – Cas 5 de la phase 3

Soit $z \notin D$. Si $x, u \in D$, alors les sommets y, v sont dominés et sont des privés de x et u respectivement. Et aussi les sommets $N_H[y] \setminus \{x\}$ et $N_H[v] \setminus \{u\}$ ne sont pas dans D. Par conséquent, D est compatible avec $(S \cup \{x, u\}, X \setminus (N_H[y] \cup N_H[v]), F \cup (N_H(y) \cup N_H(v)) \setminus \{x, u, y, v\})$.

On suppose maintenant que $x \notin D$ ou $u \notin D$. Sans perte de généralité, on suppose que $x \notin D$. Dans ce cas le sommet y doit être dominé par un sommet $w \in D \setminus S$ tel que $w \in N_H[y] \setminus \{x, z\}$. Puisque y est un sommet simplicial de $G - x$, y est un privé de w. On a alors le sommet z est dominé par w. Par conséquent, D est compatible avec $(S, X \setminus \{z\}, F)$.

Cas 6. S'il existe un sommet pendant dominé x de H tel que son voisin y est un sommet simplicial de $H - x$ et il y a $z \in N_H(y) \setminus \{x\}$ tel que z est un sommet simplicial de $H - \{x, y\}$.

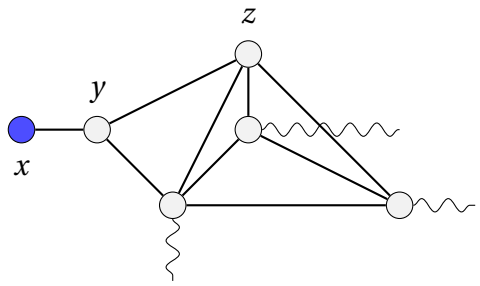

FIGURE 2.14 – Cas 6 de la phase 3

On suppose que z est dominé. Soit $z \in D$. Alors les sommets $x, y \notin D$ car ils ne peuvent pas avoir des privés dans $X \cup F$. On note également que $(N_H(y) \setminus \{x, z\}) \cap D = \emptyset$, sinon le sommet z n'a pas un privé dans $X \cup F$. De même, les sommets de $N_H[y] \setminus \{z\}$ ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{z\})$. Par conséquent, D est compatible avec $(S \cup \{z\}, X \setminus N_H[y], (F \cup N_H(y)) \setminus \{x, z\})$. On suppose que $z \notin D$, alors D est compatible avec $(S, X \setminus \{z\}, F)$.

On suppose que z n'est pas dominé.

Soit $z \in D$. De la même manière, on peut déduire que D est compatible avec $(S \cup \{z\}, X \setminus N_H[y], (F \cup N_H(y)) \setminus \{x, z\})$. On suppose maintenant que $z \notin D$. Soit $x \in D$. Le sommet y est son privé et, par conséquent, $(N_H[y] \setminus \{x\}) \cap D = \emptyset$. Puisque z n'est pas dominé, il y a $v \in D \setminus S$ qui domine z. On note que $v \notin N_H(y)$, v domine $N_H(y) \setminus \{x\}$ et z est un privé pour v. Par conséquent, D est compatible avec $(S \cup \{x\}, X \setminus N_H[y], (F \cup \{z\}) \setminus \{y\})$. Soit maintenant $x \notin D$. Le sommet dominé x ne peut être privé pour aucun sommet de $D \setminus S$. Le sommet y devrait être dominé par un sommet v de $N_H[y] \setminus \{x, z\}$. Si z est un sommet privé de v, alors $N_H[y] \cap D = \{v\}$. Donc le sommet y est un privé de v également. Par conséquent, D est compatible avec $(S, X \setminus \{x, z\}, F)$.

□

On a donné dans cette section une desciption détaillée de l'algorithme et on donne dans la section suivante une preuve de sa correction.

2.2.2 La correction de l'algorithme

Afin de montrer que l'algorithme est correct, on prouve tout d'abord que les étapes de l'algorithme sont exhaustives.

Lemme 6. *Soit* $S, X, F \subseteq V(G)$ *des sous-ensembles disjoints, alors au moins une étape de* EnumMDS(S, X, F) *est applicable.*

Démonstration. Soit S, X, F des sous-ensembles disjoints de $V(G)$. On note P l'ensemble des sommets simpliciaux pendants de H et $H' = H - P$.

On suppose qu'aucune étape des phases 1 et 2 ne peut pas être appliquée. On suppose également que l'étape 1 de la phase 3 ne s'applique pas. D'après le Lemme 4, pour tout $u \in P$, u est dominé et son voisin unique v, n'est pas simplicial et n'est adjacent à aucun autre sommet simplicial sauf u et $d_H(v) \geq 3$. Cela implique que chaque composante de H' a au moins deux sommets. Or par le Lemme 1, il existe un sommet simplicial x de H' et $y \in N_{H'}(x)$ tel que y est un sommet simplicial de $H' - x$ ou il y a un sommet simplicial $z \neq x$ de H' tel que $xz \notin E(H)$ et $y \in N_{H'}(x) \cap N_{H'}(z)$.

On suppose qu'il existe des sommets simpliciaux distincts non adjacents x et z de H' avec un voisin commun y.

On suppose que x et z ne sont pas adjacents aux sommets de P dans H. Alors x et z sont aussi des sommets simpliciaux de H. Puisque $x, z \notin P$, alors ils sont de degré au moins 2 et, par conséquent, x et z ne sont pas dominés par Lemme 4. Mais $N_H(u) \cap N_H(v) = \emptyset$ par Lemme 4; contradiction. Par conséquent, x ou z a un voisin dans P. On suppose sans perte de généralité que x est adjacent à $u \in P$. Par Lemme 4, u est l'unique voisin de x dans P. Cela signifie que x est un sommet simplicial de $H - u$. S'il y a $v \in P$ tel que v est adjacent à un sommet de $N_H(x)$, alors l'étape 2 de la phase 3 est applicable. S'il y a $v \in P$ tel que v soit adjacent à z, alors les étapes 4 ou 5 de la phase 3 sont applicables. Sinon, z est un sommet simplicial de H. Et comme $z \notin P$, alors le sommet z n'est pas dominé, et par conséquent, l'étape 3 de la phase 3 est applicable.

On suppose maintenant qu'il y a un sommet simplicial x de H' et $y \in N_{H'}(x)$ tel que y est un sommet simplicial de $H' - x$.

On suppose que x n'a pas de voisin dans P. Alors x est un sommet simplicial de H. Puisque $x \notin P$, $d_H(x) \geq 2$ alors x est un sommet non dominé par le Lemme 4. Aussi par Lemme 4, pour tout sommet simplicial $v \neq x$, $N_H(x) \cap N_H(v) = \emptyset$ et les sommets $N_H(x)$ ne sont pas des simpliciaux dans $H - x$. Donc les sommets $N_H(x)$ n'ont pas de voisins dans P et y alors est un sommet simplicial de $H - x$; contradiction. Donc x a un voisin $u \in P$ et par le Lemme 4, ce voisin est unique. S'il y a $v \in P$ tel que v est adjacent à un sommet de $N_H(x)$, alors l'étape 2 de la phase 3 est applicable. Sinon, y est un sommet simplicial de $H - \{u, x\}$ et l'étape 6 de la phase 3 est applicable.

□

On termine maintenant la correction de l'algorithme.

A chaque étape de l'algorithme EnumMDS(S, X, F), soit on produit S à l'étape 1 de la phase 1, soit on appelle EnumMDS(S', X', F') tel que $|X'| + |F'| < |X| + |F|$. Aussi, par Lemme 6, les étapes de l'algorithme sont exhaustives. Donc l'algorithme s'arrête toujours.

Afin d'énumérer tous les ensembles dominants minimaux dans un graphe cordal G, il suffit d'appeler EnumMDS$(\emptyset, V(G), \emptyset)$.

L'algorithme génère des ensembles seulement à l'étape 1 de la phase 1 et il affiche, après une vérification en temps polynomial, un ensemble s'il est uniquement un ensemble dominant minimal. Par conséquent, EnumMDS$(\emptyset, V(G), \emptyset)$ ne produit que des ensembles dominants minimaux.

On montre alors que EnumMDS$(\emptyset, V(G), \emptyset)$ produit chaque ensemble dominant minimal.

Soit D un ensemble dominant minimal. On montre, par récurrence sur $|X| + |F|$, que si D est compatible avec un triplet (S, X, F) des sous-ensembles disjoints de $V(G)$, alors

ENUMMDS(S,X,F) produit D.

Si |X| + |F| = 0, alors c'est trivial, car D = S dans ce cas et on produit D à l'étape 1 de la phase 1.

On suppose que l'affirmation est vraie pour $|X''| + |F''| < |X| + |F|$. Et on le prouve pour |X| + |F|.

Par Lemme 6, on appelle ENUMMDS(S′,X′,F′) dans l'une des étapes de l'algorithme. Et par les Lemmes 2, 3 and 5, D est compatible avec (S′,X′,F′). Comme $|X'| + |F'| < |X| + |F|$, alors par l'hypothèse de récurrence ENUMMDS(S′,X′,F′) produit D.

Donc si D est compatible avec un triplet (S,X,F) des sous-ensembles disjoints de V(G), alors ENUMMDS(S,X,F) produit D.

De plus, tout ensemble dominant minimal D est compatible avec (∅,V(G),∅). Donc ENUMMDS(∅,V(G),∅) énumère tous les ensembles dominants minimaux du graphe G.

Finalement, il reste à analyser le temps d'exécution de l'algorithme.

2.2.3 L'analyse du temps d'exécution

Afin d'analyser le temps d'exécution, on associe à chaque règle de branchement de l'algorithme, un vecteur de branchement où la mesure d'une instance (S,X,F) est définie par la somme pondérée $|S| + |X| + p|F|$, avec $p = 0.836398$.

Notant que tous les étapes de la phase 1 sont des règles de réduction. Donc il suffit d'analyser les phases 2 et 3.

Phase 2

Etape 1. Soit $t = |V(H') \cap X|$ avec $t \geq 2$. On obtient le vecteur $(\underbrace{t,\ldots,t}_{t})$. La valeur maximale du son facteur est atteinte pour $t = 3$ et vaut $3^{1/3} < 1.4423$.

Le vecteur de branchement $(\underbrace{t,\ldots,t}_{t})$ avec $t \geq 2$ est connu de type « Moon and Moser ». Remontrons que la valeur maximale de son facteur est atteinte pour $t = 3$ [25].

Démonstration. Le nombre de feuilles T(n), associé à la règle de branchement $b = (\underbrace{t,\ldots,t}_{t})$, est borné par l'équation suivante :

$$T(n) \leq t \times T(n-t) \tag{2.2}$$

Alors il suffit de résoudre l'équation :

$$T(n) = t \times T(n-t) \tag{2.3}$$

Et comme la solution est de la forme α^n alors :

$$x^n = t \times x^{n-t} \tag{2.4}$$
$$x^t = t \tag{2.5}$$
$$x = \sqrt[t]{t} \tag{2.6}$$

Afin de calculer le maximum de x, on calcule le dérivé de $\sqrt[t]{t}$:

$$(\sqrt[t]{t})' = t^{\frac{1}{t-2}}(1 - \log(t)) \tag{2.7}$$

Comme pour tout entier $t > e$, on a $(\sqrt[t]{t})' < 0$. Et le facteur de branchement $\tau(3,3,3) > \tau(2,2)$. Alors le vecteur de branchement $(\underbrace{t,\ldots,t}_{t})$ atteint son maximum pour $t = 3$.

□

Etape 2. Si $x \in F$ ou $y \in F$, la phase 1 sera appliquée. Comme $x, y \in X$ alors le vecteur de branchement est $(2,2)$ et son facteur est inférieur à 1.4143.

Etape 3. Le sommet $y \in X$, car l'étape 7 de la phase 1 n'est pas appliquée. Et dans le pire des cas, les sommets x et z sont interdits. Par conséquent, le vecteur de branchement est $(1+2p,1)$ et son facteur est arrondi à 1.5048.

Etape 4. Observant que $d_H(x) \geq 2$ car l'étape 2 de la phase 2 n'est pas appliquée et que $N_H(x) \subseteq X$ à cause de l'étape 7 de la phase 1. Dans le pire des cas, $x \in F$ et $d_H(x) = 2$, le vecteur de branchement est $(2+p,1)$ et son facteur est inférieur à 1.4843.

Etape 5. Le sommet $x \in X$, à cause de l'étape 2 de la phase 1. Et dans le pire des cas, $N_H(x) \subseteq F$ et $d_H(x) = 2$. Le vecteur de branchement est alors $(1+2p,1)$ et son facteur est arrondi à 1.5048.

Etape 6. Les sommets x et z sont libres, car l'étape 2 de la phase 1 n'est pas appliquée et dans le pire des cas, le sommet $y \in F$. Par conséquent, le vecteur de branchement est $(2+p,1)$ et son facteur est inférieur à 1.4843.

Etape 7. En déplaçant un sommet de X à F, on gagne seulement $1-p = 0.163602$. Les trois vecteurs de branchement de cette étape sont respectivement $(2+p,2)$, $(2,2+p)$ et $(3-p,3,3)$. La valeur maximale du facteur de branchement est atteinte pour le vecteur $(3-p,3,3)$ et elle est arrondi à 1.5048.

Phase 3

Dans cette phase, on considère à chaque étape des sommets pendants dominés. Ces sommets sont libres en raison de l'étape 2 de la phase 1 et leurs voisins sont de degré au moins 3 par le Lemme 4.

Etape 1.

Si $N_H(y) \setminus \{x\} \subseteq F$, alors $y \in X$ à cause de l'étape 5 de la pahse 1. Le vecteur de branchement est dans ce cas $(2,2)$ et son facteur est inférieur à 1.4143.

Sinon si $(N_H(y) \setminus \{x\}) \cap X \neq \emptyset$, alors dans le pire des cas, $d_H(y) = 3$, $y \in F$, un sommet de $N_H(y) \setminus \{x\}$ est interdit et l'autre est libre. Le vecteur de branchement est alors $(2,1+p)$ et son facteur est inférieur à 1.4356.

A partir de cette étape, on a pour tout sommet pendant dominé x et son voisin unique y, $N_H(y) \subseteq X$.

Etape 2. Les sommets $u, x, z \in X$ et dans le pire des cas le sommet $y \in F$ et $d_H(y) = 3$. Le vecteur de branchement est alors $(4,1)$ et son facteur est inférieur à 1.3803.

Etape 3. Les sommets $x, z \in X$ et dans le pire des cas $y, u \in F$. Alors le vecteur de branchement est $(2+2p,1)$ et son facteur est inférieur à 1.4039.

Etape 4. Le sommet $u \in X$ et $N_H(y) \subseteq X$ et dans le pire des cas $d_H(y) = d_H(v) = 3$ et $y \in F$. Si v est libre, alors le vecteur de branchement est $(5-p,5+p,1)$ et son facteur est inférieur à 1.4612. Sinon si v est interdit, alors le vecteur de branchement est $(4,1)$ dans ce cas et son facteur est inférieur à 1.3804.

Etape 5. Les sommets $(N_H(y) \cup N_H(v)) \subseteq X$ et dans le pire des cas $d_H(y) = d_H(v) = 3$ et $y, v \in F$. Et on a aussi $|(N_H(y) \cup N_H(v)) \setminus \{x, u\}| = 3$. Alors le vecteur de branchement est $(3+2p, 5-p, 1)$ et son facteur est inférieur à 1.5014.

Etape 6. Les sommets $N_H(y) \subseteq X$. Dans le pire des cas $d_H(y) = 3$ et $y \in F$. Si z est dominé, alors le vecteur de branchement est $(3,1)$ et son facteur est inférieur à 1.4565. Sinon si z n'est pas dominé, le vecteur de branchement est alors $(3+p, 3, 2)$ et son facteur est inférieur à 1.4731.

L'analyse du temps d'exécution montre que le pire facteur de branchement est inférieur strictement à 1.5048. Donc on peut déduire que l'algorithme s'exécute en $\mathcal{O}(1.5048)$, ainsi que le théorème suivant :

Théorème 8. *Un graphe cordal de n sommets a au plus* 1.5048^n *ensembles dominants minimaux qu'on peut énumérer en* $\mathcal{O}(1.5048^n)$.

2.3 Conclusion

D'un côté, on est arrivé à réduire l'écart entre la borne supérieure et la borne inférieure pour le nombre maximum des ensembles dominants minimaux dans les graphes cordaux. Cela implique directement que le nombre maximum des ensembles dominants minimaux dans les graphes fortement cordaux de n sommets est au plus 1.5048^n. D'un autre côté, récemment BERGOUGNOUX et collab. [5] ont réussi à donner un algorithme de programmation dynamique qui permet de compter le nombre des ensembles dominants minimaux dans les graphes fortement cordaux en temps polynomial. Cela peut servir d'améliorer la borne inférieure des graphes fortement cordaux, et même probablement les graphes cordaux.

Le problème reste ouvert aussi pour les graphes sans griffes, bipartis et les graphes planaires. Ces classes de graphes ont la même borne supérieure qu'un graphe général, malgré leurs structures très spécifiques. On note bien que la borne inférieure décrite dans la section 2.1.1 a un plongement dans le plan.

L'ensemble dominant total minimal est une variante très proche de l'ensemble dominant minimal. La seul différence est que dans un ensemble dominant total minimal, un sommet ne peut pas être un privé pour lui-même (voir la définition 4).

L'énumération des ces ensembles n'est pas encore étudié, sachant qu'on peut obtenir une borne supérieure de 1.7159^n avec l'algorithme de FOMIN et collab. [24] décrit dans la section 2.1.1 en énumérant tous les ensembles minimaux de couverture par ensembles de l'instance (U, S), où l'univers U est l'ensemble de sommets du graphe et la famille des sous-ensembles $S = \{\cup_{u \in U} N(u)\}$.

On propose aussi 1.5848^n comme une borne inférieure construite par l'union des K_5. Chaque k_5 contient 10 (combinaisons de 2 parmi 5) ensembles dominants totaux minimaux.

On espère que ces observations soient une bonne initiation pour attirer l'attention sur le problème de l'énumération des ensembles dominants totaux minimaux.

Dans le chapitre suivant, on n'a pas étudié ce dernier problème, mais on a étudié le problème d'énumération des ensembles irredondants maximaux qui est très relié aussi à l'énumération des ensembles dominants minimaux. Où chaque ensemble dominant minimal est un ensemble irredondant maximal.

Finalement on donne un survol, sous forme d'un tableau, des résultats connus sur l'énumération « input-sensitive » des ensembles dominants minimaux, où n, m et $w(G)$ sont respectivement le nombre de sommets, le nombre des arêtes et la taille de la clique de cardinalité maximale du graphe G.

Les classes de graphes	Borne inférieure	Borne supérieure
Les graphes quelconques	$15^{n/6}$ [24]	1.7159^n [24]
Les graphes cordaux	$3^{n/3}$ [17]	1.5048^n [cette thèse]
Les graphes scindés	$3^{n/3}$ [17]	$3^{n/3}$ [18]
Les graphes d'intervalles	$3^{n/3}$ [17]	$3^{n/3}$ [18]
Les cographes	$15^{n/6}$ [24]	$15^{n/6}$ [17]
Les graphes trivialement parfaits	$3^{n/3}$ [17]	$3^{n/3}$ [17]
Les seuils	$w(G)$ [17]	$w(G)$ [17]
Les chaines	$\lfloor n/2 \rfloor + m$ [17]	$\lfloor n/2 \rfloor + m$ [17]
Les graphes cobipartis	1.3195^n [16]	$n^2 + 2 \times 1.3674^n$ [61]
Les arbres	1.4195^n [59]	1.4195^n [59]

TABLEAU 2.1 – Un survol sur l'énumération des ensembles dominants minimaux

Chapitre 3

Enumération des ensembles irredondants maximaux

Sommaire

3.1 Etat de l'art

Soit G = (V, E) un graphe. Un ensemble de sommets D ⊆ V est un ensemble irredondant si pour tout $v \in D$, v a un privé dans G. L'ensemble D est maximal s'il n'existe aucun autre ensemble irredondant D′ tel que D ⊂ D′. Il est clair que chaque ensemble dominant minimal est un ensemble irredondant maximal. Mais l'inverse est incorrect, c'est à dire un ensemble irredondant maximal peut être non dominant minimal. Dans la figure 3.1, l'ensemble {1,2} est bien un ensemble irredondant maximal mais non dominant vu que le sommet 4 n'est pas dominé. Les cardinalités minimale et maximale d'un ensemble irredondant maximal dans un graphe G sont notées respectivement ir(G) et IR(G). Le problème d'irredondance dans ses deux versions est « NP-complet » [22, 41]. Compter la cardinalité minimale ir est « NP-complet » dans les graphes scindés [53]. Et donc c'est « NP-complet » dans les graphes cordaux. Par contre, on peut la compter en $\mathcal{O}(n^3)$ dans les graphes d'intervalles [12]. Le problème d'irredondance de cardinalité maximale est équivalent au nombre d'indépendance dans les graphes cordaux [44] qu'on peut résoudre en temps polynomial [28].

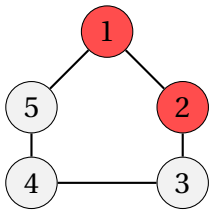

FIGURE 3.1 – Un ensemble irredondant maximal non dominant

Dans les graphes quelconques, on peut compter ir(G) en $\mathcal{O}^*(1.99914^n)$ en utilisant seulement un espace polynomial [6]. Et afin de compter IR(G), il existe deux algorithmes. Le premier s'exécute en $\mathcal{O}^*(1.9369^n)$ avec un espace polynomial, alors que le deuxième algorithme s'exécute plus rapidement en $\mathcal{O}(1.8475^n)$ mais en utilisant un espace exponentiel [6].

Concernant l'énumération des ensembles irredondants maximaux, il n'existe aucun algorithme connu mieux que l'algorithme trivial en $\mathcal{O}^*(2^n)$.

Récemment GOLOVACH, KRATSCH et SAYADI [35] ont proposé un algorithme qui énumère tous les ensembles irredondants maximaux dans les graphes sans griffes en $\mathcal{O}(1.9341^n)$. De même GOLOVACH, KRATSCH, LIEDLOFF, RAO et SAYADI [34] ont montré que le nombre maximum des ensembles irredondants maximaux dans les chemins de n sommets est $\Theta(1.4696\ldots^n)$.

Concernant les algorithmes « output-sensitive », KANTÉ et collab. [46] ont montré que chaque ensemble irredondant maximal dans les graphes scindés est un ensemble dominant minimal. Et par conséquent on peut énumérer ces ensembles en délai polynomial par l'énumération des ensembles dominants minimaux [46]. De même on peut déduire que le nombre maximum des ensembles irredondants maximaux dans les graphes scindés est $3^{n/3}$ [18].

3.1.1 Les graphes quelconques

Borne supérieure

Le meilleur algorithme connu jusqu'à présent est celui qui énumère tous les sous-ensembles possibles de V et vérifie en temps polynomial chaque ensemble s'il est un irredondant maximal. Cet algorithme trivial s'exécute en $\mathcal{O}^*(2^n)$.

Borne Inférieure

Dans l'article [35], GOLOVACH, KRATSCH et SAYADI ont proposé une borne composée par $n/5$ cycles C_5. Dans chaque cycle C_5, il y a 10 ensembles irredondants maximaux :

— On a 5 ensembles correspondant aux paires de sommets adjacents.

— Plus 5 ensembles correspondant aux paires de sommets de distance 2.

Donc on a $10^{n/5} \simeq 1.5848^n$ ensembles irredondants maximaux.
Cette borne inférieure est une famille des graphes sans griffes.

3.1.2 Les graphes sans griffes

Les auteurs GOLOVACH, KRATSCH et SAYADI [35] ont initialisé leur algorithme par le calcul d'un ensemble W de cardinalité maximale tel que $G - W$ est une union disjointe de graphes complets. Ce problème est connu sous le nom de « Cluster Vertex Deletion » qu'on peut résoudre en $\mathcal{O}(1.4765^n)$. Les auteurs ont fourni deux algorithmes selon la taille de W.

Si $|W| > \varepsilon n$, le premier algorithme branche sur un chemin induit P de 3 sommets du graphe G, avec $V(P) = \{v_1, v_2, v_3\}$ en constatant que soit $v_1 \notin D$, soit $v_1 \in D$ et $v_2 \notin D$, soit $v_1, v_2 \in D$ et $v_3 \notin D$. Une fois, il n'y a plus de chemins induits, les auteurs ont considéré tous les cas possibles par force brute. Le temps d'exécution de ce premier algorithme est $O^*(\lambda^{\varepsilon n} \cdot 2^{(1-\varepsilon)n})$ avec $\lambda \approx 1.8393$.

Sinon $|W| \leq \varepsilon n$. Les auteurs ont considéré $3^{|W|}$ partitions (A, B, C) tels que $D \cap W = A$, les sommets de B sont des sommets privés de certains sommets de D, et chaque sommet de D a au moins un privé dans $V(G) \setminus C$.

Pour chaque partition (A, B, C), et à partir de l'union disjointe de graphes complets $G - W$, l'algorithme énumère tous les ensembles irredondants maximaux en $\mathcal{O}^*(3^{(n-|W|)/3})$. Et comme il y a $3^{|W|}$ partitions alors l'énumération se fait en $\mathcal{O}^*(3^{|W|} \cdot 3^{(n-|W|)/3})$.

Finalement, la balance de deux temps d'exécutions par une technique appelée « win-win » donne un $\varepsilon \approx 0.4006$. Cela implique que l'énumération des ensembles irredondants maximaux dans les graphes sans griffes se fait en $\mathcal{O}(1.9341^n)$ et que le nombre maximum de ces ensembles dans un graphe sans griffes de n sommets est au plus 1.9341^n.

3.2 Les ensembles irredondants maximaux versus les ensembles dominants minimaux

Il semble que la conception des algorithmes d'énumération des ensembles irredondants maximaux est plus difficile que la conception des algorithmes d'énumération des ensembles dominants minimaux. Tout d'abord, contrairement à un ensemble dominant minimal, la contrainte qu'un ensemble soit irredondant maximal n'est pas locale. En fait, on peut avoir un sommet qui n'est dominé par aucun sommet d'un ensemble irredondant maximal. Et il semble que ce n'est pas facile de le fixer immédiatement.

Ensuite, vu que chaque ensemble dominant minimal est un ensemble irredondant maximal, cela implique directement que le nombre maximum des ensembles irredondants maximaux dans un graphe est plus grand que le nombre maximum des ensembles dominants minimaux. Et donc l'algorithme d'énumération « input-sensitive » est pire du point de vue temps d'exécution. Déjà l'énumération des ensembles dominants minimaux dans un graphe général se fait en $\mathcal{O}(1.7159^n)$ [24] alors qu'il n'existe aucun algorithme

connu mieux que l'algorithme en $\mathcal{O}^*(2^n)$ pour énumérer tous les ensembles irredondants maximaux.

Même au niveau d'optimisation, les algorithmes de BINKELE-RAIBLE et collab. [6] sont pires que celui du problème de dominance qui s'exécute en $\mathcal{O}(1.4689^n)$ [43].

Concernant l'énumération « output-sensitive », il existe un algorithme qui énumère tous les ensembles dominants minimaux en délai quasi-polynomial [26]. Et l'existence d'un algorithme en délai polynomial reste ouverte jusqu'à aujourd'hui. Alors que BOROS [9] a prouvé que l'énumération « output-sensitive » des ensembles irredondants maximaux est « NP-complet ».

Le défi alors nous a amené à étudier ces ensembles. On a conçu, des algorithmes d'énumération des ensembles irredondants maximaux dans les graphes cordaux, les graphes d'intervalles et les forêts en $\mathcal{O}(1.7549^n)$, $\mathcal{O}(1.6957^n)$ et $\mathcal{O}(1.6181^n)$ respectivement au lieu de l'algorithme trivial en $\mathcal{O}^*(2^n)$. On a proposé aussi comme une borne inférieure une famille de forêts avec au moins 1.5292^n ensembles irredondants maximaux. Et on a prouvé que chaque ensemble irredondant maximal dans les cographes est un ensemble dominant minimal. Cela implique que le nombre maximum des ensembles irredondants maximaux dans les cographes est $15^{n/6}$ [17].

3.3 Les cographes

Lemme 7. *Chaque ensemble irredondant maximal dans un cographe est un ensemble dominant minimal.*

Démonstration. Soit G = (V, E) un cographe. On suppose, par l'absurde, qu'il existe un ensemble irredondant maximal D dans G tel que D n'est pas un ensemble dominant minimal de G. On suppose que D est un ensemble dominant dans G mais n'est pas minimal. Donc il existe un sommet $x \in D$ qui n'a pas un privé. Donc D n'est pas un ensemble irredondant ; contradiction. On suppose maintenant que D est un ensemble non dominant dans G. Donc il existe un sommet $x \in V$ tel que $N_G[x] \cap D = \emptyset$. Par la maximalité de l'ensemble irredondant D, il existe un sommet $y \in N_G(x)$ tel que y est un voisin privé pour un sommet $u \in D$, sinon x doit appartenir à l'ensemble D. Comme u n'est pas un privé pour lui-même donc il existe un sommet $v \in N_G(u) \cap D$. De plus, on a $vx \notin E$ et $ux \notin E$, sinon le sommet x sera dominé. On a aussi $vy \notin E$ car y est le voisin privé de u. Donc le sous-graphe induit $G[\{v, u, y, x\}]$ est un P_4 ; contradiction. Donc chaque ensemble irredondant maximal dans un cographe est un ensemble dominant minimal.

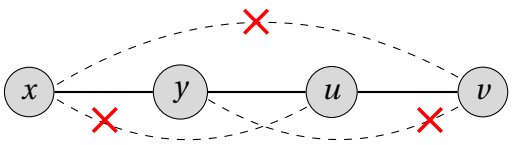

FIGURE 3.2 – $G[\{v, u, y, x\}]$

□

Le lemme 7 ainsi que la preuve de COUTURIER et collab. [17] qui montre que le nombre maximum des ensembles dominants minimaux dans les cographes de n sommets est au plus $15^{n/6}$, nous permet de déduire le théorème suivant :

Théorème 9. *Un cographe de n sommets a au plus* $15^{n/6}$ *ensembles irredondants maximaux qu'on peut énumérer en* $\mathcal{O}^*(15^{n/6})$.

De plus, l'union disjointe de l'octaèdre de la figure 2.1, qui est bien un cograph, a $15^{n/6}$ ensembles dominants minimaux [24]. Et donc il existe une famille de cographes ayant $15^{n/6}$ ensembles irredondants maximaux.

Remarque : Dans les graphes sans P_5, qui sont des super-classes de cographes, le lemme 7 n'est pas vrai. Déjà dans la figure 3.1, qui est bien un graphe sans P_5, on a des ensembles irredondants maximaux qui ne sont pas des ensembles dominants minimaux. En revanche, la cardinalité minimale d'un ensemble irredondant maximal dans un P_5 est égale à la cardinalité minimale d'un ensemble dominant minimal [58].

3.4 Les graphes cordaux

Notre algorithme s'appuie sur le théorème suivant des sommets simpliciaux d'un graphe cordal. Un théorème qui est plus compact et plus fort que le théorème 6 proposé par Faisal N.Abu-Khzam et Pinar Heggernes et que le lemme 1 du chapitre 2.

Soit G un graphe et S(G) l'ensemble des tous les sommets simpliciaux de G. Pour un sommet u, on note $S_G(u)$ l'ensemble de tous les sommets simpliciaux v de G tels que $v \in N_G(u)$. On dit que u est un sommet *semi-simplicial* si $S_G(u) \neq \emptyset$ et u est un sommet simplicial de $G - S_G(u)$.

Théorème 10. *Si* G *est un graphe cordal connexe non trivial, alors il existe un sommet semi-simplicial qui peut être trouvé en temps polynomial.*

Démonstration. Le théorème est trivial si G est un graphe complet. On suppose alors que G est un graphe non complet. Soit S l'ensemble des sommets simpliciaux de G et on note $H = G - S$. Comme G n'est pas un graphe complet, H n'est pas vide. Clairement, H est un graphe cordal. Par conséquent, H a un sommet simplicial u qui peut être trouvé en temps polynomial.

Soit $G' = G - S_G(u)$. Supposons que u n'est pas un sommet simplicial de G'. Donc u a deux voisins x et y dans G' tels que $xy \notin E(G')$. On note que $x, y \notin S$, sinon $x, y \in S_G(u)$ car x, y sont adjacents à u. Alors, $x, y \in V(H)$. Donc le sommet u a deux voisins non adjacents dans H; contradiction car u est un sommet simplicial de H. Par conséquent, u est un sommet simplicial de G'.

Supposons que $S_G(u) = \emptyset$. Alors $G' = G$ et u est un sommet simplicial de G. Par contre, $u \in V(H)$ et H a été obtenu en supprimant tous les sommets simpliciaux de G; contradiction. Par conséquent, $S_G(u) \neq \emptyset$.

Finalement, comme $S_G(u) \neq \emptyset$ et u est un sommet simplicial de $G' = G - S_G(u)$. Alors u est un sommet semi-simplicial dans G. □

3.4.1 L'algorithme d'énumération

Soit G un graphe cordal. Soit EnumIS(S,F,X) un algorithme récursif de type « Brancher et réduire », où $S, X, F \subseteq V(G)$ sont des sous-ensembles disjoints. L'algorithme EnumIS(S,F,X) énumère tous les ensembles irredondants maximaux D dans G tels que $S \subseteq D$, $D \setminus S \subseteq X$ et pour chaque $v \in D \setminus S$, v a un privé dans $F \cup X$. Afin d'énumérer tous les ensembles irredondants maximaux de G, on appelle EnumIS($\emptyset, \emptyset, V(G)$).

Soit $H = G[X \cup F]$, le graphe induit de G par $X \cup F$ tel que $F \cap D = \emptyset$. Un sommet de F ne peut donc appartenir à aucun ensemble irredondant maximal D. Pour cela, on dit qu'un sommet de H est *interdit* s'il appartient à F, alors qu'un sommet est *libre* s'il appartient à X.

On dit aussi qu'un sommet de H est *dominé* s'il est dominé dans G par un sommet de S.

À chaque étape de l'algorithme, on réduit l'instance par une règle de réduction ou on branche. On suppose toujours qu'une étape de l'algorithme est appliquée si aucune étape précédente n'est applicable.

À chaque étape, si on décide d'ajouter un sommet $u \in X$ dans un ensemble irredondant potentiel S, alors on doit s'assurer qu'il obtienne un privé v dans $F \cup X$ qui ne soit pas dominé par d'autres sommets. Pour cela, on déclare comme interdit, tous les voisins de v sauf u.

On dit qu'un ensemble irredondant maximal D est *compatible* avec le triplet (S, F, X) si les conditions suivantes sont satisfaisantes :

(i) $S \subseteq D$,

(ii) $D \setminus S \subseteq X$,

(iii) pour tout $v \in D \setminus S$, v a un privé dans $F \cup X$.,

(iv) pour tout $u \in S$, u a un privé $v \in V(G) \setminus (F \cup X)$ tel que $N_G(v) \cap X = \emptyset$.

L'idée principale de la preuve de correction de EnumIS(S, F, X) est de prouver que l'algorithme prendre en compte chaque ensemble irredondant maximal compatible avec le triplet (S, F, X) et il le produit.

EnumIS(S, F, X)

1. Si $X = \emptyset$, alors vérifiez si S est un ensemble irredondant maximal dans G et le retournez si le cas ; arrêtez.
2. Si H a un sommet interdit et dominé x, appelez EnumIS$(S, F \setminus \{x\}, X)$.
3. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont dominés, appelez EnumIS$(S, F, X \setminus \{x\})$.
4. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont interdits, appelez EnumIS$(S, F \setminus \{x\}, X)$.
5. Si H a une composante H' qui est un graphe complet, alors pour chaque sommet $x \in V(H') \cap X$, appelez EnumIS$(S \cup \{x\}, F \setminus V(H'), X \setminus V(H'))$.
6. S'il existe un sommet simplicial dominé x de H, alors branchez :
   (i) pour chaque sommet non dominé $v \in N_H(x)$,
   appelez EnumIS$(S \cup \{x\}, (F \cup N_H(v)) \setminus N_H[x], X \setminus N_H[v])$,
   (ii) appelez EnumIS$(S, F, X \setminus \{x\})$.
7. S'il existe un sommet simplicial x de H tel que $d_H(x) \geq 3$, alors branchez :
   (i) pour chaque $v \in N_H[x] \cap X$, appelez EnumIS$(S \cup \{v\}, F \setminus N_H[x], X \setminus N_H[x])$,
   (ii) appelez EnumIS$(S, F \setminus \{x\}, X \setminus \{x\})$.
8. S'il existe un sommet simplicial interdit x, alors branchez :
   (i) pour chaque $v \in N_H(x) \cap X$, appelez EnumIS$(S \cup \{v\}, F \setminus N_H[x], X \setminus N_H[x])$,
   (ii) appelez EnumIS$(S, F \setminus \{x\}, X)$.
9. S'il y a deux sommets simpliciaux adjacents x et y, alors branchez :
   (i) appelez EnumIS$(S \cup \{x\}, (F \cup N_H(x)) \setminus \{x, y\}, X \setminus N_H[x])$,
   (ii) appelez EnumIS$(S, F, X \setminus \{x\})$.

10. S'il existe un sommet simplicial x tel que $N_H(x) \subseteq F$, alors branchez :
    (i) appelez EnumIS($S \cup \{x\}, F \setminus N_H[x], X \setminus \{x\}$),
    (ii) appelez EnumIS($S, F, X \setminus \{x\}$).
11. S'il existe un sommet simplicial x tel que $N_H(x) \cap F \neq \emptyset$, alors notez $y \in N_H(x) \cap X$ et branchez :
    (i) appelez EnumIS($S \cup \{x\}, F \setminus N_H[x], X \setminus N_H[x]$),
    (ii) appelez EnumIS($S \cup \{y\}, F \setminus N_H[x], X \setminus N_H[x]$),
    (iii) appelez EnumIS($S, F, X \setminus \{x\}$).
12. S'il y a deux sommets simpliciaux x et y tels que $N_H(x) = N_H(y)$, alors branchez :
    (i) appelez EnumIS($S \cup \{x, y\}, F, X \setminus (N_H[x] \cup \{y\})$),
    (ii) appelez EnumIS($S, F \cup \{y\}, X \setminus \{x, y\}$).
13. S'il y a deux sommets simpliciaux x et y tels que $N_H(y) \subset N_H(x)$, alors notez z le voisin commun de x, y et u le voisin de x qui n'est pas adjacent à y et branchez :
    (i) appelez EnumIS($S \cup \{x, y\}, F, X \setminus (N_H[x] \cup \{y\})$),
    (ii) appelez EnumIS($S \cup \{u, y\}, F, X \setminus (N_H[x] \cup \{y\})$),
    (iii) appelez EnumIS($S \cup \{z\}, F, X \setminus (N_H[x] \cup \{y\})$),
    (iii) appelez EnumIS($S, F, X \setminus \{x\}$).
14. S'il y a un sommet semi-simplicial dominé y de H, alors branchez :
    (i) pour chaque $x \in S_H(y)$,
    appelez EnumIS($S \cup \{y\}, F \setminus (S_H(y) \cup N_H(x)), X \setminus (S_H(y) \cup N_H(x))$),
    (ii) appelez EnumIS($S, F, X \setminus \{y\}$).
15. S'il y a un sommet semi-simplicial y de H tel que $|S_H(y)| \geq 3$ et il existe $x \in S_H(y)$ avec $d_H(x) = 2$, alors notez u l'adjacent de x distinct de y et branchez :
    (i) appelez EnumIS($S \cup \{x\}, F, X \setminus \{x, u, y\}$),
    (ii) appelez EnumIS($S \cup \{y\}, F, X \setminus (S_H(y) \cup \{u\})$),
    (iii) appelez EnumIS($S \cup \{u\}, F, X \setminus \{x, u, y\}$),
    (iv) appelez EnumIS($S, F, X \setminus \{x\}$).
16. S'il y a un sommet semi-simplicial y de H tel que $|S_H(y)| = 1$ et $x \in S_H(y)$ est un pendant, alors branchez :
    (i) appelez EnumIS($S \cup \{x\}, F, X \setminus \{x, y\}$),
    (ii) appelez EnumIS($S \cup \{y\}, F, X \setminus \{x, y\}$),
    (iii) pour chaque $z \in (N_H(y) \setminus \{x\}) \cap X$, appelez EnumIS($S \cup \{z\}, F \setminus N_H[y], X \setminus N_H[y]$).
17. S'il y a un sommet semi-simplicial y de H tel que $|S_H(y)| = 1$ et pour $x \in S_H(y)$, $d_H(x) = 2$, alors notez z l'adjacent de x dans H distinct de y et branchez :
    (i) appelez EnumIS($S \cup \{x\}, F, X \setminus \{x, y, z\}$),
    (ii) appelez EnumIS($S \cup \{y\}, F, X \setminus \{x, y, z\}$),
    (iii) appelez EnumIS($S \cup \{z\}, F, X \setminus \{x, y, z\}$),
    (iv) pour chaque $v \in (N_H(y) \setminus \{x, z\}) \cap X$, appelez EnumIS($S \cup \{v\}, F \setminus N_H[y], X \setminus N_H[y]$),
    (v) appelez EnumIS($S, F \cup \{z\}, X \setminus \{x, y, z\}$).

18. S'il y a un sommet semi-simplicial y de H tel que $|S_H(y)| = 2$, $N_H(x) \neq N_H(z)$ pour distinct $x, z \in S_H(y)$ et $d_H(x) = d_H(z) = 2$, alors notez u et v les voisins de x et z respectivement qui sont distincts de y et branchez :
    (i) appelez ENUMIS$(S \cup \{x\}, F, X \setminus \{x, u, y\})$,
    (ii) appelez ENUMIS$(S \cup \{u, v\}, F, X \setminus \{x, u, y, v, z\})$,
    (iii) appelez ENUMIS$(S \cup \{u, z\}, F, X \setminus \{x, u, y, v, z\})$,
    (iv) appelez ENUMIS$(S \cup \{y\}, F, X \setminus \{x, u, y, z\})$,
    (v) appelez ENUMIS$(S, F, X \setminus \{x\})$.

La preuve de correction est basée sur le lemme suivant.

Lemme 8. *Soit* D *un ensemble irredondant maximal du graphe* G *compatible avec un triplet* (S, F, X) *des sous-ensembles disjoints de* $V(G)$. *Si l'une des conditions des étapes 1–18 est satisfaite, alors soit* $D = S$ *et l'algorithme retourne* D*, soit l'algorithme ne s'arrête pas et il y a un appel récursif de* ENUMIS(S', F', X') *dans l'appel de* ENUMIS(S, F, X) *tel que* D *est compatible avec* (S', X', F').

Démonstration.

On considère les 18 cas correspondant aux étapes 1–18 de l'algorithme. On a illustré, au travers des figures explicatives, quelques cas en représentant, si possible de le déterminer, un sommet dominé en bleu et un sommet interdit en rouge.

Cas 1. L'ensemble $X = \emptyset$.

On considère la première étape qui peut être appliquée. Si $X = \emptyset$, alors $S = D$, car $D \setminus S \subseteq X$, et l'algorithme retourne D.

Dans les 3 prochaines étapes, on réduit l'instance d'entrée.

Cas 2. Si H a un sommet x interdit et dominé.

Comme x est interdit, alors $x \notin D$. De plus x est dominé par S. Donc il ne peut pas être un privé pour aucun sommet de $D \setminus S$. Par conséquent, D est compatible avec $(S, F \setminus \{x\}, X)$.

Cas 3. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont dominés.

Le sommet $x \in X$, car l'étape 2 n'est pas applicable. On note également que $x \notin D$, sinon il n'aurait pas un privé dans $F \cup X$. De plus, x ne peut pas être un privé pour aucun sommet de $D \setminus S$, car x est dominé. Par conséquent, D est compatible avec $(S, F, X \setminus \{x\})$.

Cas 4. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont interdits.

Clairement, $x \notin D$ parce qu'il est interdit. En outre, il ne peut pas être un privé pour aucun sommet de $D \setminus S$, car $N_G[x] \cap (D \setminus S) = \emptyset$. Par conséquent, D est compatible avec $(S, F \setminus \{x\}, X)$.

A partir de maintenant on branche. De plus on suppose, dans le reste des cas, que chaque sommet dominé est libre et que chaque sommet interdit n'est pas dominé.

Cas 5. Si H a une composante H' qui est un graphe complet.

La composante H' a au moins un sommet libre à cause de l'étape 4. De même H' a des sommets non dominés à cause de l'étape 3. Par maximalité, $D \cap V(H') \neq \emptyset$. Si $|V(H') \cap D| \geq 2$, alors pour tout $x, y \in V(H') \cap D$, on a $N_H[x] = N_H[y]$ et x, y n'ont pas de privés dans $F \cup X$; contradiction. Alors, D a un sommet unique $x \in V(H')$. Par conséquent, D est compatible avec $(S \cup \{x\}, F \setminus V(H'), X \setminus V(H'))$.

Cas 6. S'il existe un sommet simplicial dominé x de H.

Si $x \in D$, alors il a un privé $v \in (F \cup X)$. Puisque x est dominé, alors v est un sommet non dominé de $N_H(x)$. Alors $(N_H[v] \setminus \{x\}) \cap D = \emptyset$. Comme x est un sommet simplicial de H, on a $N_H(x) \cap D = \emptyset$. Par conséquent, D est compatible avec $(S \cup \{x\}, (F \cup N_H(v)) \setminus N_H[x], X \setminus N_H[v])$. Si $x \notin D$, alors, car x est dominé, D est compatible avec $(S, F, X \setminus \{x\})$.

On suppose, à partir de maintenant, que tous les simpliciaux de H ne sont pas dominés.

Cas 7. S'il existe un sommet simplicial x de H tel que $d_H(x) \geq 3$.

On suppose que $x \in D$. Alors, $x \in X$ dans ce cas et $N_H(x) \cap D = \emptyset$, car sinon x n'a pas un privé. De plus, les sommets $N_H(x)$ ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{x\})$. Par conséquent, D est compatible avec $(S \cup \{v\}, F \setminus N_H[x], X \setminus N_H[x])$ pour $v = x$.

On suppose maintenant que $x \notin D$ et $N_H(x) \cap D = \{v\}$ pour un sommet v. On rappelle que x n'est pas dominé. Cela signifie que x est un privé pour v. Puisque les sommets $N_H[x]$ ne peuvent pas être des privés pour les sommets $D \setminus (S \cup \{v\})$, par conséquent, D est compatible avec $(S \cup \{v\}, F \setminus N_H[x], X \setminus N_H[x])$. Enfin, on suppose que $x \notin D$ et $|N_H(x) \cap D| \neq 1$. Alors x n'est pas un privé pour aucun sommet de D. Par conséquent, D est compatible avec $(S, F \setminus \{x\}, X \setminus \{x\})$.

A partir de maintenant, on suppose que chaque sommet simplicial est de degré 1 ou 2.

Cas 8. S'il existe un sommet simplicial interdit x.

On suppose que $N_H(x) \cap D = \{v\}$ pour un sommet v. Comme x n'est pas dominé, alors x est un privé pour v. Puisque les sommets $N_H[x]$ ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{v\})$, D est compatible avec $(S \cup \{v\}, F \setminus N_H[x], X \setminus N_H[x])$. On suppose maintenant que $|N_H(x) \cap D| \neq 1$. Alors x n'est pas un privé pour aucun sommet de D. Par conséquent, D est compatible avec $(S, F \setminus \{x\}, X)$.

A partir de maintenant, tous les sommets simpliciaux sont libres.

Cas 9. S'il y a deux sommets simpliciaux adjacents x et y.

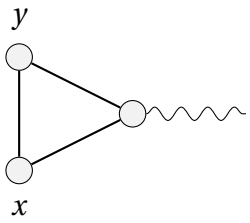

FIGURE 3.3 – Cas 9

Si $x \in D$, alors $D \cap N_H(x) = \emptyset$ et x est un privé pour lui-même. De plus, le sommet y ne peut pas être un privé pour aucun sommet de $D \setminus (S \cup \{x\})$. Par conséquent, D est compatible avec $(S \cup \{x\}, (F \cup N_H(x)) \setminus \{x, y\}, X \setminus N_H[x])$.

On suppose que $x \notin D$. On suppose que x est un privé pour un sommet $v \in D$. Puisque $N_H[x] = N_H[y]$ et y n'est pas dominé, alors y est un privé pour v. Par conséquent, D est compatible avec $(S, F, X \setminus \{x\})$.

A partir de maintenant, toute paire de sommets simpliciaux distincts de H est non adjacente.

Cas 10. S'il existe un sommet simplicial x tel que $N_H(x) \subseteq F$.

Si $x \in D$, alors $D \cap N_H(x) = \emptyset$ et x est un privé pour lui-même. De plus, les sommets $N_H(x)$ ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{x\})$. Alors, D est compatible avec $(S \cup \{x\}, F \setminus N_H[x], X \setminus \{x\})$. Si $x \notin D$, alors x n'est pas un privé pour aucun sommet de $D \setminus S$, car les voisins de x sont interdits. Par conséquent, D est compatible avec $(S, F, X \setminus \{x\})$.

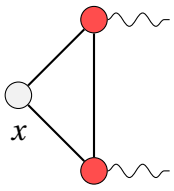

FIGURE 3.4 – Cas 10

Cas 11. S'il existe un sommet simplicial x tel que $N_H(x) \cap F \neq \emptyset$.

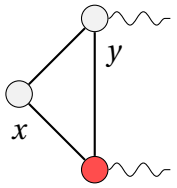

FIGURE 3.5 – Cas 11

Comme l'étape 10 est non applicable, alors x a un voisin libre. Soit $y \in N_H(x) \cap X$. De plus, $d_H(x) = 2$ à cause de l'étape 7.

Si $x \in D$, alors $D \cap N_H(x) = \emptyset$ et x est un privé pour lui-même. De plus, les sommets $N_H(x)$ ne peuvent pas être des privés pour aucun sommet de $D \setminus (S \cup \{x\})$. Par conséquent, D est compatible avec $(S \cup \{x\}, F \setminus N_H[x], X \setminus N_H[x])$.

Si $x \notin D$ et $y \in D$. Dans ce cas on a $(D \cap N_H[x]) \setminus \{y\} = \emptyset$, l'autre voisin de x est interdit et x est un privé pour y. De plus, les sommets de $N_H[x]$ ne peuvent pas être des privés pour aucun sommet de $D \setminus (S \cup \{y\})$. Par conséquent, D est compatible avec $(S \cup \{y\}, F \setminus N_H[x], X \setminus N_H[x])$.

Si $x, y \notin D$, alors x n'est pas un privé pour aucun sommet de $D \setminus S$, car le voisin de x distinct de y est interdit. Par conséquent, D est compatible avec $(S, F, X \setminus \{x\})$.

Dans ce qu'il reste des cas, tous les voisins des sommets simpliciaux sont libres.

Cas 12. S'il y a deux sommets simpliciaux x et y tels que $N_H(x) = N_H(y)$.

Les sommets x et y sont libres et non dominés.

Si $x \in D$, alors x est un privé pour lui-même, $(N_H(x) \cap D) = \emptyset$ et les sommets $N_H(x)$ ne sont pas des privés pour aucun sommet de $D \setminus (S \cup \{x\})$. Dans ce cas, le sommet $y \in D$ aussi, par la maximalité de D, et y est un privé pour lui-même. Par symétrie, on peut déduire que $x \in D$ si et seulement si $y \in D$. Par conséquent, D est compatible avec $(S \cup \{x, y\}, F, X \setminus (N_H[x] \cup \{y\}))$.

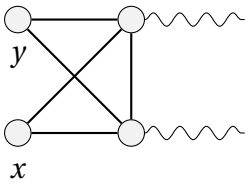

FIGURE 3.6 – Cas 12

On suppose que $x, y \notin D$. Si x est un privé pour un sommet $v \in D$, alors y est son privé aussi. Par conséquent, D est compatible avec $(S, F \cup \{y\}, X \setminus \{x, y\})$.

Cas 13. S'il y a deux sommets simpliciaux x et y tels que $N_H(y) \subset N_H(x)$.

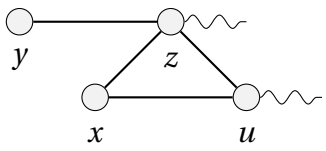

FIGURE 3.7 – Cas 13

Comme $d_H(x), d_H(y) \leq 2$, alors $d_H(x) = 2$ et $d_H(y) = 1$. On note z le voisin commun de x et y. On note également u le voisin de x qui n'est pas adjacent à y. Les sommets x, y, z et u sont libres. De plus, les deux sommets x et y ne sont pas dominés.

On suppose que $x \in D$.

Alors, le sommet x dans ce cas est un privé pour lui-même, $(N_H(x) \cap D) = \emptyset$ et les sommets $N_H(x)$ ne sont pas des privés pour aucun sommet de $D \setminus (S \cup \{x\})$. Par la maximalité de D, le sommet $y \in D$ aussi, et il est un privé pour lui-même. Par conséquent, D est compatible avec $(S \cup \{x, y\}, F, X \setminus (N_H[x] \cup \{y\}))$.

On suppose que $x \notin D$ et que $|N_H(x) \cap D| = 1$.

Soit $u \in D$ et $z \notin D$. Alors, dans ce cas, le sommet x est un privé pour u. Par la maximalité de D, le sommet $y \in D$ aussi. Par conséquent, D est compatible avec $(S \cup \{u, y\}, F, X \setminus (N_H[x] \cup \{y\}))$.

Soit $u \notin D$ et $z \in D$. Alors x est un privé pour z, et y et les sommets $N_H[x]$ ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{z\})$. Par conséquent, D est compatible avec $(S \cup \{z\}, F, X \setminus (N_H[x] \cup \{y\}))$.

Finalement, on suppose que $x \notin D$ et $|N_H(x) \cap D| \neq 1$.

Alors x ne peut pas être un privé pour aucun sommet de $D \setminus S$. Par conséquent D est compatible avec $(S, F, X \setminus \{x\})$.

Dans les étapes restantes de l'algorithme, on utilise les sommets semi-simpliciaux pour brancher. On rappelle que chaque sommet simplicial de H n'est pas dominé, libre, de degré 1 ou 2 et tous ses voisins sont libres. De plus, deux sommets simpliciaux distincts ne sont pas adjacents et leurs voisins ne sont pas comparables par inclusion.

Cas 14. S'il y a un sommet semi-simplicial dominé y de H.

On suppose que $y \in D$. Dans ce cas, $S_H(y) \cap D = \emptyset$, sinon un sommet $v \in S_H(y) \cap D$ n'a pas de privé, car $N_H[v] \subseteq N_H[y]$.

On montre qu'il y a $x \in S_H(y)$ qui est un privé pour y. Si ce n'est pas le cas, alors les sommets $S_H(y)$ sont dominés par des sommets de $D \setminus (S \cup \{y\})$, et en particulier par des sommets de $N_H(y) \setminus S_H(y)$. Cela implique que y n'a pas de privé ; contradiction.

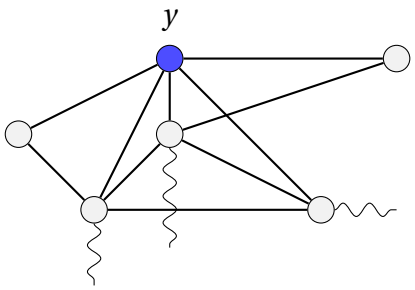

FIGURE 3.8 – Cas 14

Soit $x \in S_H(y)$ un privé de $y \in D$. Alors les sommets $N_H[x] \setminus \{y\}$ ne sont pas dans D et ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{y\})$.

Par conséquent, D est compatible avec $(S \cup \{y\}, F \setminus (S_H(y) \cup N_H(x)), X \setminus (S_H(y) \cup N_H(x)))$.

Si $y \notin D$, alors D est compatible avec $(S, F, X \setminus \{y\})$.

On suppose, à partir de maintenant, que tous les sommets semi-simpliciaux de H ne sont pas dominés.

Cas 15. S'il y a un sommet semi-simplicial y de H tel que $|S_H(y)| \geq 3$ et il existe $x \in S_H(y)$ avec $d_H(x) = 2$.

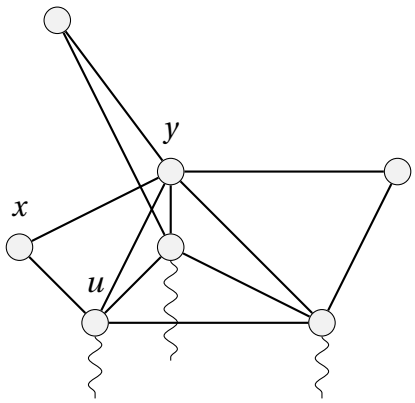

FIGURE 3.9 – Cas 15

Les sommets $S_H(y)$ et leurs voisins sont libres et le sommet x n'est pas dominé. Puisque $|S_H(y)| \geq 3$ et les sommets simpliciaux $S_H(y)$ ont des voisins incomparables par inclusion, alors on a un sommet $x \in S_H(y)$ de degré 2. On note u l'adjacent de x distinct de y. On note également que $u \notin S_H(y)$, car H n'a pas de sommets simpliciaux adjacents.

Si $x \in D$, alors x est un privé pour lui-même, $y, u \notin D$ et ils ne peuvent pas être des privés pour aucun sommet de $(D \setminus (S \cup \{x\}))$. Par conséquent, D est compatible avec $(S \cup \{x\}, F, X \setminus \{x, y, u\})$.

Soit $x \notin D$. Si $y \in D$ et $u \notin D$, alors x est un privé pour y. On a $S_H(y) \cap D = \emptyset$, sinon un sommet de $D \cap S_H(y)$ n'a pas un privé dans $F \cup X$. De plus, les sommets $S_H(y) \cup \{u\}$ ne peuvent pas être des privés pour les sommets de $(D \setminus (S \cup \{y\}))$, car ils sont dominés par y. Par conséquent, D est compatible avec $(S \cup \{y\}, F, X \setminus (S_H(y) \cup \{u\}))$.

Sinon, si $y \notin D$ et $u \in D$, alors x est un privé pour u. De plus, x et y ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{u\})$. Par conséquent, D est compatible avec $(S \cup \{u\}, F, X \setminus \{x, u, y\})$. Finalement, si $y, u \in D$ ou $y, u \notin D$, alors D est compatible avec $(S, F, X \setminus \{x\})$, car x ne peut pas être un privé pour aucun sommet de $D \setminus S$.

Pour chaque sommet semi-simplicial y de H, $|S_H(y)| \leq 2$. Dans les cas suivants, on branche sur un semi-simplicial y tel que $|S_H(y)| = 1$.

Cas 16. S'il y a un sommet semi-simplicial y de H tel que $|S_H(y)| = 1$ et $x \in S_H(y)$ est un pendant.

Les sommets x et y sont libres et non dominés. On note également que $N_H(y) \setminus \{x\} \neq \emptyset$, à cause de l'étape 5.

Si $x \in D$, alors x est un privé pour lui-même, $y \notin D$ et il ne peut pas être un privé pour aucun sommet de $D \setminus (S \cup \{x\})$. Par conséquent, D est compatible avec $(S \cup \{x\}, F, X \setminus \{x, y\})$.

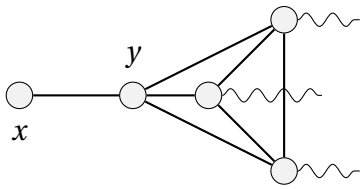

FIGURE 3.10 – Cas 16

Soit $x \notin D$. Si $y \in D$, alors x est un privé pour y. Par conséquent, D est compatible avec $(S \cup \{y\}, F, X \setminus \{x, y\})$. On suppose que $x, y \notin D$. On suppose que $D \cap N_H(y) = \{z\}$ pour un $z \in N_H(y) \setminus \{x\}$. Alors y est un privé pour z. De plus, les sommets $N_H[y]$ ne peuvent pas être des privés pour les sommets de $D \setminus (S \cup \{z\})$. Cela implique que D est compatible avec $(S \cup \{z\}, F \setminus N_H[y], X \setminus N_H[y])$.

Finalement, on suppose que $x, y \notin D$ et $|N_H(y) \cap D| \neq 1$. Puisque $|N_H(y) \cap D| \neq 1$, alors le sommet y ne peut pas être un privé pour aucun sommet de $D \setminus S$. Donc par la maximalité de l'ensemble irredondant D, le sommet x doit être dans D et x dans ce cas est un privé pour lui-même. Or $x \notin D$; contradiction. Par conséquent, le dernier cas ne se produit pas.

Cas 17. S'il y a un sommet semi-simplicial y de H tel que $|S_H(y)| = 1$ et pour $x \in S_H(y)$, $d_H(x) = 2$.

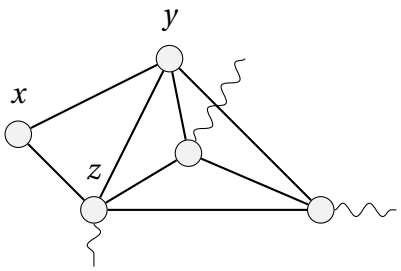

FIGURE 3.11 – Cas 17

On note z l'adjacent de x dans H distinct de y. Les sommets x, y et z sont libres et les sommets x et y ne sont pas dominés. On note également que $N_H(y) \setminus \{x, z\} \neq \emptyset$, à cause de l'étape 9.

On suppose que $x \in D$. Alors x est un privé pour lui-même, $y, z \notin D$ et ils ne peuvent pas être des privés pour aucun sommet de $D \setminus (S \cup \{x\})$. Par conséquent, D est compatible avec $(S \cup \{x\}, F, X \setminus \{x, y, z\})$.

On suppose que $x \notin D$. Si $y \in D$, alors x est un privé pour y et $z \notin D$, car $N_H[y] \subseteq N_H[z]$. De plus, z ne peut pas être un privé pour aucun sommet de $D \setminus (S \cup \{y\})$. Par conséquent, D est compatible avec $(S \cup \{y\}, F, X \setminus \{x, y, z\})$.

Soit $x, y \notin D$. Si $z \in D$, alors x est un privé pour z. Aussi y ne peut pas être un privé pour aucun sommet de $D \setminus (S \cup \{z\})$. Par conséquent, D est compatible avec $(S \cup \{z\}, F, X \setminus \{x, y, z\})$.

On suppose maintenant que $x, y, z \notin D$ mais $N_H(y) \cap D = \{v\}$ pour un sommet $v \in (N_H(y) \setminus \{x, z\}) \cap X$. Alors y est un privé pour v. Clairement, les sommets $N_H[y]$ ne peuvent pas être des privés pour aucun sommet de $D \setminus (S \cup \{v\})$. Par conséquent, D est compatible avec $(S \cup \{v\}, F \setminus N_H[y], X \setminus N_H[y])$.

Finalement, on suppose que $x, y, z \notin D$ et $|N_H(y) \cap D| \neq 1$. Alors x et y ne peuvent pas être des privés pour aucun sommet de $D \setminus S$. Par conséquent, D est compatible avec $(S, F \cup \{z\}, X \setminus \{x, y, z\})$.

Enfin, il ne reste que des sommets semi-simpliciaux y avec $|S_H(y)| = 2$ tels que les deux simpliciaux de $S_H(y)$ ne sont pas adjacents et ont des voisins incomparables par inclusion.

Cas 18. S'il y a un sommet semi-simplicial y de H tel que $|S_H(y)| = 2$, $N_H(x) \neq N_H(z)$ pour distinct $x, z \in S_H(y)$ et $d_H(x) = d_H(z) = 2$.

On note u et v les voisins de x et z respectivement qui sont distincts de y. Les sommets x, y, z, u et v sont libres. De plus, les sommets x, y et z ne sont pas dominés.

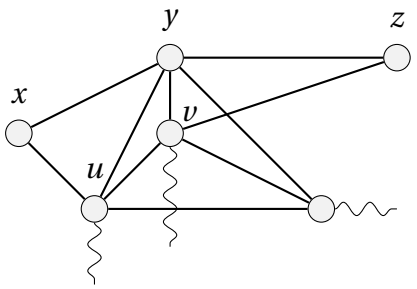

FIGURE 3.12 – Cas 18

Si $x \in D$, alors $y, u \notin D$ et ces sommets ne peuvent être des privés pour aucun sommet de $D \setminus (S \cup \{x\})$. Aussi x est un privé pour lui-même. Par conséquent, D est compatible avec $(S \cup \{x\}, F, X \setminus \{x, u, y\})$.

On suppose que $x \notin D$. Si $u, v \in D$, alors x et z sont leurs privés. De plus, $y \notin D$ et il ne peut pas être un privé pour aucun sommet de $D \setminus (S \cup \{u, v\})$. Par conséquent, D est compatible avec $(S \cup \{u, v\}, F, X \setminus \{x, u, y, v, z\})$.

Soit $u \in D$ et $y, v \notin D$. Dans ce cas, x est un privé pour u. Et par la maximalité de l'ensemble irredondant D, le sommet z doit être dans D et z dans ce cas est un privé pour lui-même. De plus, le sommet y ne peut pas être un privé pour aucun sommet de $D \setminus (S \cup \{u, z\})$. Par conséquent, D est compatible avec $(S \cup \{u, z\}, F, X \setminus \{x, u, y, v, z\})$.

Soit maintenant $u \notin D$ et $y \in D$. Alors x est un privé pour y, les sommets $u, z \notin D$ et ils ne peuvent pas être des privés pour aucun sommet de $D \setminus (S \cup \{y\})$. Par conséquent, D est compatible avec $(S \cup \{y\}, F, X \setminus \{x, u, y, z\})$.

Finalement, si $y, u \in D$ ou $y, u \notin D$, x ne peut pas être un privé pour aucun sommet de $D \setminus S$ et D est compatible avec $(S, F, X \setminus \{x\})$.

□

On a donné dans cette section une desciption détaillée de l'algorithme et on donne dans la section suivante une preuve de sa correction.

3.4.2 La correction de l'algorithme

Afin de montrer que l'algorithme est correct, on prouve tout d'abord que les étapes de l'algorithme sont exhaustives.

Lemme 9. *Soit* $S, F, X \subseteq V(G)$ *des sous-ensembles disjoints, alors au moins une étape de* ENUMIS(S, F, X) *est applicable.*

Démonstration. On observe que l'analyse des cas de l'algorithme est exhaustive, c'est-à-dire au moins une étape de ENUMIS(S, F, X) peut être appliquée pour une instance (S, F, X). On note que si les étapes 1–13 de l'algorithme ne peuvent pas être appliquées, alors chaque sommet simplicial de H n'est pas dominé, libre, de degré 1 ou 2 et tous ses voisins sont libres. De plus, deux sommets simpliciaux ne sont pas adjacents et leurs voisins sont incomparables par inclusion. On note également qu'à cause de l'étape 5, toute composante connexe de H a au moins 2 sommets. Alors, par le théorème 10, chaque composante de H a au moins un sommet semi-simplicial. Par conséquent, l'une des étapes 14–18 peut être appliquée.

□

On termine maintenant la correction de l'algorithme.

A chaque étape de l'algorithme ENUMIS(S, F, X), soit on produit une solution S à l'étape 1, soit on appelle ENUMIS(S', F', X') tel que $|X'|+|F'| < |X|+|F|$. Aussi, par le lemme 9, les étapes de l'algorithme sont exhaustives. Donc l'algorithme s'arrête toujours. Afin

d'énumérer tous les ensembles irredondants maximaux dans un graphe cordal G, il suffit d'appeler ENUMIS$(\emptyset,\emptyset,V(G))$. L'algorithme génère des ensembles seulement à l'étape 1 et il affiche, après une vérification en temps polynomial, un ensemble s'il est uniquement un ensemble irredondant maximal. Par conséquent, ENUMIS$(\emptyset,\emptyset,V(G))$ ne produit que des ensembles irredondants maximaux. On montre alors que ENUMIS$(\emptyset,\emptyset,V(G))$ produit chaque ensemble irredondant maximal. Soit D un ensemble irredondant maximal. On montre, par récurrence sur $|X|+|F|$, que si D est compatible avec un triplet (S,F,X) des sous-ensembles disjoints de V(G), alors ENUMIS(S,F,X) produit D. Si $|X|+|F|=0$, alors c'est trivial, car $D=S$ dans ce cas et on produit D à l'étape 1. On suppose que l'affirmation est vraie pour $|X''|+|F''|<|X|+|F|$. Et on le prouve pour $|X|+|F|$. Par le lemme 9, on appelle ENUMIS(S',F',X') dans l'une des étapes de l'algorithme. Et par le lemme 8, D est compatible avec (S',F',X'). Comme $|X'|+|F'|<|X|+|F|$, alors par l'hypothèse de récurrence ENUMIS(S',F',X') produit D. Donc si D est compatible avec un triplet (S,F,X) des sous-ensembles disjoints de V(G), alors ENUMIS(S,F,X) produit D. De plus, tout ensemble irredondant maximal D est compatible avec $(\emptyset,\emptyset,V(G))$. Donc ENUMIS$(\emptyset,\emptyset,V(G))$ énumère tous les ensembles irredondants maximaux du graphe G.

Finalement, il reste à analyser le temps d'exécution de l'algorithme.

3.4.3 L'analyse du temps d'exécution

Afin d'analyser le temps d'exécution, on associe à chaque règle de branchement de l'algorithme, un vecteur de branchement où la mesure d'une instance (S,F,X) est définie par la somme $|F|+|X|=|V(H)|$.

Notant que tous les étapes 1–4 sont des règles de réduction.
Donc il suffit d'analyser les étapes 5–18.

Etape 5.

Le facteur du vecteur de branchement $(\underbrace{t,\ldots,t}_{t})$ où $t=|V(H')|$ atteint son maximum si $V(H')\subseteq X$. La valeur maximale est atteinte alors pour $t=3$ et vaut $3^{1/3}<1.4423$ (voir 2.2.3).

Etape 6.

Dans le pire des cas, tous les sommets $N_H(x)$ ne sont pas dominés. Alors le vecteur de branchement est $(\underbrace{t+1,\ldots,t+1}_{t},1)$ avec $t=d_H(x)$. La valeur maximale de son facteur est atteinte si $t=2$ qui est inférieure à 1.6957.

Etape 7.

Le facteur du vecteur de branchement $(\underbrace{t,\ldots,t}_{t},1)$, pour $t=d_H(x)+1\geq 4$, atteint son maximum si $N_H[x]\subseteq X$. La valeur maximale est atteinte alors pour $t=4$ et elle est inférieure à 1.7485.

Etape 8.

Le facteur du vecteur de branchement $(\underbrace{t+1,\ldots,t+1}_{t},1)$, pour $t=d_H(x)$, atteint son maximum si $N_H(x)\subseteq X$. La valeur maximale est atteinte alors pour $t=2$ et elle est inférieure à 1.6957.

Etape 9.

Le vecteur de branchement est $(2,1)$ et son facteur est inférieur à 1.6181.

Etape 10.

Comme $d_H(x) \leq 2$, le vecteur de branchement est $(3,1)$ ou $(2,1)$ et le maximum de deux facteurs est inférieur à 1.6181.

Etape 11.

Comme $d_H(x) = 2$, alors le vecteur de branchement est $(3,3,1)$ et son facteur est inférieur à 1.6957.

Etape 12.

Comme $d_H(x) \leq 2$, le vecteur de branchement est $(3,1)$ ou $(2,1)$ et le maximum de deux facteurs est inférieur à 1.6181.

Etape 13.

Le vecteur de branchement est $(4,4,4,1)$ et son facteur est inférieur à 1.6581.

Etape 14.

Soit $t = |S_H(y)|$.

Si $t = 1$, alors pour l'unique $x \in S_H(y)$, $d_H(x) \leq 2$. Le maximum facteur de branchement est atteint lorsque $d_H(x) = 1$. Le vecteur de branchement dans ce cas est $(2,1)$ et son facteur est inférieur à 1.6181.

Soit $t \geq 2$. Comme les adjacents de $S_H(y)$ sont incomparables par inclusion, alors $d_H(x) = 2$ pour tout $x \in S_H(y)$ et le vecteur de branchement est $(\underbrace{t+2,\ldots,t+2}_{t},1)$. La valeur maximale est atteinte alors pour $t = 2$ et elle est inférieure à 1.5437.

Etape 15.

Le vecteur de branchement est $(3,t+2,3,1)$, pour $t = |S_H(y)| \geq 3$. Son facteur atteint son maximum lorsque $t = 3$ avec une valeur inférieure à 1.7549.

Etape 16.

Le facteur du vecteur de branchement $(2,2,\underbrace{t+2,\ldots,t+2}_{t})$ où $t = d_H(y) - 1$ atteint son maximum si $N_H(y) \setminus \{x\} \subseteq X$. La valeur maximale est atteinte alors pour $t = 2$ et elle est inférieure à 1.6529.

Etape 17.

Le facteur du vecteur de branchement $(3,3,3,\underbrace{t+3,\ldots,t+3}_{t},2)$ où $t = d_H(y) - 2$ atteint son maximum si $N_H(y) \setminus \{x,z\} \subseteq X$. La valeur maximale est atteinte alors pour $t = 2$ et elle est inférieure à 1.7549.

Etape 18.

Le vecteur de branchement est $(3,5,5,4,1)$ et sa valeur est inférieure à 1.7393.

L'analyse du temps d'exécution montre que le pire facteur de branchement est inférieur strictement à 1.7549. Donc on peut déduire que l'algorithme s'exécute en $\mathcal{O}(1.7549)$, ainsi que le théorème suivant :

Théorème 11. *Un graphe cordal de n sommets a au plus* 1.7549^n *ensembles irredondants maximaux qu'on peut énumérer en* $\mathcal{O}(1.7549^n)$.

Dans la section suivante, on a établi un algorithme d'énumération des ensembles irredondants maximaux dans les forêts qui sont des sous-classes de graphes cordaux.

3.5 Les forêts

On montre dans cette section, que l'algorithme d'énumération et la borne supérieure des ensembles irredondants maximaux pour les graphes cordaux peuvent être améliorés dans le cas des forêts, ainsi que les arbres.

On dit qu'un graphe est une forêt s'il est sans cycles. Une forêt est dite un arbre si elle est connexe. Un sommet dans un arbre est dit une feuille s'il est de degré 1. Un arbre a au moins deux feuilles (DIRAC [21]).

Clairement, une forêt est un graphe cordal. Par conséquent, notre algorithme d'énumération dans les forêts est une version simplifiée de l'algorithme ENUMIS des graphes cordaux présenté dans la section précédente.

Dans ENUMIS, on a branché principalement sur les sommets simpliciaux et semi-simpliciaux. Or dans une forêt, tout sommet simplicial est une feuille ou un sommet isolé. Par conséquent, on peut éliminer de ENUMIS toutes les étapes dans lesquelles on a branché sur des sommets simpliciaux de degré strictement supérieur à 1. De plus, un sommet u dans une forêt est un semi-simplicial s'il a au moins une feuille adjacente et au plus un voisin non feuille.

3.5.1 L'algorithme d'énumération

Soit T une forêt. Soit ENUMIS-F(S,F,X) un algorithme récursif de type « Brancher et réduire », où $S,X,F \subseteq V(T)$ sont des sous-ensembles disjoints. L'algorithme ENUMIS-F(S,F,X) énumère tous les ensembles irredondants maximaux D dans T tels que $S \subseteq D$, $D \setminus S \subseteq X$ et pour chaque $v \in D \setminus S$, v a un privé dans $F \cup X$.

Soit $H = T[X \cup F]$, le graphe induit de T par $X \cup F$ tel que $F \cap D = \emptyset$. On dit qu'un sommet est *interdit* s'il appartient à F et il ne peut pas être inclus dans un ensemble irredondant maximal D alors qu'un sommet est *libre* s'il appartient à X. On dit aussi qu'un sommet de H est *dominé* s'il est dominé dans T par un sommet de S. À chaque étape de l'algorithme, on réduit l'instance par une règle de réduction ou on branche. On suppose toujours qu'une étape de l'algorithme est appliquée si aucune étape précédente n'est applicable. À chaque étape, si on décide d'ajouter un sommet $u \in X$ dans un ensemble irredondant potentiel S, alors on doit s'assurer qu'il obtienne un privé v dans $F \cup X$ qui ne soit pas dominé par d'autres sommets. Pour cela, on déclare comme interdit, tous les voisins de v sauf u. D'une façon formelle, pour tout $u \in S$, u a un privé $v \in V(G) \setminus (F \cup X)$ tel que $N_G(v) \cap X = \emptyset$.

Afin d'énumérer tous les ensembles irredondants maximaux de T, on appelle ENUMIS-F($\emptyset, \emptyset, V(G)$).

ENUMIS-F(S,F,X)

1. Si $X = \emptyset$, alors vérifiez si S est un ensemble irredondant maximal dans T et le retournez si le cas ; arrêtez.
2. Si H a un sommet interdit et dominé x, appelez ENUMIS-F($S, F \setminus \{x\}, X$).
3. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont dominés, appelez ENUMIS-F($S, F, X \setminus \{x\}$).
4. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont interdits, appelez ENUMIS-F($S, F \setminus \{x\}, X$).
5. Si H a une composante H' qui est un sommet isolé ou une arête, alors pour chaque sommet $x \in V(H') \cap X$, appelez ENUMIS-F($S \cup \{x\}, F \setminus V(H'), X \setminus V(H')$).

6. S'il y a une feuille x de H qui est adjacente à un sommet y tel que y est adjacent à une feuille z distincte de x, alors branchez selon les statuts des x, y et z.

   Si le sommet x ou z, soit x, est dominé, alors branchez :

   (i) appelez ENUMIS-F$(S \cup \{x\}, (F \cup N_H(y)) \setminus \{x,y,z\}, X \setminus (N_H(y) \cup \{x,y,z\}))$,

   (ii) appelez ENUMIS-F$(S, F \setminus \{x\}, X \setminus \{x\})$.

   Si les deux sommets x et z ne sont pas dominés, mais l'un de deux, soit x, est interdit, alors branchez :

   (i) appelez ENUMIS-F$(S \cup \{y\}, F \setminus \{x,y,z\}, X \setminus \{x,y,z\})$,

   (ii) appelez ENUMIS-F$(S, (F \cup \{y\}) \setminus \{x\}, X \setminus \{x,y\})$.

   Si les deux sommets x et z sont libres non dominés et y est interdit, alors branchez :

   (i) appelez ENUMIS-F$(S \cup \{x,z\}, F \setminus \{x,y,z\}, X \setminus \{x,y,z\})$,

   (ii) appelez ENUMIS-F$(S, F \setminus \{x,z\}, X \setminus \{x,z\})$.

   Si les deux sommets x et z sont libres non dominés et y est libre, alors branchez :

   (i) appelez ENUMIS-F$(S \cup \{x,z\}, F \setminus \{x,y,z\}, X \setminus \{x,y,z\})$,

   (ii) appelez ENUMIS-F$(S \cup \{y\}, F \setminus \{x,y,z\}, X \setminus \{x,y,z\})$,

   (iii) appelez ENUMIS-F$(S, (F \cup \{y\}) \setminus \{x,z\}, X \setminus \{x,y,z\})$.

7. S'il y a une feuille x de H qui est adjacente à un sommet y tel que y est une feuille de $H - x$, alors branchez selon les statuts des x et y.

   Si x est dominé, alors branchez :

   (i) appelez ENUMIS-F$(S \cup \{x\}, (F \cup N_H(y)) \setminus \{x,y\}, X \setminus \{x,y\})$,

   (ii) appelez ENUMIS-F$(S, F \setminus \{x\}, X \setminus \{x\})$.

   Si x est interdit, alors branchez :

   (i) appelez ENUMIS-F$(S \cup \{y\}, F \setminus \{x,y\}, X \setminus \{x,y\})$,

   (ii) appelez ENUMIS-F$(S, (F \cup \{y\}) \setminus \{x\}, X \setminus \{x,y\})$.

   Si x est libre non dominé et y est interdit, alors branchez :

   (i) appelez ENUMIS-F$(S \cup \{x\}, F \setminus \{x,y\}, X \setminus \{x,y\})$,

   (ii) appelez ENUMIS-F$(S, F \setminus \{x\}, X \setminus \{x\})$.

   Si x et y sont libres et x est non dominé, alors branchez :

   (i) appelez ENUMIS-F$(S \cup \{x\}, F \setminus \{x,y\}, X \setminus \{x,y\})$,

   (ii) appelez ENUMIS-F$(S \cup \{y\}, F \setminus \{x,y\}, X \setminus \{x,y\})$,

   (iii) si l'adjacent z de y dans $H - x$ est libre, alors appelez ENUMIS-F$(S \cup \{z\}, F \setminus \{x,y,z\}, X \setminus \{x,y,z\})$.

3.5.2 La correction de l'algorithme

Afin d'énumérer tous les ensembles irredondants maximaux d'une forêt T, on appelle ENUMIS-F$(\emptyset, \emptyset, V(T))$. Puisque ENUMIS-F est une version simplifiée de l'algorithme ENUMIS, obtenue en supprimant des étapes correspondant à des conditions jamais satisfaisantes dans les forêts, alors sa correction est prouvable exactement de la même manière que celle de ENUMIS en utilisant le théorème 10.

3.5.3 L'analyse du temps d'exécution

Afin d'analyser le temps d'exécution de l'algorithme ENUMIS-F, on associe à chaque règle de branchement de l'algorithme, un vecteur de branchement où la mesure d'une instance (S, F, X) est définie par la somme |F| + |X|.

Notant que tous les étapes 1–4 sont des règles de réduction.
Donc il suffit d'analyser les étapes 5–7.

Etape 5.

On branche uniquement si H′ est une arête et que V(H′) ⊆ X. Alors le vecteur de branchement est (2,2) et sa valeur est inférieure à 1.4143.

Etape 6.

Les vecteurs de branchement dans ce cas sont (2,1), (3,1), (3,2) et (3,3,2). Le maximum facteur, qui correspond au vecteur (2,1), est inférieur à 1.6181.

Etape 7.

Les vecteurs de branchement sont (2,1), (2,1), (2,1) et (2,2,3). Le maximum facteur est inférieur à 1.6181.

L'analyse du temps d'exécution montre que le pire facteur de branchement est inférieur strictement à 1.6181. Donc on peut déduire que l'algorithme s'exécute en $\mathcal{O}(1.6181^n)$, ainsi que le théorème suivant :

Théorème 12. *Une forêt de n sommets a au plus* 1.6181^n *ensembles irredondants maximaux qu'on peut énumérer en* $\mathcal{O}(1.6181^n)$.

La borne supérieure des ensembles irredondants maximaux dans une forêt de n sommets est 1.6181^n. Dans la section suivante, on propose une nouvelle borne inférieure de 1.5292^n.

3.5.4 Borne inférieure

Dans cette section, on montre qu'il existe une famille de forêts qui a au moins 1.5292^n ensembles irredondants maximaux.

Lemme 10. *Pour tout* $k \geq 1$, *il existe une forêt de* $n = 11k$ *sommets avec au moins* 1.5292^n *ensembles irredondants maximaux.*

Démonstration.

Soit le graphe $K_{1,r}$.
On désigne par $x_1, \ldots, x_r$ les r feuilles et par z le sommet non feuille du $K_{1,r}$. Soit S_r une *subdivision* du $K_{1,r}$. Le graphe S_r est obtenu par la subdivision élémentaire de chaque arête du $K_{1,r}$. D'une façon formelle, pour tout $i \in [\![1;r]\!]$, on ajoute un sommet y_i et on remplace l'arête $x_i z$ par les deux arêtes $x_i y_i$ et $y_i z$.

Clairement, si D est un ensemble irredondant maximal, alors $|\{x_i, y_i\} \cap D| \leq 1$ pour tout $i \in [\![1;r]\!]$. Alors on a deux cas possibles :

— $|\{x_i, y_i\} \cap D| = 1$, pour tout $i \in [\![1;r]\!]$:

Chaque ensemble D dans ce cas peut être obtenu en sélectionnant un sommet de chaque paire $\{x_i, y_i\}$ et en ajoutant z si tous les $x_1, \ldots, x_r$ sont sélectionnés. Par conséquent, on a 2^r ensembles irredondants maximaux.

— Il existe un $i \in [\![1;r]\!]$ tel que $\{x_i, y_i\} \cap D = \emptyset$:
Dans ce cas, par la maximalité de D, on a $z \in D$ et $\{x_j, y_j\} \cap D \neq \emptyset$ pour tout $j \neq i$ et il y a un $j \in [\![1;r]\!]$ tel que $y_j \in D$. Donc pour un tel i on a $2^{r-1}-1$ ensembles irredondants maximaux. Mais comme l'entier i varie entre 1 et r, alors on a $r(2^{r-1}-1)$ ensembles irredondants maximaux.

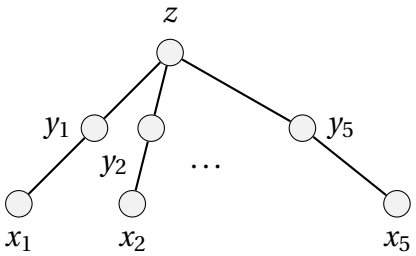

FIGURE 3.13 – Un S_5

Par conséquent, le graphe S_r a $2^r + r(2^{r-1}-1)$ ensembles irredondants maximaux. En particulier, le graphe S_5 a 107 ensembles irredondants maximaux. Par conséquent, l'union de k copies de S_5 a $n = 11k$ sommets et $107^{n/11} > 1.5292^n$ ensembles irredondants maximaux.

□

3.6 Les graphes d'intervalles

On montre dans cette section, que l'algorithme d'énumération et la borne supérieure des ensembles irredondants maximaux pour les graphes cordaux peuvent être améliorés encore dans le cas de graphes d'intervalles qui sont des sous-classes de graphes cordaux. Pour un graphe d'intervalles G, on suppose que sa représentation par un ensemble d'intervalles est donnée en entrée. On rappelle que dans l'ensemble d'intervalles, chaque sommet correspond à un intervalle de la droite réelle. Deux sommets sont adjacents dans le graphe G si et seulement si leurs intervalles correspondants ont une intersection non vide. Afin d'améliorer la lisibilité de l'algorithme, on ne distingue pas un sommet et son intervalle correspondant dans la représentation. Pour un sommet v, on note ℓ_v et r_v les deux extrémités gauche et droite respectivement de l'intervalle correspondant à v dans la représentation.

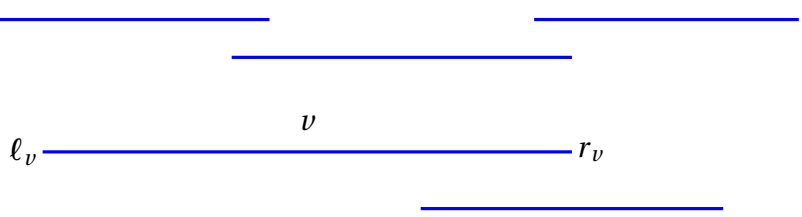

FIGURE 3.14 – Un graphe d'intervalles

3.6.1 L'algorithme d'énumération

Soit G un graphe d'intervalle donné avec sa représentation. Soit ENUMIS-I(S,F,X) un algorithme récursif de type « Brancher et réduire », où $S, X, F \subseteq V(G)$ sont des sous-

ensembles disjoints tels que $\ell_x \leq r_y$ pour $x \in S$ et $y \in F \cup X$.

L'algorithme EnumIS-I(S,F,X) énumère tous les ensembles irredondants maximaux D dans G tels que $S \subseteq D$, $D \setminus S \subseteq X$ et pour chaque $v \in D \setminus S$, v a un privé dans $F \cup X$.

Afin d'énumérer tous les ensembles irredondants maximaux de G, on appelle EnumIS-I($\emptyset, \emptyset, V(G)$).

Soit $H = G[X \cup F]$, le graphe induit de G par $X \cup F$ tel que $F \cap D = \emptyset$. On dit qu'un sommet est *interdit* s'il appartient à F et ne peut pas être inclus dans un ensemble irredondant maximal D alors qu'un sommet est *libre* s'il appartient à X. On dit aussi qu'un sommet de H est *dominé* s'il est dominé dans G par un sommet de S.

À chaque étape de l'algorithme, on réduit l'instance par une règle de réduction ou on branche. On suppose toujours qu'une étape de l'algorithme est appliquée si aucune étape précédente n'est applicable. À chaque étape, si on décide d'ajouter un sommet $u \in X$ dans un ensemble irredondant potentiel S, alors on doit s'assurer qu'il obtienne un privé v dans $F \cup X$ qui ne soit pas dominé par d'autres sommets. Pour cela, on déclare comme interdit, tous les voisins de v sauf u.

On dit que le triplet (S,X,F) des sous-ensembles de V(G) est *régulier*, si S, F et X sont disjoints et $\ell_x \leq r_y$ pour $x \in S$ et $y \in F \cup X$. On dit qu'un ensemble irredondant maximal D est *compatible* avec un triplet régulier (S,F,X) si les conditions suivantes sont satisfaisantes :

(a) $S \subseteq D$,

(b) $D \setminus S \subseteq X$,

(c) pour tout $v \in D \setminus S$, v a un privé dans $F \cup X$,

(d) pour tout $u \in S$, u a un privé $v \in V(G) \setminus (F \cup X)$ tel que $N_G(v) \cap X = \emptyset$.

L'idée principale de la preuve de correction de EnumIS-I(S,F,X) est de prouver que l'algorithme prend en compte chaque ensemble irredondant maximal compatible avec (S,F,X) et il le produit.

EnumIS-I(S,F,X)

1. Si $X = \emptyset$, alors vérifiez si S est un ensemble irredondant maximal dans G et le retournez si le cas ; arrêtez.
2. Si H a un sommet interdit et dominé x, alors appelez EnumIS-I($S, F \setminus \{x\}, X$).
3. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont dominés, appelez EnumIS-I($S, F, X \setminus \{x\}$).
4. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont interdits, appelez EnumIS-I($S, F \setminus \{x\}, X$).
5. Trouvez un sommet u de H avec l'extrémité droite minimum r_u, et notez $N_H(u) = \{v_1, \ldots, v_k\}$, où les sommets $v_1, \ldots, v_k$ sont ordonnés selon un ordre croissant de leurs extrémités droites.
6. Si u est dominé, alors recherchez l'index minimum $i \in \{1, \ldots, k\}$ tel que v_i ne soit pas dominé et branchez :
   (i) appelez EnumIS-I($S \cup \{u\}, F \setminus N_H[v_i], X \setminus N_H[u]$),
   (ii) appelez EnumIS-I($S, F \setminus \{u\}, X \setminus \{u\}$).
7. Si u est interdit, alors branchez :
   (i) pour tout $i = 1, \ldots, k$, si v_i est libre, alors appelez EnumIS-I($S \cup \{v_i\}, F \setminus N_H[u], X \setminus N_H[u]$),

(ii) appelez ENUMIS-I$(S, F \setminus \{u\}, X)$.

8. Si $N_H(u) = \emptyset$, alors appelez ENUMIS-I$(S \cup \{u\}, F \setminus \{u\}, X \setminus \{u\})$.
9. Soit $W_0 = \emptyset$, et pour $i = 1, \ldots, k$, on a

$$W_i = \big(\{w \in N_H(v_i) \setminus N_H[u] \mid r_w > r_{v_i}\} \cap X\big) \setminus \big(\bigcup_{j=0}^{i-1} W_j\big)$$

.

10. Si $W_i = \emptyset$ pour tout $i \in \{1, \ldots, k\}$, alors branchez :
    (i) appelez ENUMIS-I$(S \cup \{u\}, F \setminus N_H[u], X \setminus N_H[u])$,
    (ii) pour tout $i = 1, \ldots, k$, si v_i est libre, alors appelez ENUMIS-I$(S \cup \{v_i\}, F \setminus N_H[u], X \setminus N_H[u])$.
11. Sinon, branchez :
    (i) appelez ENUMIS-I$(S \cup \{u\}, F \setminus N_H[u], X \setminus N_H[u])$,
    (ii) pour tout $i = 1, \ldots, k$, si v_i est libre, alors appelez ENUMIS-I$(S \cup \{v_i\}, F \setminus N_H[u], X \setminus N_H[u])$,
    (iii) pour tout $i = 1, \ldots, k$, si $W_i \neq \emptyset$, alors pour tout $w \in W_i$, appelez ENUMIS-I$(S \cup \{w\}, F \setminus N_H[v_i], X \setminus N_H[v_i])$.

La preuve de correction de l'algorithme ENUMIS-I(S, F, X) est basée sur le lemme suivant.

Lemme 11. *Soit* D *un ensemble irredondant maximal du graphe* G *compatible avec un triplet régulier* (S, F, X) *des sous-ensembles disjoints de* $V(G)$. *Si l'une des conditions des étapes 1–11 est satisfaite, alors soit* $D = S$ *et l'algorithme retourne* D*, soit l'algorithme ne s'arrête pas et il y a un appel récursif de* ENUMIS-I(S', F', X') *dans l'appel de* ENUMIS-I(S, F, X) *tel que* D *est compatible avec le triplet régulier* (S', X', F').

Démonstration.

On note qu'aux étapes 5 et 9, on a juste spécifié des sommets et on a construit des ensembles. Donc ces étapes ne doivent pas être analysées dans la démonstration.

On note également que si (S, F, X) est régulier, alors (S, F', X'), où $F' \subseteq F$ et $X' \subseteq X$, est régulier par définition.

De même, si $S' = S \cup \{u\}$, où u est un sommet de H avec l'extrémité droite minimum r_u, et $F' \subseteq F \setminus \{u\}$ et $X' \subseteq X \setminus \{u\}$, alors (S', F', X') est régulier. Cela signifie que les triplets (S', F', X') construits aux étapes 1–6, 7 (ii), 10 (i) et 11 (i) sont réguliers. Par conséquent, on ne devrait vérifier la régularité que pour les autres étapes.

On considère 9 cas correspondant aux étapes 1–4, 6–8, 10 et 11 de l'algorithme.

Cas 1. L'ensemble $X = \emptyset$. Comme $D \setminus S \subseteq X$, $D = S$ et D est un ensemble irredondant maximal de G, alors l'algorithme retourne D.

Cas 2. Si H a un sommet x interdit et dominé.

Comme x est interdit, alors $x \notin D$. De plus x est dominé par S. Donc il ne peut pas être un privé pour aucun sommet de $D \setminus S$. Par conséquent, D est compatible avec $(S, F \setminus \{x\}, X)$.

Cas 3. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont dominés.

Le sommet $x \in X$, car l'étape 2 n'est pas applicable. On note également que $x \notin D$, sinon il n'aurait pas un privé dans $F \cup X$. De plus, x ne peut pas être un privé pour aucun sommet de $D \setminus S$, car x est dominé. Par conséquent, D est compatible avec $(S, F, X \setminus \{x\})$.

Cas 4. S'il y a un sommet $x \in V(H)$ tel que tous les sommets $N_H[x]$ sont interdits.

Clairement, $x \notin D$ parce qu'il est interdit. En outre, il ne peut pas être un privé pour aucun sommet de $D \setminus S$, car $N_G[x] \cap (D \setminus S) = \emptyset$. Par conséquent, D est compatible avec $(S, F \setminus \{x\}, X)$.

Remarque :

On rappelle que $u \in V(H)$ est un sommet avec l'extrémité droite minimum r_u et $N_H(u) = \{v_1, \ldots, v_k\}$, où les sommets $v_1, \ldots, v_k$ sont ordonnés selon un ordre croissant de leurs extrémités droites.

On note que u est un sommet simplicial de H et que $N_H[u] \subseteq N_H[v_1] \subseteq \cdots \subseteq N_H[v_k]$. Cela implique notamment que $|N_H[u] \cap D| \leq 1$. Pour le prouver, on suppose, par l'absurde, que $|N_H[u] \cap D| > 1$. On suppose qu'il existe un sommet $v_i \in D$ pour $i \in \{1, \ldots, k\}$ et $u \in D$. Mais comme $N_H[u] \subseteq N_H[v_i]$ alors le sommet u n'a pas un privé ; contradiction avec d). Donc on suppose qu'il existe deux sommets $v_i, v_j \in D$ tels que $i < j \leq k$. De même, comme $N_H[v_i] \subseteq N_H[v_j]$ alors v_i n'a pas de privé ; contradiction avec d). Par conséquent, $|N_H[u] \cap D| \leq 1$.

Cas 6. Le sommet u de H avec l'extrémité droite minimum r_u est dominé.

Comme le sommet u est dominé et les étapes 2 et 3 ne sont pas applicables, alors u n'est pas interdit et il a au moins un voisin non dominé dans H. Soit v_i un tel voisin avec l'index minimum i. On suppose que $u \in D$. Alors, $N_H(u) \cap D = \emptyset$ car $|N_H[u] \cap D| \leq 1$. De même les sommets $N_H(u)$ ne peuvent pas être des privés pour les sommets de $D \setminus \{u\}$, car ils sont dominés par u.

On suppose que le sommet u a un privé $v \in N_H(u)$ qui n'est pas dominé par S. Donc $r_{v_i} \leq r_v$.

Soit $w \in V(H) \setminus N_H[u]$. On montre maintenant par contraposition que si $wv_i \in E(H)$ alors $wv \in E(H)$.

On suppose donc que $wv \notin E(H)$. Comme le sommet u a l'extrémité droite minimum et $uv \in E(H)$, alors on a $\ell_v \leq r_u \leq r_v$. Et comme $wv \notin E(H)$ et $wu \notin E(H)$, alors $\ell_v \leq r_u \leq r_v < l_w \leq r_w$. Par le choix de v_i, on a $r_{v_i} \leq r_v$. Donc $r_{v_i} < l_w$ et $wv_i \notin E(H)$. Par conséquent, si $wv_i \in E(H)$, alors $wv \in E(H)$. On peut conclure que v_i est un privé de u et $N_H[v_i] \cap D = \{u\}$.

Soit $S' = S \cup \{u\}$, $F' = F \setminus N_H[v_i]$ et $X \setminus N_H[u]$. Par conséquent, D est compatible avec (S', F', X').

Si $u \notin D$, alors D est compatible avec $(S, F \setminus \{u\}, X \setminus \{u\})$, car u est dominé et il ne peut pas être un privé pour aucun sommet de D.

Cas 7. Le sommet u de H avec l'extrémité droite minimum r_u est interdit.

Comme les cas 2 et 4 ne s'appliquent pas, alors le sommet u n'est pas dominé et il a au moins un voisin libre dans H.

Puisque u est interdit, alors $u \notin D$.

On suppose que $N_H(u) \cap D = \{v_i\}$, pour $i \in \{1, \ldots, k\}$. C'est évident que $v_i \in X$. Puisque u n'est pas dominé, alors u est un privé pour v_i.

Comme les sommets $N_H[u]$ forment une clique, alors les sommets de $N_H(u) \setminus \{v_i\}$ ne peuvent pas être des privés pour les sommets de $D \setminus \{v_i\}$.

Soit $S' = S \cup \{v_i\}$, $F' = F \setminus N_H[u]$ et $X' = X \setminus N_H[u]$.

On observe que $\ell_{v_i} \leq r_u \leq r_y$ pour $y \in F' \cup X'$, c'est-à-dire que le triplet (S', F', X') est régulier. Par conséquent, D est compatible avec (S', F', X').

On suppose que $|N_H(u) \cap D| \neq 1$. Alors u ne peut pas être un privé pour aucun sommet $x \in D \setminus S$. Par conséquent, D est compatible avec $(S, F \setminus \{u\}, X)$.

A partir de maintenant, on suppose que u est libre et non dominé, car les étapes 6 et 7 ne s'appliquent pas.

Cas 8. Le sommet u de H avec l'extrémité droite minimum r_u est un sommet isolé dans H.

Le sommet u est isolé, c'est-à-dire que $N_H(u) = \emptyset$. Par la maximalité de D, le sommet $u \in D$ et il est un privé pour lui-même. Par conséquent, D est compatible avec $(S \cup \{u\}, F \setminus \{u\}, X \setminus \{u\})$.

A partir de maintenant, on suppose que $N_H(u) \neq \emptyset$.
On rappelle que $W_0 = \emptyset$ par définition et que pour $i \in \{1, \ldots, k\}$, on a (voir la figure 3.15) :

$$W_i = \left(\{w \in N_H(v_i) \setminus N_H[u] \mid r_w > r_{v_i}\} \cap X\right) \setminus \left(\bigcup_{j=0}^{i-1} W_j\right)$$

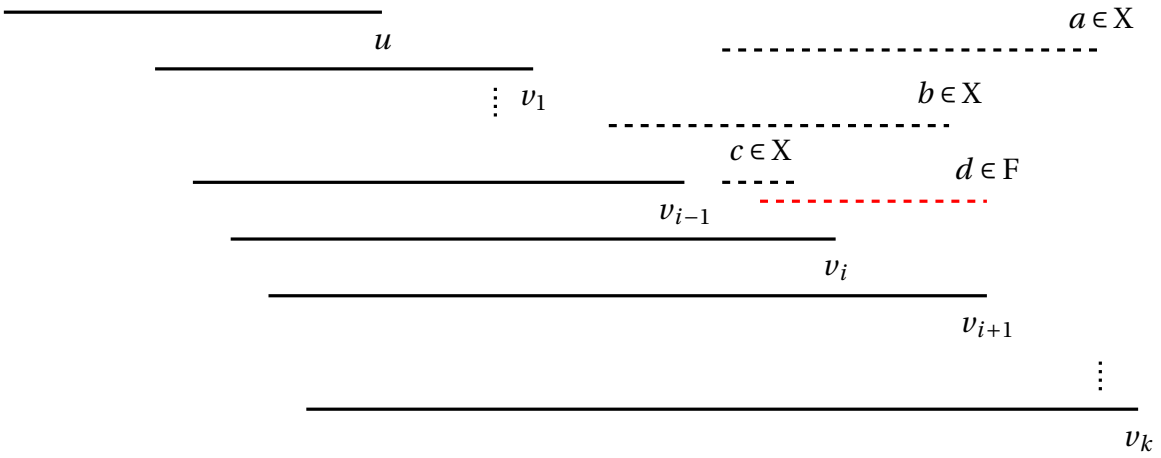

FIGURE 3.15 – $b, c, d \notin W_i$ car $d \in F$, $b \in W_j$ pour un $j < i$ et $r_c < r_{v_i}$, alors que $a \in W_i$.

On montre maintenant que pour $W = N_H[u] \cup \bigcup_{i=1}^{k} W_k$, $W \cap D \neq \emptyset$.

Démonstration. Pour obtenir une contradiction, on suppose que $W \cap D = \emptyset$. On considère $D' = D \cup \{u\}$. On montre que D' est un ensemble irredondant contredisant la maximalité de D. On suppose que D' ne soit pas un ensemble irredondant. Donc il y a $x \in D'$ tel que x n'a pas de privé par rapport à D'. Comme $N_H(u) \cap D' = \emptyset$, alors u est un privé pour lui-même par rapport à D'. Par conséquent, $x \neq u$.

Par d), on a $x \notin S$. Puisque $N_H(u) \cap D' = \emptyset$, on a $x \in X \setminus N_H[u]$. En particulier, $r_u < \ell_x$.

Parce que x a des privés par rapport à D et non pas par rapport à $D' = D \cup \{u\}$, alors tous ses privés sont en $N_H(u)$. Soit v_i un privé de x par rapport à D. On note que x n'est pas un privé pour lui-même par rapport à D. Par conséquent, x est dominé dans D.

Soit $y \in D$ un sommet adjacent à x. On suppose que $y \in S$. Parce que (S, F, X) est régulier, alors $\ell_y \leq r_u$. Comme $xy \in E(G)$ et $r_u < \ell_x$, on a $\ell_y \leq r_u \leq r_y$ et $yu \in E(G)$. Cela signifie que u est dominé par S ; contradiction. Donc $y \notin S$.

Parce que $yv_i \notin E(G)$, on a $\ell_x \leq r_{v_i} < \ell_y \leq r_x$. Cela signifie que $x \in N_H(v_i)$ et $r_x > r_{v_i}$. Par conséquent, il existe un minimum $j \in \{1, \ldots, k\}$ tel que $x \in N_H(v_j) \setminus N_H[u]$ et $r_x > r_{v_j}$. Alors $x \in W_j \subseteq W$; contradiction car $D \cap W = \emptyset$. □

Cas 10. Si $W_i = \emptyset$ pour tout $i \in \{1,\dots,k\}$.

On rappelle que $|N_H[u] \cap D| \leq 1$. Puisque $W_i = \emptyset$ pour tout $i \in \{1,\dots,k\}$, alors $W = N_H[u]$ et $W \cap D \neq \emptyset$. Par conséquent, $|N_H[u] \cap D| = 1$.

Soit $F' = F \setminus N_H[u]$ et $X' = X \setminus N_H[u]$. Si $u \in D$, alors u est un privé pour lui-même et les sommets de $N_H(u)$ ne sont pas des privés pour les sommets de $D \setminus \{u\}$. Par conséquent, D est compatible avec $(S \cup \{u\}, F', X')$.

Si $v_i \in D$ pour un $i \in \{1,\dots,k\}$, alors u est un privé pour v_i et les sommets de $N_H[u] \subseteq N_H[v_i]$ ne sont pas des privés pour les sommets de $D \setminus \{v_i\}$. Comme $\ell_{v_i} \leq r_u$, $\ell_{v_i} \leq \ell_y$ pour tout $y \in V(H) \setminus N_H[u] = F' \cup X'$. Par conséquent, D est compatible avec le triplet régulier $(S \cup \{v_i\}, F', X')$.

Case 11. Il existe un $i \in \{1,\dots,k\}$ tel que $W_i \neq \emptyset$.

On rappelle que $|N_H[u] \cap D| \leq 1$. Si $u \in D$, alors, de la même manière que le cas 10, D est compatible avec $(S \cup \{u\}, F \setminus N_H[u], X \setminus N_H[u])$.

Toujours avec les mêmes arguments utilisés dans le cas 10, si $v_i \in D$ pour un $i \in \{1,\dots,k\}$, alors D est compatible avec le triplet régulier $(S \cup \{v_i\}, F \setminus N_H[u], X \setminus N_H[u])$.

On suppose que $N_H[u] \cap D = \emptyset$. On rappelle que $W \cap D \neq \emptyset$. Par conséquent, il y a un $i \in \{1,\dots,k\}$ tel que $W_i \neq \emptyset$. On montre qu'il existe un $i \in \{1,\dots,k\}$ tel que $|D \cap W_i| = 1$ et $D \cap W_j = \emptyset$ pour tout $j \in \{0,\dots,i-1\}$. Soit l'index minimum $i \in \{1,\dots,k\}$ tel que $W_i \cap D \neq \emptyset$.

On suppose que $|W_i \cap D| \geq 2$.

Soit $x, y \in \{W_i \cap D\}$ et on suppose que $r_x \leq r_y$. Par la définition de W_i, $x, y \in N_H(v_i) \setminus N_H[u]$, $\ell_x \leq r_{v_i} < r_x$ et $\ell_y \leq r_{v_i} < r_y$. En particulier, cela signifie que $xy, xv_i, yv_i \in E(G)$. De plus, $xv_h, yv_h \in E(G)$ si $i \leq h \leq k$. On note également que $xv_h, yv_h \notin E(G)$ si $1 \leq h < i$.

On considère $D' = D \cup \{u\}$. Puisque D est un ensemble irredondant maximal, alors D' ne peut pas être un ensemble irredondant. Cela implique qu'il existe un sommet $v \in D'$ qui n'a pas de privé. Puisque $N_H[u] \cap D = \emptyset$ et u n'est pas dominé par S, alors u est un privé pour lui-même par rapport à D' et $v \neq u$. Puisque D est compatible avec (S,F,X), par d), on a $v \notin S$, c'est-à-dire, $v \in D \setminus S$. Cela implique que $v \in X \setminus N_H[u]$. Parce que v a un privé par rapport à D et non pas par rapport à $D' = D \cup \{u\}$, alors ce privé est dans $N_H(u)$. Soit v_j un tel privé pour $j \in \{1,\dots,k\}$. Comme $xv_h, yv_h \notin E(G)$ si $1 \leq h < i$ et $xv_h, yv_h \in E(G)$ si $i \leq h \leq k$, alors $v \neq x, y$ et $j < i$. Clairement $\ell_v \leq r_{v_j}$, et comme $W_h \cap D = \emptyset$ pour tout $h \in \{0,\dots,i-1\}$, alors $r_v \leq r_{v_j}$. Puisque v n'est pas un privé pour lui-même par rapport à D, alors v est dominé dans D. Mais tout sommet de $D \setminus \{v\}$ adjacent à v est également adjacent à v_j, car $\ell_{v_j} < \ell_v \leq r_v \leq r_{v_j}$; contradiction puisque v_j est un privé de v par rapport à D.

On considère $i \in \{1,\dots,k\}$ tel que $|D \cap W_i| = 1$ et $D \cap W_j = \emptyset$ pour tout $j \in \{0,\dots,i-1\}$. Soit $D \cap W_i = \{w\}$. On montre que $N_H[v_i] \cap D = \{w\}$. Pour obtenir une contradiction, on suppose qu'il existe un $x \neq w$ tel que $x \in N_H[v_i] \cap D$. Clairement, le sommet $x \in (N_H[v_i] \cap X) \setminus N_H[u]$. On note également que $x \notin W_j$ pour $j \in \{1,\dots,i\}$, car $W_j \cap D = \emptyset$ si $1 \leq j < i$ et $W_i \cap D = \{w\}$. Par conséquent, il y a $j \in \{1,\dots,i\}$ tel que $\ell_{v_j} < r_u < \ell_x \leq r_x \leq r_{v_j}$ et $r_{v_h} < \ell_x$ si $1 \leq h < j$.

On suppose, sans perte de généralité, que x et j sont choisis pour minimiser j.

Soit $D' = D \cup \{u\}$. Puisque D est un ensemble irredondant maximal, alors D' ne peut pas être un ensemble irredondant, c'est-à-dire qu'il y a un sommet $v \in D'$ qui n'a pas de privé. Puisque $N_H[u] \cap D = \emptyset$ et u n'est pas dominé par S, alors u est un privé pour lui-même par rapport à D' et $v \neq u$. De plus, par d), on a $v \notin S$, c'est-à-dire que $v \in D \setminus S$. Il s'ensuit donc que $v \in X \setminus N_H(u)$.

Parce que v a un privé par rapport à D et non pas par rapport à $D' = D \cup \{u\}$, le sommet v a de privé seulement dans $N_H(u)$. Comme v n'est pas un privé pour lui-même, v est dominé par un sommet $y \in D \setminus \{v\}$. De plus, le sommet u n'est pas dominé dans D et le

triplet (S,F,X) est régulier. Donc $y \in D \setminus S$.
Si $v = x$, alors $v_h y \in E(G)$ pour tout $h \in \{j,\dots,k\}$, car $\ell_{v_j} < \ell_x \leq r_x \leq r_{v_j}$ et $[\ell_x, r_x] \cap [\ell_y, r_y] \neq \emptyset$. Alors v n'a pas de privé dans $\{v_1,\dots,v_k\}$. Par conséquent, $v \neq x$ et tous les privés de v par rapport à D sont dans $\{v_1,\dots,v_{j-1}\}$.
Puisque $W_h = \emptyset$ pour tout $h \in \{1,\dots,j-1\}$, alors il existe un $h \in \{1,\dots,j-1\}$ tel que $\ell_{v_h} < \ell_v \leq r_v \leq r_{v_h}$ et $r_{v_s} < \ell_x$ si $1 \leq s < h$, mais cela contredit le choix de x et j. Par conséquent, D′ est irredondant ; contradiction.

Maintenant, on affirme que les sommets de $N_H[v_i]$ ne sont pas des privés pour les sommets de $D \setminus (S \cup \{w\})$. On rappelle que $\ell_w \leq r_{v_i} < r_w$.
S'il existe un sommet $x \in N_H[v_i]$ qui est un privé pour un sommet $v \in D \setminus (S \cup \{w\})$, alors $xw \notin E(G)$ et, par conséquent, $r_v < \ell_w$. Cela implique que $v \in N_H[v_i]$ et $v \neq w$ se contredisent $N_H[v_i] \cap D = \{w\}$.

Soit $S' = S \cup \{w\}$, $F' = F \setminus N_H[v_i]$ et $X' = X \setminus N_H[v_i]$. On note que si $y \in F' \cup X'$, alors $\ell_y > r_{v_i}$ et $\ell_w \leq r_{v_i} < \ell_y \leq r_y$ et, par conséquent, (S',F',X') est régulier.
Par conséquent, D est compatible avec (S',F',X').

3.6.2 La correction de l'algorithme

A chaque étape de l'algorithme EnumIS-I(S,F,X), soit on produit une solution S à l'étape 1, soit on appelle EnumIS-I(S′,F′,X′) tel que $|X'| + |F'| < |X| + |F|$. Aussi, c'est facile de voir que les étapes de l'algorithme sont exhaustives. Donc l'algorithme s'arrête toujours.
Afin d'énumérer tous les ensembles irredondants maximaux dans un graphe d'intervalles G, il suffit d'appeler EnumIS-I($\emptyset,\emptyset,V(G)$). L'algorithme génère des ensembles seulement à l'étape 1 et il affiche, après une vérification en temps polynomial, un ensemble s'il est uniquement un ensemble irredondant maximal. Par conséquent, EnumIS-I($\emptyset,\emptyset,V(G)$) ne produit que des ensembles irredondants maximaux.

On montre alors que EnumIS-I($\emptyset,\emptyset,V(G)$) produit chaque ensemble irredondant maximal. Soit D un ensemble irredondant maximal. On montre, par récurrence sur $|X| + |F|$, que si D est compatible avec un triplet régulier (S,F,X) des sous-ensembles disjoints de V(G), alors EnumIS-I(S,F,X) produit D. Si $|X| + |F| = 0$, alors c'est trivial, car D = S dans ce cas et on produit D à l'étape 1.

On suppose que l'affirmation est vraie pour $|X''| + |F''| < |X| + |F|$. Et on le prouve pour $|X| + |F|$. Comme les étapes de l'algorithme sont exhaustives, on appelle EnumIS-I(S′,F′,X′) dans l'une des étapes de l'algorithme. Et par le lemme 11, D est compatible avec le triplet régulier (S′,F′,X′). Comme $|X'| + |F'| < |X| + |F|$, alors par l'hypothèse de récurrence EnumIS-I(S′,F′,X′) produit D. Donc si D est compatible avec un triplet régulier (S,F,X) des sous-ensembles disjoints de V(G), alors EnumIS-I(S,F,X) produit D.

De plus, tout ensemble irredondant maximal D est compatible avec $(\emptyset,\emptyset,V(G))$. Donc EnumIS-I($\emptyset,\emptyset,V(G)$) énumère tous les ensembles irredondants maximaux du graphe G.

3.6.3 L'analyse du temps d'exécution

Afin d'analyser le temps d'exécution, on associe à chaque règle de branchement de l'algorithme, un vecteur de branchement où la mesure d'une instance (S,F,X) est définie par la somme $|F| + |X| = |V(H)|$. Donc il suffit d'analyser les étapes correspondant aux règles de branchement : les étapes 6–7 et 10–11.

Etape 6.

Soit $t = d_H(u) + 1$. On a $t \geq 2$, car les étapes 2 et 3 ne s'appliquent pas. Alors le vecteur de branchement est $(t, 1)$ qui a un facteur de branchement maximum, atteint si $t = 2$, inférieur à 1.6181.

Etape 7.

Soit $t = |N_H(u) \cap X|$. On rappelle que $t \geq 1$, car les étapes 2 et 4 ne s'appliquent pas. Le vecteur de branchement dans le pire des cas est $(\underbrace{t+1, \ldots, t+1}_{t}, 1)$ et son facteur maximum est atteint si $t = 2$. Ce dernier est inférieur strictement à 1.6957.

Etape 10.

Au pire des cas, tous les adjacents de u sont libres et le vecteur de branchement dans ce cas est $(\underbrace{t, \ldots, t}_{t})$, avec $t = d_H(u) + 1 \geq 2$. Le facteur maximum est atteint pour $t = 3$, (voir 2.2.3), et qui est inférieur à 1.4423.

Etape 11.

Soit $t = d_H(u) + 1 \geq 2$. On note $i_1 < \ldots < i_h$ les index de $\{1, \ldots, k\}$ (On rappelle que $k = d_H(u) = t - 1$) tel que $W_{i_j} \neq \emptyset$ pour $j \in \{1, \ldots, h\}$. Soit $t_j = |W_{i_j}|$ et $r_j = \sum_{s=1}^{j} t_s$ pour $j \in \{1, \ldots, h\}$. Il est clair que le nombre de branches à l'étape 11 (i) et (ii) est au plus t. Alors pour tout $j \in \{1, \ldots, h\}$, on a t_j branches. Par conséquent, le vecteur de branchement dans le pire des cas est

$$(\underbrace{t, \ldots, t}_{t}, \underbrace{t + r_1, \ldots, t + r_1}_{t_1}, \ldots, \underbrace{t + r_h, \ldots, t + r_h}_{t_h}). \tag{3.1}$$

Le facteur maximum dans le cas considéré est inférieur à 1.6529 et il est atteint pour $t = 2$, $h = 1$ et $t_1 = 2$ (cela implique que $r_1 = 2$), autrement dit pour un vecteur de branchement $(2, 2, 4, 4)$.

L'analyse du temps d'exécution montre que le pire facteur de branchement est inférieur strictement à 1.6957. Donc on peut déduire que l'algorithme s'exécute en $\mathcal{O}(1.6957)$, ainsi que le théorème suivant :

Théorème 13. *Un graphe d'intervalles de n sommets a au plus* 1.6957^n *ensembles irredondants maximaux qu'on peut énumérer en* $\mathcal{O}(1.6957^n)$.

On complète cette borne supérieure obtenue avec la citation d'une borne inférieure dans la sous-section suivante.

3.6.4 Borne inférieure

La famille de forêts décrite dans la section 3.5.4 n'est pas un graphe d'intervalles. Alors on cite comme une borne inférieure pour cette famille de graphes le lemme suivant :

Lemme 12 ([34])**.** *Le nombre maximum des ensembles irredondants maximaux dans les graphes d'intervalles est au moins* 1.4696^n.

3.7 Conclusion

Chaque fois qu'on ajoute un sommet v à un ensemble irredondant maximal D du graphe G, on fixe pour le sommet v un privé x qui ne soit pas dominé par d'autres sommets de l'ensemble D. Pour cela, on marque tous les adjacents de x, sauf v, comme des

sommets interdits. La fixation d'un privé parmi tous les adjacents, nécessite une régle de branchement qui est généralement coûteuse du point de vue temps d'exécution. De plus, on peut avoir dans un ensemble irredondant maximal un sommet qui n'est pas dominé. Du coup la conception des bons algorithmes d'énumération des ensembles irredondants maximaux n'est nulle part facile.

Mais malgré ça, on a réussi à concevoir des algorithmes d'énumération des ensembles irredondants maximaux dans les graphes cordaux, les graphes d'intervalles et les forêts en $\mathcal{O}(1.7549^n)$, $\mathcal{O}(1.6957^n)$ et $\mathcal{O}(1.6181^n)$ respectivement au lieu de l'algorithme trivial en $\mathcal{O}^*(2^n)$. Afin de compléter ces résultats, on a proposé comme borne inférieure une famille de forêts, non triviale, de 1.5292^n ensembles irredondants maximaux.

On a montré aussi que chaque ensemble irredondant maximal dans un cographe est un ensemble dominant minimal et cela signifie qu'on peut énumérer en $\mathcal{O}^*(15^{n/6})$. Alors le problème semble facile à résoudre dans le cas des cographes et des graphes scindés.

Et la question qui se pose, si on peut faire mieux que $\mathcal{O}^*(2^n)$ pour les graphes sans P_5 qui sont à la fois des super-classes de cographes et de graphes scindés. De même, les questions sont encore ouvertes pour les forêts. Récemment ROTE [59] a utilisé une technique nouvelle pour résoudre le problème d'énumération des ensembles dominants minimaux dans les forêts. Alors il semble très intéressant d'appliquer une telle approche sur les ensembles irredondants maximaux. La question reste ouverte pour les graphes cordaux aussi, vu que les bornes ne sont pas strictes. Mais l'algorithme qu'on a proposé est intéressant par le fait qu'il s'appuie sur la nouvelle propriété de semi-simpliciaux. Cette nouvelle propriété semble utile pour d'autres problèmes d'énumération ou d'optimisation. Récemment GOLOVACH, HEGGERNES, KRATSCH et SAEI [33] ont réussi à exploiter cette propriété en améliorant l'algorithme d'énumération des ensembles dominants connexes minimaux dans les graphes cordaux de $\mathcal{O}(1.7159^n)$ [32] à $\mathcal{O}(1.4736^n)$.

De même, l'étude de la borne supérieure dans les graphes quelconques reste un challenge pour la communauté, vu qu'il n'existe aucune preuve que cette borne est inférieure à 2^n. Récemment, ce problème a été mentionné comme « open problem » au séminaire 18421 de « Dagstuhl » [23].

Dans les chapitres suivants, on augmente le défi et on étudiera l'énumération des certains ensembles connexes où la contrainte de connexité est une contrainte globale.

Finalement on donne un survol, sous forme d'un tableau, des résultats connus sur l'énumération « input-sensitive » des ensembles irredondants maximaux, où n est le nombre de sommets du graphe G.

Les classes de graphes	Borne inférieure	Borne supérieure
Les graphes quelconques	1.5848^n [35]	2^n
Les graphes sans griffes	1.5848^n [35]	$\mathcal{O}(1.9341^n)$ [35]
Les chemins	$\Theta(1.4696\ldots^n)$ [34]	$\Theta(1.4696\ldots^n)$ [34]
Les cordaux	1.5292^n [cette thèse]	1.7549^n [cette thèse]
Les graphes d'intervalles	1.4696^n [34]	1.6957^n [cette thèse]
Les forêts	1.5292^n [cette thèse]	1.6181^n [cette thèse]
Les cographes	$15^{n/6}$ [24]	$15^{n/6}$ [17] [cette thèse]
Les graphes scindés	$3^{n/3}$ [17]	$3^{n/3}$ [18, 46]

TABLEAU 3.1 – Un survol sur l'énumération des ensembles irredondants maximaux

Chapitre 4

Enumération des ensembles connexes minimaux

Sommaire

4.1 Etat de l'art

Soit G = (V,E) un graphe. Un ensemble D domine un sommet v, si $v \in D$ ou $N(v) \cap D \neq \emptyset$. Un ensemble de sommets $D \subseteq V$ est un ensemble dominant de G si pour tout $v \in V$, $v \in N[D]$. On dit qu'un ensemble dominant D est minimal s'il n'existe aucun autre ensemble dominant $D' \subset V$ tel que $D' \subset D$. Un ensemble dominant D est minimal si pour tout sommet $v \in D$, v a un privé. Un ensemble de sommets D est un ensemble dominant connexe de G si D est un ensemble dominant et G[D] est connexe. L'ensemble D est minimal s'il n'existe aucun autre ensemble dominant connexe $D' \subset V$ tel que $D' \subset D$. Un ensemble dominant connexe D est minimal si pour tout $x \in D$: soit $D \setminus \{x\}$ n'est pas dominant, soit le sous graphe induit par $D \setminus \{x\}$ est non connexe [46]. En d'autres termes, chaque sommet $x \in D$ soit a un voisin privé soit x est un point d'articulation dans G[D].

On note bien que chaque point d'articulation de G appartient à chaque ensemble dominant connexe minimal de G.

Le problème, dans sa version d'optimisation, c'est à dire compter la cardinalité minimale d'un ensemble dominant connexe minimal dans un graphe quelconque, est connu depuis longtemps NP-difficile [27]. De plus, il est NP-difficile dans les graphes bipartis [57] et les graphes bipartis cordaux [56]. Le meilleur algorithme connu qui résoud ce problème dans un graphe quelconque est en $\mathcal{O}(1.8619^n)$ [2]. Toutefois, il convient de noter que le problème est « tractable » dans les graphes bipartis convexes avec un algorithme efficace en $\mathcal{O}(n^3)$ [19]. En outre, trouver un ensemble dominant de cardinalité minimale dans un graphe biparti convexe avec n sommets est résolvable en $\mathcal{O}(n^2)$ [3].

Concernant l'énumération des ensembles dominants connexes minimaux dans les graphes quelconques, le problème est connu d'être notoirement difficile. Actuellement, ces ensembles ne peuvent pas être énumérés que légèrement plus rapidement que $\mathcal{O}^*(2^n)$. En fait l'algorithme, hautement non trivial, est en $\mathcal{O}(2^{(1-\epsilon)n})$ avec $\epsilon > 10^{-50}$ [54].

D'un autre côté, la meilleure borne inférieure (figure 4.1) connue est $3^{(n-2)/3}$[32]. La borne est construite de la manière suivante :

— Pour $i \in \{1,\ldots,k\}$, construisez une clique de 3 sommets $T_i = \{x_i, y_i, z_i\}$. Pour des raisons de simplification, on a modélisé une clique par un rectangle dans la figure 4.1.

— Pour $i \in \{2,\ldots,k\}$, reliez par une arête chaque sommet de T_{i-1} avec chaque sommet de T_i.

— Ajoutez deux sommets u et v, ainsi que les arêtes ux_1, uy_1, uz_1, vx_k, vy_k et vz_k.

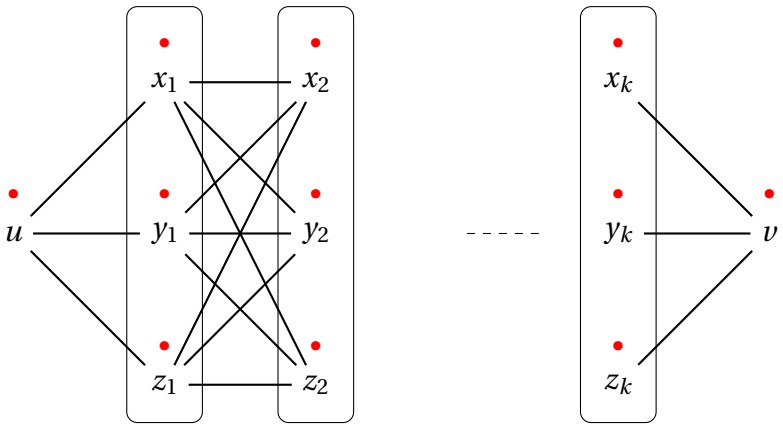

FIGURE 4.1 – La borne inférieure des ensembles dominants connexes minimaux

Malgré le grand écart entre la borne inférieure et la borne supérieure dans les graphes quelconques. GOLOVACH, HEGGERNES et KRATSCH [32] ont réussi à établir des bornes ser-

rées pour certaines classes des graphes notamment les graphes d'intervalles, les graphes sans astéroïde triples, les graphes fortement cordaux, les graphes distance-héréditaires et les cographes. Les algorithmes d'énumération établis pour ces classes des graphes sont tous en $\mathcal{O}^*(3^{n/3})$, à l'exception des cographes qui sont en $\mathcal{O}(m)$.

Récemment GOLOVACH, HEGGERNES, KRATSCH et SAEI [33], ont développé un algorithme qui énumère tous les ensembles dominants connexes minimaux dans les graphes cordaux en $\mathcal{O}(1.4736^n)$ en exploitant le nouveau théorème 10 des semi-simpliciaux du chapitre 3. Mais la borne inférieure reste toujours $3^{(n-2)/3}$ [32].

De même pour les graphes scindés et les cobipartis, le problème reste encore ouvert : Le meilleur algorithme connu est $\mathcal{O}(1.3674^n)$ [61] alors que la borne inférieure est 1.3195^n [17].

Toutefois, il est convient de noter que le meilleur algorithme connu pour les graphes bipartis est celui des graphes quelconques, c'est à dire $\mathcal{O}(2^{(1-\epsilon)n})$ avec $\epsilon > 10^{-50}$. De même, aucun algorithme d'énumération « input-sensitive » connu concernant cette classe de graphes est mieux que celui des graphes quelconques. Il semble alors que la structure de graphes bipartis ne serve pas beaucoup à la construction des algorithmes d'énumération.

Il est donc naturel de considérer dans ce chapitre une sous-classe de graphes bipartis comme les graphes bipartis convexes.

Dans ce chapitre, on propose un algorithme d'énumération des ensembles dominants connexes minimaux dans les graphes bipartis convexes en $\mathcal{O}(1.7254^n)$. D'une part, c'est déjà un progrès par rapport à la borne précédente, celle d'un graphe quelconque, qui est en $\mathcal{O}(2^{(1-\epsilon)n})$ où $\epsilon > 10^{-50}$ [54]. D'autre part, c'est le premier algorithme d'énumération « input-sensitive » établi pour une sous-classe non triviale de graphes bipartis.

Afin de varier, on a étudié aussi l'énumération d'un ensemble défini récemment : l'ensemble tropical connexe minimal. Un ensemble $S \subseteq V$ d'un graphe $G = (V, E)$, où ses sommets de V sont colorés, est dit tropical s'il contient toutes les couleurs du graphe G. De plus, G[S] est connexe et l'ensemble S est minimal, c'est-à-dire, il n'existe aucun autre ensemble topicial connexe $S' \subset V$ tel que $S' \subset S$.

Le graphe d'entrée G n'est pas nécessairement proprement colorié, c'est-à-dire, on peut avoir deux sommets adjacents qui ont la même couleur. Le problème dans sa version d'optimisation est prouvé NP-difficile dans les forêts, dans les graphes d'intervalles et dans les graphes scindés [20]. Le meilleur algorithme connu qui résout le problème d'optimisation dans les graphes quelconques est en $\mathcal{O}^*(1.5359^n)$ et en espace polynomial [13]. De même, les auteurs de ce dernier article ont proposé un algorithme d'optimisation pour les forêts en $\mathcal{O}^*(1.2721^n)$ [13]. Par contre, le problème dans sa version d'énumération n'a jamais été étudié. Alors on est les premiers à avoir étudié ce problème durant cette thèse ainsi que dans l'article [51]. Dans ce chapitre, on a proposé une borne inférieure de 1.4961^n mais sans réussir à casser la borne supérieure de 2^n. On a proposé des algorithmes d'énumération des ensembles tropicaux connexes minimaux dans les graphes scindés, cobipartis, les graphes d'intervalles, et les graphes blocs en $\mathcal{O}(1.6042^n)$, $\mathcal{O}^*(3^{n/3})$, $\mathcal{O}(1.8613^n)$ et $\mathcal{O}^*(3^{n/3})$ respectivement. On a établi une borne inférieure de 1.4766^n pour les graphes scindés et de $3^{n/3}$ pour les graphes cobipartis, les graphes d'intervalles et les graphes blocs.

4.1.1 Motivation

Contrairement à la contrainte de dominance, la connexité est une contrainte globale. Alors qu'il est bien connu qu'il est notoirement difficile de traiter les contraintes globales

même pour les problème d'optimisation. On peut citer par exemple le problème de voyageur de commerce, qui est probablement le plus populaire dans cette direction. Toutefois, aucun progrès n'a été réalisé depuis la publication de l'algorithme dynamique de BELLMAN [4] en 1962. On peut déduire alors que la conception des algorithmes d'énumération, afin d'établir des bornes supérieures, des ensembles connexes minimaux semble très difficile et très délicate. D'un autre côté, l'établissement des bornes inférieures sans union disjointe ne semble pas évident. Déjà les meilleures bornes connues à ce jour des ensembles dominants minimaux et des ensembles irredondants maximaux sont des unions disjointes d'octaèdres [24]et de C_5 [35] respectivement.

Dans la section suivante, on propose un algorithme d'énumération des ensembles dominants connexes minimaux dans les graphes bipartis convexes en $\mathcal{O}(1.7254^n)$.

4.2 Les ensembles dominants connexes minimaux dans les graphes bipartis convexes

Soit V = U ∪ W l'ensemble de sommets de G, où U et W définissent la bipartition des sommets.

Un graphe biparti G = (U, W, E) est dit convexe s'il existe un ordonnancement des sommets W tel que les voisins de chaque sommet $u \in$ U sont consécutifs en W.

Par commodité, on considère que U = {1, 2, ..., |U|}, W = $\{w_1, w_2, \ldots, w_{|W|}\}$, et que les sommets W sont donnés en entrée de l'algorithme selon l'ordre mentionné ci-dessus. De même, on dit que le sommet $w_i \in$ W est plus petit (plus grand) qu'un sommet $w_j \in$ W si l'entier i est plus petit (plus grand) que l'entier j.

Par définition d'un graphe biparti convexe, les adjacents d'un sommet $u \in$ U peuvent être représentés par un intervalle $I_u = [l(I_u), r(I_u)]$, appelé l'intervalle de voisinage de u, où $l(I_u)$ et $r(I_u)$ sont les sommets le plus petit et le plus grand, respectivement, dans l'intervalle de sommets de W adjacents à u. De plus, on appelle respectivement $l(I_u)$ et $r(I_u)$ les deux extrémités gauche et droite de l'intervalle I_u.

Alors, les voisins des sommets de U peuvent être représentés par un ensemble d'intervalles I(U). On appelle I(U) l'intervalle de représentation des voisins des sommets de U.

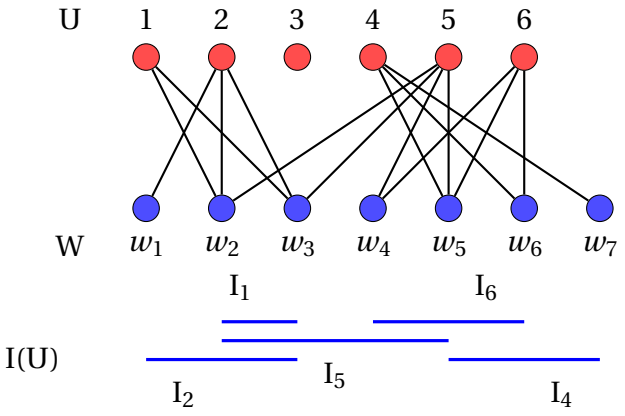

FIGURE 4.2 – Un graphe biparti convexe

Il est important de préciser, à l'occasion, que la reconnaissance d'un graphe biparti convexe se fait en temps linéaire [10].

4.2.1 Propriétés structurelles

Dans cette section, on fournit quelques propriétés utiles que tout ensemble dominant connexe minimal D d'un graphe biparti convexe G = (U, W, E) satisfait.

Observation 2. *Un graphe en étoile est un graphe biparti complet* $K_{1,n-1}$ *(voir la figure 4.3). C'est un arbre avec un noeud interne et* $n-1$ *feuilles. Un ensemble* $D \subseteq V$ *est un ensemble dominant connexe minimal d'un* $K_{1,n-1}$ *si et seulement si* D *est un singleton constitué du nœud interne de* $K_{1,n-1}$ *(les feuilles sont en bleu et le noeud interne est en rouge dans la figure 4.3). Par conséquent, si* G *est un graphe en étoile, qui est bien un graphe biparti convexe, alors le problème est résolvable en temps linéaire.*

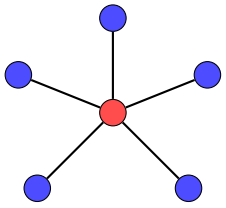

FIGURE 4.3 – Un graphe en étoile

A partir de maintenant, on considère qu'un graphe biparti convexe G = (U, W, E) a un ensemble $|W| \geq 2$ et un ensemble $|U| \geq 2$. Cela implique que chaque ensemble dominant connexe minimal de G est de cardinalité au moins égale à deux, i.e, $|D| \geq 2$.

On rappelle que les sommets W du graphe G = (U, W, E) sont donnés avec un ordre $w_1, w_2, \ldots, w_{|W|}$ satisfaisant que pour tout $u \in U$, $N(u) \subset W$ est un intervalle $I_u = [l(u), r(u)]$.

Observation 3. *Chaque sommet* $v \in V$ *a un voisin* $x \in D$.

Démonstration. Si $v \notin D$, comme D est un ensemble dominant, alors il existe un sommet $x \in D \cap N(v)$. Sinon $v \in D$, et comme D est un ensemble connexe tel que $|D| \geq 2$, alors il existe un sommet $x \in D \cap N(v)$. □

Lemme 13. *Si* $i, j \in U$ *tels que* $I_i \subseteq I_j$ *alors* $|\{i, j\} \cap D| \leq 1$.

Démonstration. Soient $i, j \in U$ tels que $I_i \subseteq I_j$ et D est un ensemble dominant connexe minimal. On suppose par contradiction que les deux sommets $i, j \in D$. Le sommet i ne peut pas avoir un voisin privé, par rapport à D, dans W car tous ses voisins sont dominés par j. De plus, i ne peut pas être un privé pour lui-même car D est connexe et $|D| \geq 2$. Par conséquent, i ne peut pas avoir un privé. On montre maintenant que le sommet i n'est pas un point d'articulation dans G[D]. Soient $x, y \in V \setminus \{i\}$. Pour tout chemin $x - y$ de G qui passe par i, il existe un chemin $x - y$ passant par j sans passer par i. Par conséquent, i ne peut pas être un point d'articulation de G[D]. Donc D n'est pas un ensemble dominant connexe minimal; une contradiction. □

Cela implique immédiatement le lemme suivant :

Lemme 14. *Si* $r(I_i) = r(I_j)$ *ou* $l(I_i) = l(I_j)$, $i, j \in U$, *alors* $|\{i, j\} \cap D| \leq 1$.

Les deux lemmes suivants sont cruciaux pour la conception de l'algorithme de branchement qu'on va détailler par la suite.

Lemme 15. *Pour tout* $i \leq |W| - 1$, *il existe un sommet* $u \in D \cap U$ *tel que* $u \in N(w_i) \cap N(w_{i+1})$.

Démonstration. On suppose par contradiction qu'il existe un $i \leq |W| - 1$ tel que $D \cap N(w_i) \cap N(w_{i+1}) = \emptyset$. On considère le sous-graphe induit $H = G[W \cup (U \cap D)]$. Ensuite, on montre que $H' = (U', W', E')$ et $H'' = (U'', W'', E'')$ tels que $U' = \{u' \in U : r(I_{u'}) \leq w_i\}$, $W' = \{w_{i'} \in W : w_{i'} \leq w_i\}$, $U'' = \{u'' \in U : l(I_{u''}) \geq w_{i+1}\}$ et $W'' = \{w_{i''} \in W : w_{i''} \geq w_i\}$, sont deux composantes connexes de $H = (U, W, E)$. Il n'est pas difficile de constater qu'il n'existe aucune arête $w_j u \in E$ tel que $w_j \in W', u \in U''$ ou $w_j \in W'', u \in U'$, sinon $u \in N(w_i) \cap N(w_{i+1})$. Par conséquent, les deux sous-graphes induits H' et H'' sont en effet deux composantes connexes de H. Par l'observation 3, $N(w_i) \cap D \neq \emptyset$ et $N(w_{i+1}) \cap D \neq \emptyset$, donc $D \cap U' \neq \emptyset$ et $D \cap U'' \neq \emptyset$. Par conséquent, G[D] est non connexe ; une contradiction. □

D'une façon similaire, on peut déduire le lemme suivant :

Lemme 16. *Pour deux consécutifs $w_i, w_j \in W \cap D$ dans G[D] (pas nécessairement consécutifs dans W), il existe un $u \in D \cap U$ tel que $u \in N(w_i) \cap N(w_j)$.*

Lemme 17. *Soient $i, j \in U$ tels que $I_i \cap I_j \neq \emptyset$. S'il existe un $k \in U$ tel que $I_k \subset (I_i \cup I_j)$ et $i, j, k \in D \cap U$, alors $I_i \cap I_j \cap D = \emptyset$.*

Démonstration. Soient $i, j, k \in D \cap U$, $I_i \cap I_j \neq \emptyset$ et $I_k \subset \{I_i \cup I_j\}$. On suppose par contradiction qu'il existe un $w \in D \cap I_i \cap I_j$. Il est clair que le sommet k est un point d'articulation de G[D]. Par conséquent, il y a $w_x, w'_x \in D \cap I_k$ et w_x, w'_x appartiennent à deux composantes connexes différentes de $G[D \setminus k]$. Sans perte de généralité, on suppose que $w_x \in I_i \cap I_k \setminus I_j$ et $w'_x \in I_j \cap I_k \setminus I_i$. Toutefois, il existe un chemin $w_x - w'_x$ passant par i, w, j sans passer par k ; une contradiction. □

Lemme 18. *Soient $i, j \in D \cap U$ tels que $I_i \cap I_j \neq \emptyset$. Alors $|\{k \in D : I_k \subset (I_i \cup I_j)\}| \leq 1$.*

Démonstration. Soient $i, j \in D$, $I_i \cap I_j \neq \emptyset$. On suppose par contradiction qu'il existe $k, l \in D$ satisfaisant $I_k \subset \{I_i \cup I_j\}$ et $I_l \subset \{I_i \cup I_j\}$. Par les lemmes 14 et 13 et sans restreindre la généralité, on suppose que $l(I_i) < l(I_k) < l(I_l) < l(I_j)$ et $r(I_i) < r(I_k) < r(I_l) < r(I_j)$. Il est clair que les sommets k, l sont des points d'articulations de G[D].

Comme k est un point d'articulation de G[D], alors il existe $w_x, w_{x'} \in D \cap I_k$ et w_x, w'_x appartiennent à deux composantes différentes de $G[D \setminus k]$. De plus, au moins l'un de deux sommets n'appartient pas à I_l. Sans perte de généralité, soit $w_x < w_{x'}$ et $w_x \notin I_l$.

Comme l est un point d'articulation de G[D], alors il existe $w_y, w_{y'} \in D \cap I_l$ et w_y, w'_y appartiennent à deux composantes différentes de $G[D \setminus l]$. Il est clair que l'un de $w_y, w_{y'}$ est non adjacent à k. Sans perte de généralité, on suppose que $w_{y'} > w_y$ et $w_{y'} > r(I_k)$.

De plus, $w_y < l(I_j)$ et $w_y > w_x$ sinon $w_x, w_{x'}$ seront adjacents à l et $w_y, w_{y'}$ seront adjacents à j. On peut déduire aussi que $w_{x'} > r(I_i)$ (voir la figure 4.4). Par conséquent, il existe un chemin $w_y - w_{y'}$ passant par $k, w_{x'}, j$ et sans passer par l ; une contradiction. □

Dans la section suivante, on présente l'algorithme d'énumération des ensembles dominants connexes minimaux dans les graphes bipartis convexes qui se base sur les propriétés structurelles ci-dessus.

4.2.2 L'algorithme d'énumération

L'algorithme d'énumération consiste à sélectionner les sommets d'un ensemble dominant connexe minimal D par des règles de réduction et de branchement. L'algorithme

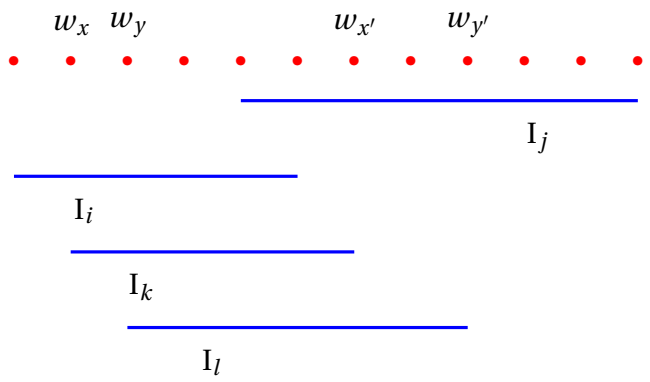

FIGURE 4.4 – Figure explicative la preuve du lemme 18

est séparé sur différentes phases. Pendant le prétraitement (phase 1), une collection d'appels récursifs d'initialisation est effectuée. Pendant la phase 2, on décide parmi les sommets de U lesquels appartiennent à D. Autrement dit, on fixe $D \cap U$. Par conséquent, lorsqu'on sélectionne un sommet $u \in U$, on l'ajoute immédiatement à l'ensemble dominant connexe minimal potentiel S. Cependant, lorsqu'on écarte un sommet u de D, on le déplace à un ensemble T afin de le dominer dans les prochaines étapes de l'algorithme par un sommet (à sélectionner) de W. En outre, lorsqu'on fixe les sommets de $D \cap U$, on marque certains sommets de W comme interdits en les ajoutant à F. Ce qui signifie que ces sommets de $W \cap F$ ne peuvent pas être sélectionnés (ils sont exclus de l'ensemble D). A la phase 3, on élimine principalement toute solution partielle de la phase 2 qui ne peut pas être étendue à un ensemble dominant connexe minimal de G. Enfin à la phase 4, les solutions partielles restantes sont complétées, si possible, en des ensembles dominants connexes minimaux.

Phase 1. Prétraitement.

On considère la procédure suivante ENUMLEVEL1(U,W), où on initialise l'appel de ENUMLEVEL2(u,U,S,T,F).

Etape 1. pour chaque $u \in N(w_1)$, appelez ENUMLEVEL2(u, $U \setminus N(w_1)$, $\{u\}$, $N(w_1) \setminus \{u\}$, $\emptyset$).

Par l'observation 3, $|N(w_1) \cap D| \geq 1$. Puisque tous les voisins de w_1 ont la même extrémité gauche, alors $|N(w_1) \cap D| \leq 1$ par le lemme 14. Donc $|N(w_1) \cap D| = 1$. Par conséquent, on ajoute exactement un sommet $u \in N(w_1)$ à S et on écarte les autres sommets $N(w_1) \setminus \{u\}$ de la solution. C'est à dire, on les déplace à l'ensemble T afin de les dominer par les sommets de W dans les prochaines étapes de l'algorithme. Enfin, on initialise $F = \emptyset$ et on appelle ENUMLEVEL2(u,U,S,T,F).

Phase 2.

On considère la procédure récursive suivante ENUMLEVEL2(u,U,S,T,F), où u est le sommet déjà sélectionné dans $S \cap U$ qui a la plus grande extrémité droite et U,S,T,F sont mentionnés ci-dessus.

Par commodité, dans la description de l'algorithme on désigne $r(I_u)$ par r.

Etape 1. S'il existe un sommet $i \in U$ tel que $r(I_i) \leq r$ ou $I_u \subseteq I_i$, alors appelez ENUMLEVEL2(u, $U \setminus \{i\}$, S, $T \cup \{i\}$, F).

Etape 2. Si $r = w_{|W|}$, alors appelez ENUMLEVEL3(S,T,F).

Etape 3. Si $N(r) = \emptyset$, alors arrêtez.

Etape 4. Si $deg(r) = 1$, alors soit $\{j\} = N(r)$ et appelez EnumLevel2$(j, U \setminus \{j\}, S \cup \{j\}, T, F)$.

Etape 5. Si $deg(r) = 2$, alors soit $\{j,k\} = N(r)$ et branchez :
Si $I_j \subseteq I_k$ ou $I_k \subseteq I_j$, alors branchez :

(i) appelez EnumLevel2$(j, U \setminus \{j,k\}, S \cup \{j\}, T \cup \{k\}, F)$,

(ii) appelez EnumLevel2$(k, U \setminus \{j,k\}, S \cup \{k\}, T \cup \{j\}, F)$.

Sinon, soit $r(j) > r(k)$ et branchez :

(i) appelez EnumLevel2$(j, U \setminus \{j,k\}, S \cup \{j\}, T \cup \{k\}, F)$,

(ii) appelez EnumLevel2$(k, U \setminus \{j,k\}, S \cup \{k\}, T \cup \{j\}, F)$,

(iii) appelez EnumLevel2$(j, U \setminus \{j,k\}, S \cup \{j,k\}, T, F \cup (I_j \cap I_u))$.

Etape 6. Si $deg(r) \geq 3$, alors soit j l'adjacent de r qui a la plus grande extrémité droite et branchez :

(i) appelez EnumLevel2$(j, U \setminus N(r), S \cup \{j\}, T \cup N(r) \setminus \{j\}, F)$,

(ii) pour chaque $x \in N(r)$ tel que $l(I_u) < l(I_x) < l(I_j)$ et $r(I_u) < r(I_x) < r(I_j)$, appelez EnumLevel2$(j, U \setminus N(r), S \cup \{x,j\}, T \cup N(r) \setminus \{x,j\}, F \cup \{I_j \cap I_u\})$,

(iii) appelez EnumLevel2$(u, U \setminus \{j\}, S, T \cup \{j\}, F)$.

Soit j le voisin de r avec la plus grande extrémité droite. Soit $j \in D$ ou $j \notin D$.
Si $j \notin D$, alors dans le cas (iii), on ajoute j à T afin de le dominer dans les prochaines étapes par les sommets de W. Par conséquent, on appelle EnumLevel2$(u, U \setminus \{j\}, S, T \cup \{j\}, F)$.
On suppose maintenant que $j \in D$. Si $N(r) \cap D \setminus \{u,j\} = \emptyset$, alors dans le cas (i), on appelle EnumLevel2$(j, U \setminus N(r), S \cup \{j\}, T \cup N(r) \setminus \{j\}, F)$. On suppose maintenant que $N(r) \cap D \setminus \{u,j\} \neq \emptyset$. Dans ce cas, comme l'étape 1 est non applicable et par les lemmes 13 et 14, on a pour tout $x \in D \cap N(r) \setminus \{u,j\}$, $l(I_u) < l(I_x) < l(I_j)$ et $r(I_u) < r(I_x) < r(I_j)$. Comme $I_x \subset (I_u \cup I_j)$ et par le lemme 18, $|N(r) \cap D \setminus \{u,j\}| \leq 1$, alors on a $|N(r) \cap D \setminus \{u,j\}| = 1$. On suppose que $N(r) \cap D \setminus \{u,j\} = \{x\}$. Dans ce cas, on interdit par le lemme 17 $(I_j \cap I_u)$. Par conséquent, on branche pour chaque $x \in N(r)$ satisfaisant $l(I_u) < l(I_x) < l(I_j)$ et $r(I_u) < r(I_x) < r(I_j)$ et on appelle EnumLevel2$(j, U \setminus N(r), S \cup \{x,j\}, T \cup N(r) \setminus \{x,j\}, F \cup \{I_j \cap I_u\})$.
S'il existe un $x \in U$ tel que $I_x \subseteq I_j$, alors x sera traité par l'étape 1 de l'appel récursif.

Phase 3.

A cette phase, la procédure EnumLevel3(S, T, F) écarte les solutions partielles générées durant la phase précédente mais qui ne peuvent pas être étendues à des ensembles dominants connexes minimaux de G. Aussi la procédure prétraite les solutions partielles restantes pour la phase suivante.

Soit $J(T, S)$ un ensemble d'intervalles, où pour chaque intervalle il faut sélectionner au moins un sommet soit pour dominer un sommet de T, soit pour connecter des sommets de S.

Par conséquent, on initialise $J(T, S)$ par des intervalles correspondant aux voisins des sommets de T et on considère la graphe $G[W \cup S]$. Soit I' l'ensemble des intervalles correspondant aux voisins des sommets de S.

Etape 1. S'il existe $u, v \in S \cap U$ tels que $I'_u \subseteq I'_v$, alors arrêtez.

Etape 2. pour tout $w \in W$
Faire
Si $deg(w) \geq 4$, alors alors arrêtez.

Si $deg(w) = 3$, alors $F \leftarrow F \cup \{w\}$.
fin faire.

Etape 3. Tant que ($I' \neq \emptyset$)
Debut

Soit I'_i l'intervalle qui a l'extrémité droite le plus à gauche dans I' et on désigne $r(I'_i)$ par r.

Si $I' = \{I'_i\}$, alors appelez EnumLevel4(S, J, F).

Sinon si $deg(r) = 2$, alors soit $N(r) = \{i, j\}$:
$J(T,S) \leftarrow J(T,S) \cup \{I'_i \cap I'_j\}$;
$I' \leftarrow I' \setminus I'_i$.

Sinon si $deg(r) = 3$, alors soit $N(r) = \{i, j, k\}$, I'_j l'intervalle qui a la plus grande extrémité et I'_k l'autre intervalle tel que $I'_k \subset (I'_i \cup I'_j)$:
$J(T,S) \leftarrow J(T,S) \cup \{I'_i \cap I'_k \setminus I'_j\} \cup \{I'_j \cap I'_k \setminus I'_i\}$;
$I' \leftarrow I' \setminus I'_i$.

Fin.

La procédure EnumLevel3(S, T, F) élimine, en particulier à l'étape 1 et 2, les solutions partielles S qui ne peuvent pas être étendues à des ensembles dominants connexes minimaux de G. A l'étape 1, s'il existe $u, v \in S \cap U$ tels que $I'_u \subseteq I'_v$, alors on s'arrête car par le lemme 13, la solution partielle $S \not\subset D$. A l'étape 2, on vérifie chaque sommet $w \in W$. Si $deg(w) \geq 4$ dans $G[W \cup (U \cap S)]$, alors par le lemme 18, la solution partielle $S \not\subset D$. Sinon si $deg(w) = 3$, alors le sommet w est interdit par le lemme 17.

La procédure EnumLevel3(S, T, F) prétraite aussi les solutions partielles pour la phase suivante en construisant J(T, S).

Dans un premier temps, chaque sommet de T devrait être dominé par au moins un de ses voisins. Alors, on a initialisé J(T, S) par un ensemble d'intervalles où les voisins de chaque sommet de T sont représentés par un intervalle.
Ensuite, à l'étape 3, on ajoute les autres ensembles d'intervalles nécessaires en parcourant l'ensemble des intervalles I' du sous-graphe induit $G[W \cup S]$. On prend chaque fois l'intervalle I'_i qui a l'extrémité droite le plus à gauche dans I' et on désigne $r(I'_i)$ par r. Si $deg(r) = 1$, c'est à dire $I' = \{I'_i\}$, alors on a terminé le parcours et on appelle la procédure EnumLevel4(S, J, F). Comme $deg(r) \leq 3$, à cause de l'étape 2, alors il ne nous reste que deux possibilités : soit $deg(r) = 2$, soit $deg(r) = 3$. Si $deg(r) = 2$, on suppose par contradiction que $I'_i \cap I'_j \cap D = \emptyset$. Par l'observation 3, il y a un $w_i \in I'_i \cap D$ et un $w_j \in I'_j \cap D$. Sans perte de généralité, on suppose que w_i, w_j sont consécutifs dans G[D]. Comme $deg(r) = 2$, alors on ne peut pas avoir un $u \cap U \cap D$ tel que $u \in N(w_i) \cap N(w_j)$, une contradiction avec le lemme 16. Donc on peut déduire que $I'_i \cap I'_j \cap D \neq \emptyset$. Par conséquent, $J(T,S) \leftarrow J(T,S) \cup \{I'_i \cap I'_j\}$.
Maintenant si $deg(r) = 3$, alors les sommets de $I'_i \cap I'_j$ sont interdits et k est un point d'articulation de G[D] qui relie deux sommets dans $W \cap D$. Si ces deux sommets appartiennent les deux à $(I'_i \cap I'_k \setminus I'_j)$ ou à $(I'_j \cap I'_k \setminus I'_i)$, alors k n'est pas un point d'articulation. Donc on a un premier sommet appartient à $(I'_i \cap I'_k \setminus I'_j)$ et l'autre à $(I'_j \cap I'_k \setminus I'_i)$. Par conséquent, on a ajouté $(I'_i \cap I'_k \setminus I'_j)$ et $(I'_j \cap I'_k \setminus I'_i)$ à J(T, S).

Phase 4.

On considère la procédure récursive suivante appelée ENUMLEVEL4(W, J, D). A ce niveau, on essaie de sélectionner au moins un sommet de chaque intervalle de J afin de dominer les sommets de T et de connecter S par des sommets de W. En d'autres termes, on fixe à cette phase $D \cap W$.

Soit $J_i \in J$ l'intervalle avec l'extrémité droite la plus à gauche. Si on a plus d'un intervalle candidat, alors on choisit l'intervalle le plus court.

Etape 1. Si $J = \emptyset$, alors vérifiez si S est un ensemble dominant connexe minimal dans G et le retournez si le cas ; arrêtez.

Etape 2. Si tous les sommets de J_i sont interdits, alors arrêtez.

Etape 3. Pour chaque sommet non interdit $w \in J_i$, appelez ENUMLEVEL4$(W \setminus J_i, J \setminus \{J_k : w \in J_k\}, S \cup \{w\})$.

Par la construction de J, on doit sélectionner au moins un sommet de chaque intervalle de J. Donc $|D \cap J_i| \geq 1$. Par conséquent, à l'étape 2, on s'arrête lorsque tous les sommets de J_i sont interdits. On montre maintenant que $|D \cap J_i| \leq 1$. On suppose par contradiction que $|D \cap J_i| \geq 2$. Alors, il existe $w_i, w_j \in J_i \cap D$. Sans restreindre la généralité, on suppose que $w_i < w_j$. Dans ce cas, pour tout intervalle $J_k \in J$, si $w_i \in J_k$ alors $w_j \in J_k$. Par conséquent, D n'est pas minimal car si D est un ensemble dominant connexe alors $D \setminus \{w_i\}$ est un ensemble dominant connexe aussi ; une contradiction. D'où $|D \cap J_i| \leq 1$ et parce que $|D \cap J_i| \geq 1$ alors $|D \cap J_i| = 1$.

4.2.3 L'analyse du temps d'exécution

Afin d'analyser le temps d'exécution de l'algorithme, on associe à chaque règle de branchement des procédures ENUMLEVEL4(W, J, D) et ENUMLEVEL2(u, U, D, T, F) un vecteur de branchement puis on calcule son facteur. On note que ENUMLEVEL1 et ENUMLEVEL3 s'exécutent en temps polynomial car aucun branchement n'a été appliqué. La mesure d'une instance est définie par $|U| + |W \setminus F|$. Par conséquent, en déplaçant un sommet $u \in U$ à T ou en interdisant un sommet $w \in W$, la mesure de l'instance diminue par 1.

En premier lieu, on analyse la procédure ENUMLEVEL2(u, U, D, T, F). On note que dans les étapes 1−4, on réduit l'entrée sans brancher (règles de réduction). Par conséquent, afin d'analyser le temps d'exécution, il suffit d'analyser uniquement les étapes 5 et 6 (règles de branchement).

Etape 5. Le premier vecteur de branchement à l'étape 5 est (2,2). Alors que pour la deuxième règle de branchement, dans le pire des cas, on n'interdit qu'un seul sommet de W. Ainsi le vecteur de branchement dans ce cas est (2,2,3). Par conséquent, le facteur maximum est atteint pour le vecteur (2,2,3) qui est strictement inférieur à 1.6181.

Etape 6. Le facteur de branchement atteint son maximum si $|I_u \cap I_j| = 1$ et s'il n'existe aucun $x \in N(r)$ tel que $I_x \subseteq I_j$. Ainsi on branche pour tous les voisins de r et on interdit dans chaque branchement, le seul sommet $I_u \cap I_j$. Le vecteur de branchement correspondant est $(t, \underbrace{t+1, \ldots, t+1}_{t-1}, 1)$ où $t = deg(r) \geq 3$, et la valeur maximale de facteur de branchement est inférieure à 1.7254, qui est obtenu pour $t = 3$.

Pour ENUMLEVEL4(W, J, D), on a besoin uniquement d'analyser l'étape 3 parce que les étapes 1 et 2 sont des règles de réduction. Le vecteur de branchement correspondant est

$(\underbrace{t,\ldots,t}_{t})$ où $t = |J_i \setminus F|$ et la valeur maximale de son facteur est inférieure à 1.4423, qui est obtenu pour $t = 3$.

Le plus grand facteur de branchement est majoré par 1.7254. Cela nous permet de conclure immédiatement le théorème suivant.

Théorème 14. *Un graphe biparti convexe de n sommets a au plus* 1.7254^n *ensembles dominants connexes minimaux qu'on peut énumérer en* $\mathcal{O}(1.7254^n)$.

Afin de compléter la borne supérieure obtenue par le théorème 14, on construit dans la section suivante une borne inférieure de $3^{(n-2)/3}$.

4.2.4 Borne inférieure

Pour obtenir une borne inférieure des ensembles dominants connexes minimaux dans les graphes bipartis convexes, on a modifié légèrement la borne inférieure des graphes cordaux décrite dans l'article [32].

Lemme 19. *Il existe des graphes bipartis convexes avec au moins* $3^{(n-2)/3}$ *ensembles dominants connexes minimaux.*

Démonstration. Soit k un entier positif et impair. Pour obtenir une borne inférieure des ensembles dominants connexes minimaux dans les graphes bipartis convexes, on construit un graphe G de la manière suivante :

— Pour $i \in \{1,\ldots,k\}$, construisez un stable d'un triplet de sommets $T_i = \{x_i, y_i, z_i\}$.

— Pour $i \in \{2,\ldots,k\}$, reliez par une arête chaque sommet de T_{i-1} avec chaque sommet de T_i.

— Ajoutez deux sommets u et v, ainsi que les arêtes ux_1, uy_1, uz_1, vx_k, vy_k et vz_k.

Clairement, G a $n = 3k+2$ sommets. On note que $D \subseteq V(G)$ est un ensemble dominant connexe minimal de G si et seulement si $u, v \notin D$ et $|D \cap T_i| = 1$ pour $i \in \{1,\ldots,k\}$. Par conséquent, G a $3^k = 3^{(n-2)/3}$ ensembles dominants connexes minimaux. Il reste à observer, à travers son modèle ci-dessous, que G est un graphe biparti convexe.

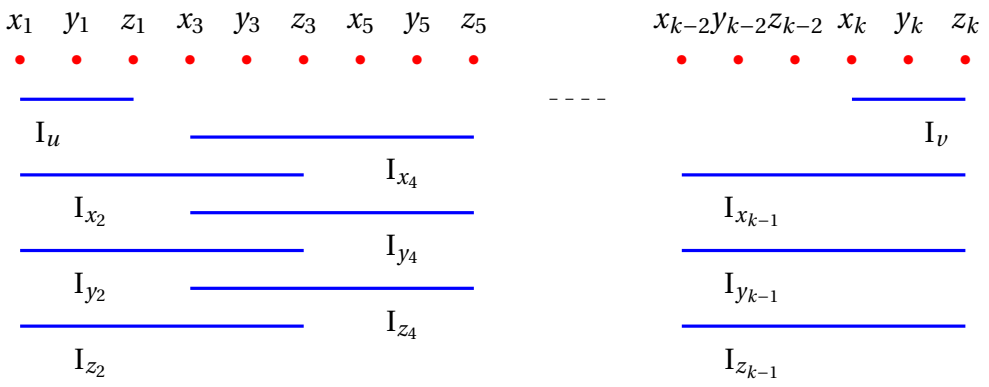

FIGURE 4.5 – La borne inférieure des ensembles dominants connexes minimaux dans les graphes bipartis convexes

Il est clair que le graphe G est biparti vu qu'il existe une partition de son ensemble de sommets en deux sous-ensembles stables U et W tels que l'ensemble U est constitué par u et v, ainsi que les T_i d'indices pairs alors que W est constitué par les T_i d'indices impairs. De même, il existe un ordonnancement, comme l'illustre la figure 4.5, des sommets W tel que les voisins de chaque sommet $u \in U$ sont consécutifs dans W. □

Dans la section suivante, on a étudié l'énumération des ensembles tropicaux connexes minimaux.

4.3 Les ensembles tropicaux connexes minimaux

On a étudié dans la première sous-section l'énumération des ces ensembles dans les graphes quelconques.

4.3.1 Les ensembles tropicaux connexes minimaux dans les graphes quelconques

La borne supérieure

On n'a pas réussi à avoir un algorithme mieux que celui de la force brute. L'algorithme 5 énumère tous les sous-ensembles des sommets et vérifie en temps polynomial si l'ensemble est un tropical connexe minimal.

Algorithme 5 : EnumTCS(G, S)

Données : Un graphe G = (V, E) colorié et un ensemble S
Résultat : Tous les ensembles tropicaux connexes minimaux
si S *est un ensemble tropical connexe minimal* **alors**
  retourner S et arrêtez
fin
Sélectionner un sommet v quelconque du G
retourner EnumTCS(G − v, S ∪ {v})
retourner EnumTCS(G − v,S)

Donc on peut énumérer en $\mathcal{O}^*(2^n)$ tous les ensembles tropicaux connexes minimaux dans un graphe G = (V, E) colorié en appelant EnumTCS(G, ∅). Au contraire de la borne supérieure, on a réussi à établir une nouvelle borne inférieure non triviale décrite ci-dessous.

La borne inférieure

On construit la borne inférieure de la manière suivante : Soit T un arbre de hauteur h dans lequel la racine est de degré 5 et chaque sommet interne a 4 fils. Après, on remplace la racine et chaque sommet interne v par un K_5 et on le désigne par K_5^v. De même, chaque feuille dans T est remplacée par un K_3. Pour chaque K_5^v, il existe une bijection entre les sommets de K_5 et les 5 voisins de v dans l'arbre T.

Soit v un sommet interne de l'arbre T et u son parent. Soit K_5^v (respectivement K_5^u) le K_5 associé à v (respectivement u). Soit x_u l'unique sommet de K_5^v associé dans la bijection au parent u de v. Et soit x_v respectivement l'unique sommet de K_5^u associé dans la bijection au fils v de u. Alors on ajoute une arête entre toute paire de sommets (y, z) tels que $y \in K_5^v$, $z \in K_5^u$, $y \neq x_u$ et $z \neq x_v$ (Voir la figure 4.6). Soit v une feuille et u son parent dans T, on note K_3^v le K_3 associé à v et par K_5^u le K_5 associé à u. On ajoute une arête entre toute paire de sommets (y, z) tels que $y \in K_5^u$, $z \in K_3^v$ et $y \neq x_v$, où x_v est le sommet de K_5^u associé dans la bijection au fils v de u.

Concernant les couleurs, on associe à tous les sommets d'un K_5 (respectivement k_3) une unique couleur. Autrement dit deux sommets ont la même couleur si et seulement s'ils appartiennent à la même clique. Comme chaque clique a une couleur unique dans

cette famille de graphe, alors chaque ensemble tropical connexe minimal contient au moins un sommet de chaque clique (la contrainte de la tropicalité).

De plus, pour chaque K_5^v on doit sélectionner plutôt deux sommets sinon l'ensemble tropical ne peut pas être connexe vu qu'il n'existe aucune arête entre le sommet sélectionné, soit x, et les sommets de K_5 qui est associé dans la bijection à x. Par conséquent, on doit sélectionner un autre sommet y, autre que x, dans chaque clique K_5^v. Concernant les sommets de K_3, on doit sélectionner exactement un sommet dans chaque K_3, car toute clique K_3 a une couleur unique dans le graphe.

On calcule maintenant le nombre des ensembles tropicaux connexes minimaux dans le graphe décrit ci-dessus. On rappelle que l'hauteur de l'arbre T est h. Au niveau $i = 0$, il n'y a qu'un seul K_5 (correspondant à la racine), et deux sommets doivent être sélectionnés de cet ensemble. Pour chaque i, $1 \leq i \leq h$, le nombre de K_5 ou K_3 au niveau i est $5 \cdot 4^{i-1}$.

Par conséquent, le nombre total des sommets est :

$$n(h) = 5 + \sum_{i=1}^{h-1} 5^2 \cdot 4^{i-1} + 5 \cdot 4^{h-1} \cdot 3.$$

Alors que le nombre des ensembles tropicaux connexes minimaux est :

$$\mathrm{TCS}(h) = \binom{5}{2} \cdot \binom{5}{2}^{\sum_{i=1}^{h-1} 5 \cdot 4^{i-1}} \cdot 3^{5 \cdot 4^{h-1}}$$

Pour $h \geq 9$, on a $\mathrm{TCS}(h)^{1/n(h)} > 1.4916$. Donc on peut déduire le lemme suivant :

Lemme 20. *Le nombre maximum des ensembles tropicaux connexes minimaux est au moins* 1.4916^n.

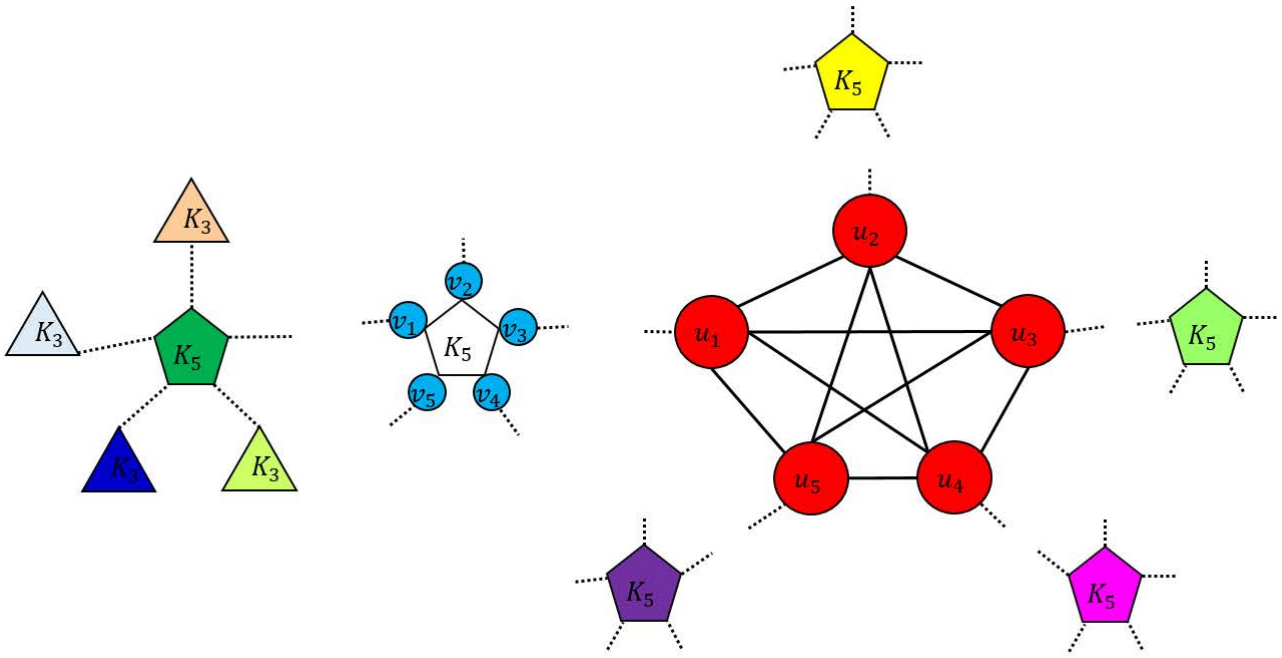

FIGURE 4.6 – La borne inférieure des ensembles tropicaux connexes minimaux

Dans la section suivante, on a étudié les ensembles tropicaux connexes minimaux dans les graphes cordaux.

4.3.2 Les ensembles tropicaux connexes minimaux dans les graphes cordaux

On n'a pas réussi à concevoir un algorithme mieux que l'algorithme 5 qui s'exécute en $\mathcal{O}^*(2^n)$. De même, on prouvera que le graphe décrit dans la figure 4.6 est un graphe cordal. On rappelle qu'on a prouvé avec ce graphe que la borne inférieure des ensembles tropicaux connexes minimaux est au moins 1.4916^n. On rappelle aussi qu'un graphe est dit cordal si chacun de ses cycles de quatre sommets ou plus possède une corde. En d'autres termes, chaque cycle induit dans un graphe cordal a au plus trois sommets.

Démonstration. Soit G = (V, E), le graphe décrit dans la section 4.3.1. On suppose, par l'absurde, que G n'est pas cordal. En conséquence, il est juste de dire qu'il existe un cycle induit C de longueur d'au moins 4. Par construction de G, chaque sommet $z \in C$ appartient à une clique (soit un K_5 soit un K_3) qui correspond à un sommet v de l'arbre T. Etant donné un tel sommet z, on désigne par $f(z)$, son sommet correspondant v dans T.
Soit $x \in C$ tel que $f(x)$ est le sommet le plus profond (i.e. le plus bas) de l'arbre T. Il est clair que $f(x)$ n'est pas la racine de T, sinon C est une clique. Donc $f(x)$ a un parent dans T. Soit $v = f(x)$ et soit u le parent de v dans T.
On note bien que le cycle C ne peut pas avoir trois sommets consécutifs x, y, z tels que $f(x) = f(y) = f(z)$, sinon dans ce cas les sommets x, y, z appartiennent à la même clique, et par conséquent C n'est pas un cycle induit.
On suppose tout d'abord que le sommet consécutif de x dans C est le sommet y tel que $f(x) = f(y)$. Par définition de C, on a $C = (\ldots, a, x, y, b, \ldots)$ où $f(a) = f(b) = u$ parce que $f(x)$ est le sommet le plus profond de l'arbre T et u est son parent. Mais, par construction de G, les sommets a et b appartiennent au même K_5 (ou K_3) et donc ils sont adjacents. Cela contredit le fait que C est induit.
On suppose maintenant qu'il n'existe ni un prédécesseur ni un successeur y à x dans C tel que $f(x) = f(y)$. Avec les mêmes arguments, on a $C = (\ldots, a, x, b, \ldots)$ tel que $f(a) = f(b) = u$. Cela contredit aussi le fait que C est induit.
Par conséquent, tout cycle induit dans G est de longueur 3 et donc G est un graphe cordal. □

Cette dernière démonstration, nous permet de déduire le lemme suivant :

Lemme 21. *Le nombre maximum des ensembles tropicaux connexes minimaux dans les graphes cordaux est au moins* 1.4916^n.

Dans la section suivante, on a étudié les ensembles tropicaux connexes minimaux dans les graphes scindés.

4.3.3 Les ensembles tropicaux connexes minimaux dans les graphes scindés

La borne supérieure

Un graphe G = (K, I, E) est dit scindé si l'ensemble de ses sommets V peut être partitionné en une clique K et un stable I. On note bien que les graphes scindés sont des graphes cordaux (voir le diagramme 1.9 du chapitre 1).

KRATSCH, LIEDLOFF et SAYADI [51] ont proposé un algorithme d'énumération des ensembles tropicaux connexes minimaux dans les graphes scindés qui s'exécute en $\mathcal{O}(1.6042^n)$.

L'idée principale de l'algorithme est que lorsqu'on sélectionne un sommet $x \in K$, on l'ajoute immédiatement à S où S est une solution potentielle d'un ensemble tropical connexe minimal. Cependant, lorsqu'on sélectionne $y \in I$, on le déplace d'abord vers un ensemble intermédiaire T, et on ne l'ajoute à S qu'uniquement lorsque au moins l'un de ses voisins est sélectionné. De cette façon, on garantit que G[S] est connexe.

Afin d'optimiser le temps d'exécution, les auteurs de l'article [51] ont analysé le temps d'exécution avec la technique Mesurer pour Conquérir. Ils ont associé à l'instance du problème (K, I, T, S) une mesure de $|K| + |I| + |T| \cdot w$, où $w = 0.533244$. Autrement dit, si on sélectionne un sommet de la clique K, la mesure de l'instance diminue de 1. Par contre, si on déplace un sommet du stable I à T, on gagne uniquement 0.466756. Puis on gagne le reste 0.533244 une fois on déplacera le sommet de T à S. Avec la technique décrite ci-dessus, les auteurs de l'article ont prouvé le théorème suivant :

Théorème 15 ([51])**.** *Un graphe scindé de n sommets a au plus* 1.6042^n *ensembles tropicaux connexes minimaux qu'on peut énumérer en* $\mathcal{O}(1.6042^n)$.

La borne inférieure

On considère un graphe scindé $G = (K, I, E)$ où $K = \{v_i : 1 \leq i \leq 5\}$ est une clique et $I = \{x_i, y_i, z_i : 1 \leq i \leq 5\}$ est un stable. En plus des arêtes de la clique K, chaque sommet v_i est relié par tous les sommets x_j, y_j, z_j, tels que $i, j \in \{1, \ldots, 5\}$ et $i \neq j$ (voir la figure 4.7). Pour des raisons de simplification, on a modélisé une clique par un rectangle. De plus, on n'a pas modélisé les arêtes qui relient les sommets de l'ensemble K par les sommets de l'ensemble I, mais à l'inverse, on a modélisé par une ligne pointillée toute paire de sommet (v, w) telle que v est un sommet de K, w est un sommet de I et v, w sont non adjacents.

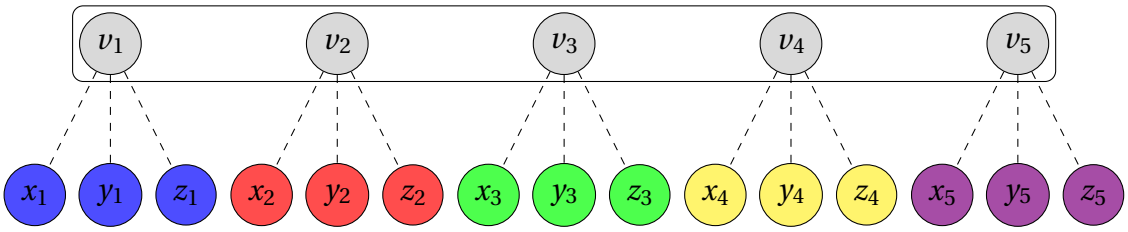

FIGURE 4.7 – La borne inférieure des ensembles tropicaux connexes minimaux dans les graphes scindés

Puis, on a colorié le graphe de la figure 4.7 de la façon suivante :

— On a associé à chaque sommet v_i tel que $i \in \{1, \ldots, 5\}$ la couleur c_0.

— Pour tout $j \in \{1, \ldots, 5\}$, on a associé la couleur c_j aux sommets x_j, y_j, z_j.

On calcule maintenant le nombre des ensembles tropicaux connexes minimaux dans ce graphe où chaque ensemble tropical connexe minimal S est obtenu par la sélection d'un unique sommet de l'ensemble $\{x_j, y_j, z_j\}$, ainsi que exactement deux sommets de la clique K pour que l'ensemble S soit connexe. Par conséquent, le nombre des ensembles tropicaux connexes minimaux dans le graphe G est $3^5 \cdot \binom{5}{2} = 2430$.

On considère maintenant un graphe scindé de n sommets, où $n = 20 \cdot k$ pour un entier k. Le graphe est construit à partir de k copies disjointes de G, chaque copie a ses couleurs distinctes, et tous les sommets de chaque clique sont transformés en une unique grande clique. Ainsi, un tel graphe a au moins $2430^{n/20}$ ensembles tropicaux connexes minimaux qui nous permet de déduire le lemme suivant :

Lemme 22. *Il existe des graphes scindés avec au moins* 1.4766^n *ensembles tropicaux connexes minimaux.*

4.3.4 Les ensembles tropicaux connexes minimaux dans les graphes cobipartis

Un graphe $G = (K_1, K_2, E)$ est dit cobiparti si l'ensemble de ses sommets V peut être partitionné en deux cliques K_1 et K_2. Soit S un ensemble tropical connexe minimal. Chaque ensemble tropical connexe minimal S du graphe G appartient à l'un des deux types suivants :

— Type A : L'ensemble $S \subseteq K_i$ tel que $i \in \{1,2\}$.

— Type B : Pour tout $i \in \{1,2\}$, $K_i \cap S \neq \emptyset$.

Algorithme d'énumération

On peut énumérer tous les ensembles tropicaux connexes minimaux du graphe G à l'aide de l'algorithme suivant :

Type A : Pour chaque $i \in \{1,2\}$ tel que la clique K_i (tropical) contient toutes les couleurs, on pique exactement un unique sommet v de chaque couleur.

Type B : Pour toute arête v_1v_2 telle que $v_1 \in K_1$ et $v_2 \in K_2$, on crée une solution partielle S en l'initialisant avec les deux sommets v_1 et v_2. Ensuite, pour chaque couleur c différente de $col(v_1)$ et $col(v_2)$, on ajoute exactement un sommet v tel que $col(v) = c$.

On note bien que notre algorithme énumère dans un graphe cobiparti tous les ensembles tropicaux connexes minimaux possibles qui soient du type A ou type B.

Clairement, pour tout ensemble S du type A, S est une clique et donc G[S] est connexe. Concernant maintenant les ensembles du type B, on a pour tout $i \in \{1,2\}$, $K_i \cap S \neq \emptyset$. Donc il devrait y avoir une paire de sommets adjacents v_1, v_2 tels que $v_i \in K_i$ pour tout $i \in \{1,2\}$. Ensuite, par la minimalité de S, on ajoute exactement un sommet v tel que $col(v) = c$, pour toute couleur c différente de $col(v_1)$ et $col(v_2)$. L'algorithme vérifie finalement si S est un ensemble tropical connexe minimal dans G et le retourne si le cas. Cette vérification en temps polynomial permet d'éliminer les solutions partielles correspondant aux faux choix possibles de v_1 et v_2.

Temps d'exécution

Soit pour énumérer les ensembles du type A, soit pour énumérer ceux du type B, on utilise une unique règle de branchement : parmi tous les sommets d'une classe de couleur c, on sélectionne un unique sommet v. Cette règle de branchement est bien connue sous le nom de Moon & Moser et qui a un vecteur de branchement $(\underbrace{t,\ldots,t}_{t})$ où $t = |\{v : col(v) = c\}|$. La valeur maximale de son facteur est atteinte si $t = 3$ et elle vaut $3^{1/3} < 1.4423$ (voir 2.2.3). Afin d'énumérer toutes les solutions du type B, on a appliqué cette dernière règle pour toute paire de sommets adjacents (au plus n^2 paires de sommets). Et cela nous permet de déduire le théorème suivant :

Théorème 16. *Un graphe cobiparti de n sommets a* $\mathcal{O}(n^2 \cdot 3^{n/3})$ *ensembles tropicaux connexes minimaux qu'on peut énumérer en* $\mathcal{O}^*(3^{n/3})$.

Borne inférieure

Pour obtenir une borne inférieure des ensembles tropicaux connexes minimaux dans les graphes cobipartis, on construit un graphe G de la manière suivante :

— Pour tout $i \in \{1,\dots,k\}$, construisez une clique de trois sommets $T_i = \{x_i, y_i, z_i\}$.

— Assignez à tous les sommets de la clique T_i une couleur unique i.

— Construisez un graphe complet G, en ajoutant les arêtes nécessaires, avec l'ensemble de sommets $\cup_{1\leq i\leq K} T_i$.

Clairement G a $n = 3k$ sommets. On note que $S \subseteq V(G)$ est un ensemble tropical connexe minimal de G si et seulement si $|S \cap T_i| = 1$ pour tout $i \in \{1,\dots,k\}$. Par conséquent, G a $3^k = 3^{n/3}$ ensembles tropicaux connexes minimaux. Il nous reste à noter que tout graphe complet est un graphe cobiparti. Cela nous permet de déduire le lemme suivant :

Lemme 23. *Il existe des graphes cobipartis avec au moins $3^{n/3}$ ensembles tropicaux connexes minimaux.*

Dans la section suivante, on a étudié les ensembles tropicaux connexes minimaux dans les graphes blocs.

4.3.5 Les ensembles tropicaux connexes minimaux dans les graphes blocs

Un bloc d'un graphe est soit un sous-graphe biconnexe maximal, soit un pont, soit un sommet isolé [50]. Un graphe est dit bloc si et seulement si chaque bloc est une clique. Deux blocs distincts dans un graphe bloc ont au plus un sommet en commun [38].

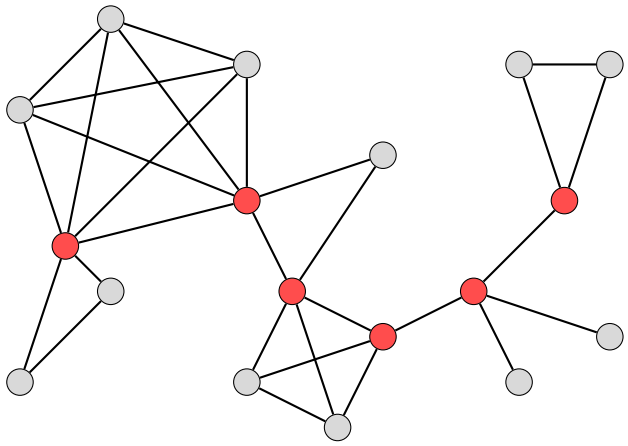

FIGURE 4.8 – Un graphe bloc

Algorithme d'énumération

Deux blocs distincts dans un graphe bloc G ont au plus un sommet en commun qui est un point d'articulation du graphe G (voir la figure 4.8 où les points d'articulations sont en rouge). De plus, il existe entre deux sommets non adjacents de G un chemin induit unique constitué par des sommets qui ne sont que des points d'articulations de G. Notre algorithme, par la minimalité de S, pique exactement un sommet de chaque classe de couleur. Puis, afin de connecter l'ensemble S, on ajoute les points d'articulations nécessaires de telle façon qu'il existe un chemin entre toute paire de sommets de G[S].

Temps d'exécution

On note bien que l'énumération de tous les points d'articulation du graphe G se fait en temps linéaire grâce à l'algorithme de Tarjan [62]. Ensuite, on n'a utilisé qu'une unique règle de branchement du type Moon & Moser et qui a au pire un facteur de $3^{1/3}$. Et cela nous permet de déduire le théorème suivant :

Théorème 17. *Un graphe bloc de n sommets a au plus* $3^{n/3}$ *ensembles tropicaux connexes minimaux qu'on peut énumérer en* $\mathcal{O}^*(3^{n/3})$.

Borne inférieure

On note bien que tout graphe complet est un graphe bloc. Donc on peut considérer que la borne inférieure décrite dans la section précédente 4.3.4 est une borne inférieure aussi pour les graphes blocs et cela nous permet de déduire le lemme suivant :

Lemme 24. *Il existe des graphes blocs avec au moins* $3^{n/3}$ *ensembles tropicaux connexes minimaux.*

4.3.6 Les ensembles tropicaux connexes minimaux dans les graphes d'intervalles

Un graphe d'intervalles G = (V, E) est présenté par un ensemble d'intervalles dans lequel chaque sommet correspond à un intervalle de la droite réelle. Deux sommets sont adjacents dans le graphe G si et seulement si leurs intervalles correspondants ont une intersection non vide.

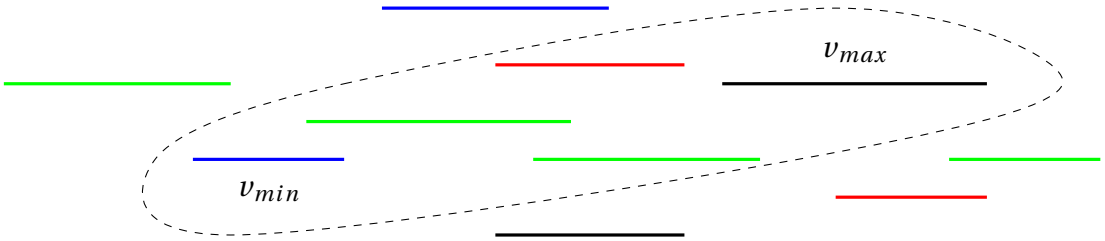

FIGURE 4.9 – Un ensemble tropical connexe minimal dans un graphe d'intervalles

Algorithme d'énumération

Soit S un ensemble tropical connexe minimal de G. Soit $\mathscr{I}$ un modèle d'intervalles du graphe G. Soit $v_{\min} = \arg\min_{v \in S} l(v)$ et $v_{\max} = \arg\max_{v \in S} r(v)$.
Comme G[S] est connexe, alors il existe un chemin P entre $v_{\min}$ et $v_{\max}$ dans G[S] (voir la figure 4.9 où l'ensemble S est marqué avec est une ligne pointillée). De plus, on peut déduire immédiatement le lemme suivant :

Lemme 25. *Chaque sommet de* S *a au moins un adjacent dans* P.

Démonstration. Il est clair que si $x \in P$, alors x a au moins, par la définition d'un chemin, un adjacent dans P.
Alors on suppose que $x \notin P$. Par les définitions de $v_{\min}$ et $v_{\max}$, on a $l(v_{\min}) \leq l(x)$ et

$r(x) \leq r(v_{\max})$. Donc l'intervalle correspondant au sommet x dans le modèle d'intervalles $\mathscr{I}$ s'intersecte avec au moins un autre intervalle correspondant à un sommet du chemin P. Alors, le sommet x est adjacent au moins à un sommet de P.
Par conséquent, chaque sommet de S a au moins un adjacent dans P. □

En exploitant le lemme 25, notre algorithme, pour deux sommets donnés $v_{\min}$ et $v_{\max}$, procède comme suit :

— Étape 1 : On énumère tout chemin induit P entre les deux sommets $v_{\min}$ et $v_{\max}$.
— Étape 2 : Pour chaque chemin induit P, qui représente déjà une solution partielle S, on lui ajoute exactement un sommet de chaque classe de couleur sans sommets dans P.

Temps d'exécution

L'analyse du temps d'exécution de l'algorithme se base sur la technique Mesurer pour Conquérir où on a associé à un graphe d'intervalles $G = (V, E)$ la mesure $\mu = |X| + \omega|T|$ tels que X et T sont des sous-ensembles disjoints de V et ω est un réel entre 0 et 1 qui représente le poids d'un sommet. On note bien qu'initialement l'ensemble $T = \emptyset$ et que l'ensemble $X = V$. Donc $\mu \leq |V|$. Soit $T(\mu)$ le nombre maximum des ensembles tropicaux connexes minimaux dans un graphe d'intervalles d'une mesure au plus μ.

La première étape de l'algorithme consiste à construire un chemin induit P entre les deux sommets $v_{\min}$ and $v_{\max}$ en sélectionnant chaque fois un unique sommet adjacent parmi tous les adjacents du dernier sommet sélectionné et qui a une extrémité droite plus droite que ce dernier sommet, sinon le chemin produit ne serait pas induit. On ajoute ce dernier sommet sélectionné à S et on l'élimine de X. Alors que les autres sommets non sélectionnés seront déplacés de X à T. Si on sélectionne un sommet parmi t sommets candidats, alors on obtient la récurrence suivante :

$$T(\mu) \leq t \cdot T(\mu - (1 + (t-1)(1-\omega)))$$

Dans la deuxième étape de l'algorithme, de même parmi tous les sommets d'une classe de couleur, qui sont dans T, sans sommets dans P, on sélectionne exactement un sommet. Ainsi, on désigne par s la taille de la classe d'une couleur, alors on a la récurrence suivante :

$$T(\mu) \leq s \cdot T(\mu - s \cdot \omega)$$

Pour un $\omega = 0.5895$, les deux récurrences sont bornées par 1.8613^{μ}. Le maximum est atteint pour un $t = 4$ et un $s = 3$ et qui correspond à un facteur de branchement majoré par 1.8613. On rappelle que l'algorithme est appliqué pour toute paire possible de $v_{\min}$ et $v_{\max}$ (au plus n^2 paires de sommets) et que $\mu \leq n$. Et cela nous permet de déduire le théorème suivant :

Théorème 18. *Un graphe d'intervalles de n sommets a $\mathcal{O}(n^2 \cdot 1.8613^n)$ ensembles tropicaux connexes minimaux qu'on peut énumérer en $\mathcal{O}(1.8613^n)$.*

Borne inférieure

On note bien que tout graphe complet est un graphe d'intervalles. Donc on peut considérer que la borne inférieure décrite dans la section précédente 4.3.4 est une borne inférieure aussi pour les graphes d'intervalles et cela nous permet de déduire le lemme suivant :

Lemme 26. *Il existe des graphes d'intervalles avec au moins $3^{n/3}$ ensembles tropicaux connexes minimaux.*

4.4 Conclusion

Dans ce chapitre, on a étudié l'énumération des ensembles tropicaux connexes minimaux et des ensembles dominants connexes minimaux. Les deux ensembles satisfont une contrainte sur la globalité du graphe qui est la connexité. On voit que cette contrainte augmente la difficulté de concevoir des « bons » algorithmes d'énumération input-sensitive d'une part, et de trouver des familles de graphes qui représentent des « bonnes » bornes inférieures d'autre part. Déjà le meilleur algorithme connu pour énumérer les ensembles dominants connexes minimaux s'exécute en $\mathcal{O}(2^{(1-\epsilon)n})$ avec $\epsilon > 10^{-50}$ [54] alors que la borne inférieure qu'on a est seulement $3^{(n-2)/3}$[32]. De même, concernant les ensembles tropicaux connexes minimaux, on n'a pas réussi durant cette thèse à prouver que le nombre maximum de ces ensembles est inférieur à 2^n. En revanche, on a proposé une borne inférieure hautement non triviale de 1.4916^n. On voit qu'à travers les familles de graphes proposées que la découverte de bonnes bornes inférieures des ensembles connexes n'est pas facile vu l'impossibilité de concevoir des familles de graphes en utilisant l'union disjointe. Le problème persiste aussi avec les algorithmes d'énumération output-sensitive. Et au meilleur de nos connaissances, il n'existe pas d'algorithme d'énumération des ensembles dominants connexes minimaux qui touchent des grandes classes de graphes. On note bien que le résultat de ce chapitre sur l'énumération des ensembles dominants connexes minimaux dans les graphes bipartis convexes a ouvert la voie pour GRAS et LIEDLOFF [37] à améliorer la borne supérieure de ces ensembles dans les graphes cordaux bipartis.

Pour conclure ce chapitre, on donne un survol des résultats connus sur l'énumération « input-sensitive » des ensembles dominants connexes minimaux et des ensembles tropicaux connexes minimaux, où n et m sont respectivement le nombre de sommets et le nombre des arêtes du graphe.

Les classes de graphes	**Borne inférieure**	**Borne supérieure**
Les graphes quelconques	$3^{(n-2)/3}$ [32]	$\mathcal{O}(2^{(1-\epsilon)n})$ [54]
Les cordaux	$3^{(n-2)/3}$ [32]	$\mathcal{O}(1.4736^n)$ [33]
Les graphes scindés	1.3195^n [17]	1.3674^n [61]
Les cobipartis	1.3195^n [17]	$n^2+2\times 1.3674^n$ [61]
Les graphes d'intervalles	$3^{(n-2)/3}$ [32]	$3^{(n-2)/3}$ [32]
Les graphes sans astéroïde triples	$3^{(n-2)/3}$ [32]	$\mathcal{O}^*(3^{(n-2)/3})$ [32]
Les graphes fortement cordaux	$3^{(n-2)/3}$ [32]	$3^{n/3}$ [32]
Les distance-héréditaires	$3^{(n-2)/3}$ [32]	$3^{n/3}\times n$ [32]
Les cographes	m [32]	m [32]
Les bipartis convexes	$3^{(n-2)/3}$ [Cette thèse]	$\mathcal{O}(1.7254^n)$ [Cette thèse]
Les bipartis cordaux	$3^{(n-2)/3}$ [Cette thèse]	$\mathcal{O}^*(1.7990^n)$ [37]

TABLEAU 4.1 – Un survol sur l'énumération des ensembles dominants connexes minimaux

Les classes de graphes	**Borne inférieure**	**Borne supérieure**
Les graphes quelconques	1.4961^n [Cette thèse]	2^n
Les graphes cordaux	1.4961^n [Cette thèse]	2^n
Les graphes scindés	1.4766^n [Cette thèse]	1.6042^n [51]
Les graphes cobipartis	$3^{n/3}$ [Cette thèse]	$n^2\cdot 3^{n/3}+n^2$ [Cette thèse]
Les graphes d'intervalles	$3^{n/3}$ [Cette thèse]	1.8613^n [Cette thèse]
Les graphes blocs	$3^{n/3}$ [Cette thèse]	$3^{n/3}$ [Cette thèse]

TABLEAU 4.2 – Un survol sur l'énumération des ensembles tropicaux connexes minimaux

Chapitre 5

Résumé et conclusion

Sommaire

Une solution qui satisfait certains critères est dite réalisable. Le nombre de solutions réalisables est parfois exponentiel, on dit dans ce cas que notre problème est combinatoire. Ce nombre exponentiel est l'origine de la diversité de la nature des problèmes combinatoires. Un problème d'optimisation consiste à trouver la meilleure solution réalisable satisfaisant certains critères. La meilleure solution est appelée la solution optimale. La solution optimale est la solution qui réalise le maximum du profit ou qui minimise au maximum le coût. Un problème d'énumération consiste à énumérer explicitement tous les solutions réalisables possibles. Alors qu'un problème de dénombrement consiste à compter le nombre de solutions possibles satisfaisant certains critères sans les énumérer. L'énumération des tous les ensembles réalisables d'un problème donne immédiatement la solution optimale et le nombre exact de tous les solutions. Donc on peut réduire les problèmes d'optimisation et de dénombrement aux problèmes d'énumération. Ce qui montre bien que les problèmes d'énumération sont indispensables et plus difficiles du point de vue complexité. Récemment l'énumération a suscité un grand intérêt et un établissement de plusieurs nouveaux résultats et articles. Cet intérêt a touché plusieurs domaines de l'informatique comme la bio-informatique, l'intelligence artificielle, l'exploration de données, les bases de données, ...

Cet intérêt croissant pour l'énumération a crée deux communautés ou deux approches : l'une est appelée « output-sensitive » et l'autre « input-sensitive ».

La mesure du temps d'exécution des algorithmes d'énumération de type « Output-sensitive » se base sur la taille d'entrée, mais surtout sur la taille de sortie. C'est-à-dire le temps d'exécution nécessaire pour produire deux solutions consécutives, appelé délai, ainsi que le temps nécessaire pour trouver la première solution.

Alors que l'analyse du temps d'exécution des algorithmes d'énumération « input-sensitive » se base uniquement sur la taille d'entrée.

L'objectif de cette dernière approche n'est pas seulement d'énumérer des objets mais aussi de donner une « borne supérieure » ou un « majorant » pour le nombre maximum de ces objets. Le temps d'exécution d'un algorithme d'énumération « input-sensitive » donne systématiquement cette borne. Et la question qui reste à poser est si on peut concevoir un algorithme plus rapide, autrement dit, si la « borne supérieure » est bien stricte. Pour cela, la trouvaille d'une famille de graphes particulière, appelée « borne inférieure », avec le nombre le plus élevé possible de ces objets énumérés est nécessaire pour s'assurer que l'algorithme est optimal et que la « borne supérieure » est bien stricte.

Parmi les travaux les plus connus dans cette direction, il y a celles de MOON et MOSER [55] qui ont montré que le nombre maximum des ensembles stables maximaux dans un graphe de n sommets est $3^{n/3}$.

Au contraire de l'énumération des ensembles stables maximaux, avoir deux bornes qui se coïncident n'est pas évident du tout. Et c'est assez courant dans l'énumération « input-sensitive » d'avoir un grand écart. Ce problème concerne même les ensembles les plus classiques et les plus étudiés comme les ensembles dominants minimaux où le meilleur algorithme connu pour énumérer ces ensembles est en $\mathcal{O}(1.7159^n)$ et la meilleure borne inférieure connue est seulement 1.5704^n.

Durant cette thèse, on a proposé un nouvel algorithme pour énumérer tous les ensembles dominants minimaux dans les graphes cordaux en $\mathcal{O}(1.5048^n)$. On a étudié aussi l'énumération des ensembles dominants connexes minimaux et les ensembles irredondants maximaux qui sont très proches des ensembles dominants minimaux. On a proposé un algorithme d'énumération des ensembles dominants connexes minimaux dans les graphes bipartis convexes en $\mathcal{O}(1.7254^n)$. D'une part c'est déjà un progrès par rapport à la borne précédente, celle d'un graphe général, qui est en $\mathcal{O}(2^{(1-\epsilon)n})$ où $\epsilon > 10^{-50}$. D'autre

part, c'est le premier algorithme d'énumération « input-sensitive » établi pour une sous-classe non triviale de graphes bipartis qui concernent l'ensemble dominant minimal et ses variants.

On a conçu aussi des algorithmes d'énumération des ensembles irredondants maximaux dans les graphes cordaux, les graphes d'intervalles et les forêts en $\mathcal{O}(1.7549^n)$, $\mathcal{O}(1.6957^n)$ et $\mathcal{O}(1.6181^n)$ respectivement au lieu de l'algorithme trivial en $\mathcal{O}^*(2^n)$. On a proposé aussi comme une borne inférieure une famille de forêts avec 1.5292^n ensembles irredondants maximaux. On a réduit l'écart entre les deux bornes à néant dans les co-graphes en montrant que le nombre maximum de ces ensembles est exactement $15^{n/6}$.

Finalement, afin de varier, on a étudié un nouvel ensemble défini récemment : L'ensemble tropical connexe minimal. On a proposé une borne inférieure de 1.4961^n, mais sans réussir à améliorer la borne supérieure de 2^n. On a proposé des algorithmes d'énumération des ensembles tropicaux connexes minimaux dans les graphes cobipartis, les graphes d'intervalles et les graphes blocs en $\mathcal{O}^*(3^{n/3})$, $\mathcal{O}(1.8613^n)$ et $\mathcal{O}^*(3^{n/3})$ respectivement. On a établi une borne inférieure de 1.4766^n pour les graphes scindés et de $3^{n/3}$ pour les graphes cobipartis, les graphes d'intervalles et les graphes blocs.

Contrairement à l'énumération « input-sensitive », la complexité des problèmes d'énumération « output-sensitive » est bien étudiée et les algorithmes sont classés selon le délai entre deux solutions consécutives. Alors que la classification des problèmes d'énumération « input-sensitive » semble ne pas avoir de sens ou n'existe pas. Pour cela, on a essayé durant cette thèse de varier la nature de la contrainte et les classes de graphes. Tout d'abord, on a commencé par l'étude, dans le chapitre 2, des ensembles dominants minimaux où la contrainte de dominance est une contrainte locale. Puis dans le chapitre 3, on a étudié l'énumération des ensembles irredondants maximaux. Et contrairement à un ensemble dominant minimal, la contrainte qu'un ensemble soit irredondant maximal n'est pas locale. En fait, on peut avoir un sommet qui n'est pas dominé par aucun sommet d'un ensemble irredondant maximal. Et cela n'est pas facile de le fixer immédiatement. Finalement, on a augmenté le défi et on a étudié, dans le chapitre 4, l'énumération des ensembles dominants connexes minimaux et des ensembles tropicaux connexes minimaux où la contrainte de connexité est une contrainte globale. De même, on a essayé d'étudier l'énumération de ces ensembles dans différentes classes de graphes. Alors on a réussi à concevoir des algorithmes d'énumération dans les graphes d'intervalles, les forêts, les graphes bipartis convexes..., mais surtout dans les graphes cordaux où on a établi une nouvelle propriété. On a montré l'existence d'un sommet particulier appelé semi-simplicial qui peut être trouvé en temps polynomial. La notion du sommet semi-simplicial semble très prometteuse pour la résolution de certains autres problèmes encore ouverts dans les graphes cordaux.

Par conséquent, on a constaté que la conception des algorithmes d'énumération « input-sensitive » des ensembles qui ont des contraintes globales est plus délicate que la conception des algorithmes concernant des ensembles avec des contraintes locales. De même, pour ces problèmes étudiés, il semble très difficile d'exploiter les propriétés de certains graphes en particulier les graphes planaires et les graphes bipartis.

On note bien que les techniques utilisées pour résoudre les problèmes d'énumération « input-sensitive » jusqu'à nos jours, soit dans cette thèse soit dans les différents articles publiés, sont principalement « Brancher et réduire » et « Mesurer pour Conquérir ».

Uniquement ces deux techniques, parmi les différentes techniques connues des algorithmes exponentiels, semblent bonnes pour résoudre ce type de problème.

Dans la limite des techniques utilisées, c'est assez courant dans l'énumération « input-sensitive » d'avoir un grand écart entre la borne supérieure et la borne inférieure. Donc

est-il possible qu'on ait pas réussi à découvrir encore la technique adéquate à l'énumération « input-sensitive » ? D'un autre côté positif, ce grand écart de certaines bornes montre qu'il est possible d'améliorer certains algorithmes d'optimisation en énumérant toutes les solutions possibles. Par exemple, le meilleur algorithme connu pour trouver l'ensemble dominant connexe minimum est $\mathcal{O}(1.8619^n)$ [2] alors que la meilleure borne inférieure connue des ensembles dominants connexes minimaux est 1.4422^n [32]. De même, concernant l'ensemble irredondant maximal de cardinalité minimale, le meilleur algorithme d'optimisation connu est en $\mathcal{O}^*(1.99914^n)$ [6] alors que l'algorithme d'énumération est en $\mathcal{O}^*(2^n)$. Donc toute amélioration dans la borne supérieure triviale implique facilement un nouvel algorithme d'optimisation.

Finalement, comme une perspective et pour attirer l'attention de la communauté sur l'énumération des ensembles dominants totaux minimaux, on a donné une borne inférieure de 1.5848^n.

5.1 Résumé

Moon et Moser ont prouvé que le nombre maximum des ensembles stables maximaux dans un graphe de n sommets est au plus $3^{n/3}$. Cette borne, appelée borne supérieure, est stricte vu l'existence d'une famille des graphes avec un tel nombre appelée borne inférieure. Au contraire de l'énumération des ensembles stables maximaux, avoir deux bornes qui se coïncident n'est pas évident du tout. Et c'est assez courant dans l'énumération « input-sensitive » d'avoir un grand écart. Ce problème concerne même les ensembles les plus classiques et les plus étudiés comme les ensembles dominants minimaux où le meilleur algorithme connu pour énumérer ces ensembles est en $\mathcal{O}(1.7159^n)$ et la meilleure borne inférieure connue est seulement 1.5704^n.

Durant cette thèse, on a proposé un nouvel algorithme pour énumérer tous les ensembles dominants minimaux dans les graphes cordaux en $\mathcal{O}(1.5048^n)$. On a étudié aussi l'énumération des ensembles dominants connexes minimaux et les ensembles irredondants maximaux qui sont très proches des ensembles dominants minimaux. On a proposé un algorithme d'énumération des ensembles dominants connexes minimaux dans les graphes bipartis convexes en $\mathcal{O}(1.7254^n)$. D'une part c'est déjà un progrès par rapport à la borne précédente, celle d'un graphe général, qui est en $\mathcal{O}(2^{(1-\epsilon)n})$ où $\epsilon > 10^{-50}$. D'autre part, c'est le premier algorithme d'énumération « input-sensitive » établi pour une sous-classe non triviale de graphes bipartis qui concernent l'ensemble dominant minimal et ses variants.

On a conçu aussi des algorithmes d'énumération des ensembles irredondants maximaux dans les graphes cordaux, les graphes d'intervalles et les forêts en $\mathcal{O}(1.7549^n)$, $\mathcal{O}(1.6957^n)$ et $\mathcal{O}(1.6181^n)$ respectivement au lieu de l'algorithme trivial en $\mathcal{O}^*(2^n)$. On a proposé aussi comme une borne inférieure une famille de forêts avec 1.5292^n ensembles irredondants maximaux.

Afin de varier, on a étudié un nouvel ensemble défini récemment : L'ensemble tropical connexe minimal. On a proposé une borne inférieure de 1.4961^n, mais sans réussir à améliorer la borne supérieure de 2^n. On a proposé des algorithmes d'énumération des ensembles tropicaux connexes minimaux dans les graphes cobipartis, les graphes d'intervalles et les graphes blocs en $\mathcal{O}^*(3^{n/3})$, $\mathcal{O}(1.8613^n)$ et $\mathcal{O}^*(3^{n/3})$ respectivement. On a établi une borne inférieure de 1.4766^n pour les graphes scindés et de $3^{n/3}$ pour les graphes cobipartis, les graphes d'intervalles et les graphes blocs.

Finalement, comme perspective et pour attirer l'attention de la communauté sur l'énumération des ensembles dominants totaux minimaux, on a donné une borne inférieure de 1.5848^n.

5.2 Summary

Moon and Moser proved that the maximum number of maximal independent sets in a graph of n vertices is at most $3^{n/3}$. This maximum number, called upper bound, is tight given the existence of a family of graphs with such a number called lower bound.

Unlike the enumeration of maximal independent sets, having a tight bounds is not obvious at all. And it's quite common in the "input-sensitive" enumeration to have a big gap. This problem concerns even the most studied sets as minimal dominating sets where the best known algorithm to enumerate those sets runs in time $\mathcal{O}(1.7159^n)$ and the best known lower bound is only 1.5704^n.

During this thesis, we proposed a "Measure and Conquer" algorithm to enumerate all minimal dominating sets for chordal graphs in time $\mathcal{O}(1.5048^n)$. The enumeration of minimal connected dominating sets and maximal irredundant sets, which are closely related to the enumeration of minimal dominating sets, were also studied.

An enumeration algorithm of minimal connected dominating sets in convex bipartite graphs has been proposed with a running time in $\mathcal{O}(1.7254^n)$. On the one hand, it is already a progress compared to the previous one, that for general graphs, which is in time $\mathcal{O}(2^{(1-\epsilon)n})$ where $\epsilon > 10^{-50}$. On the other hand, it is the first "input-sensitive" enumeration algorithm for some non trivial subclass of bipartite graphs.

Enumeration algorithms of maximal irredundant sets have also been given for chordal graphs, interval graphs, and forests in times $\mathcal{O}(1.7549^n)$, $\mathcal{O}(1.6957^n)$ and $\mathcal{O}(1.6181^n)$ respectively instead of the trivial algorithm in time $\mathcal{O}^*(2^n)$. We complement those upper bounds with a lower bound by providing a family of forests having 1.5292^n maximal irredundant sets.

Finally, to vary, we studied a new set has been defined recently: The minimal tropical connected set. A lower bound of 1.4961^n has been proposed but we failed to improve the upper bound of 2^n. Enumeration algorithms of minimal tropical connected sets have been proposed for cobipartite, interval and block graphs in times $\mathcal{O}^*(3^{n/3})$, $\mathcal{O}(1.8613^n)$ and $\mathcal{O}^*(3^{n/3})$ respectively. A lower bound of 1.4766^n for splits graphs and $3^{n/3}$ for cobipartite, interval graphs and block graphs have been provided.

We provide a new lower bound of 1.5848^n, as a perspective and in order to draw community attention to the maximum number of minimal total dominating sets.

Chapitre 6

Bibliographie

Bibliographie

[1] ABU-KHZAM, F. N. et P. HEGGERNES. 2016, «Enumerating minimal dominating sets in chordal graphs», *Inf. Process. Lett.*, vol. 116, nº 12, doi :10.1016/j.ipl.2016.07.002, p. 739–743. URL https://doi.org/10.1016/j.ipl.2016.07.002. 5, 6, 24, 27

[2] ABU-KHZAM, F. N., A. E. MOUAWAD et M. LIEDLOFF. 2011, «An exact algorithm for connected red-blue dominating set», *J. Discrete Algorithms*, vol. 9, nº 3, doi :10.1016/j.jda.2011.03.006, p. 252–262. URL https://doi.org/10.1016/j.jda.2011.03.006. 72, 96

[3] BANG-JENSEN, J., J. HUANG, G. MACGILLIVRAY et A. YEO. 1999, «Domination in convex bipartite and convex-round graphs», cahier de recherche. URL https://dl.acm.org/citation.cfm?id=870702. 72

[4] BELLMAN, R. 1962, «Dynamic programming treatment of the travelling salesman problem», *J. ACM*, vol. 9, nº 1, doi :10.1145/321105.321111, p. 61–63, ISSN 0004-5411. URL http://doi.acm.org/10.1145/321105.321111. 74

[5] BERGOUGNOUX, B., F. CAPELLI et M. M. KANTÉ. 2019, «Counting minimal transversals of β-acyclic hypergraphs», *J. Comput. Syst. Sci.*, vol. 101, doi :10.1016/j.jcss.2018.10.002, p. 21–30. URL https://doi.org/10.1016/j.jcss.2018.10.002. 41

[6] BINKELE-RAIBLE, D., L. BRANKOVIC, M. CYGAN, H. FERNAU, J. KNEIS, D. KRATSCH, A. LANGER, M. LIEDLOFF, M. PILIPCZUK, P. ROSSMANITH et J. O. WOJTASZCZYK. 2011, «Breaking the 2^n-barrier for irredundance : Two lines of attack», *J. Discrete Algorithms*, vol. 9, nº 3, doi :10.1016/j.jda.2011.03.002, p. 214–230. URL https://doi.org/10.1016/j.jda.2011.03.002. 44, 46, 96

[7] BONAMY, M., O. DEFRAIN, M. HEINRICH et J. RAYMOND. 2019, «Enumerating minimal dominating sets in triangle-free graphs», dans *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, p. 16 :1–16 :12, doi :10.4230/LIPIcs.STACS.2019.16. URL https://doi.org/10.4230/LIPIcs.STACS.2019.16. 24

[8] BOOTH, K. S. et J. H. JOHNSON. 1982, «Dominating sets in chordal graphs», *SIAM J. Comput.*, vol. 11, nº 1, doi :10.1137/0211015, p. 191–199. URL https://doi.org/10.1137/0211015. 26

[9] BOROS, E. «Generating maximal irredundant and minimal redundant subfamilies of hypergraphs», *WEPA 2016 : First Workshop on Enumeration Problems and Applications 21-22 Nov 2016 Aubiere (France)*. 46

[10] BRANDSTÄDT, A., V. B. LE et J. P. SPINRAD. 1999, *Graph Classes : A Survey*, vol. 3, Society for Industrial and Applied Mathematics, ISBN 0-89871-432-X, doi : 10.1137/1.9780898719796. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898719796. 74

[11] CASPARD, N., B. LECLERC, B. MONJARDET et collab.. 2007, *Ensembles ordonnés finis : concepts, résultats et usages*, vol. 60, Springer, ISBN 978-3-540-73755-1, doi :10.1007/978-3-540-73756-8. URL https://doi.org/10.1007/978-3-540-73756-8. 11

[12] CHANG, M.-S., P. NAGAVAMSI et C. RANGAN. 1998, «Weighted irredundance of interval graphs», *Information Processing Letters*, vol. 66, nº 2, doi :10.1016/s0020-0190(98)00040-4, p. 65–70. URL https://doi.org/10.1016/s0020-0190(98)00040-4. 44

[13] CHAPELLE, M., M. COCHEFERT, D. KRATSCH, R. LETOURNEUR et M. LIEDLOFF. 2017, «Exact exponential algorithms to find tropical connected sets of minimum size», *Theor. Comput. Sci.*, vol. 676, doi :10.1016/j.tcs.2017.03.003, p. 33–41. URL https://doi.org/10.1016/j.tcs.2017.03.003. 73

[14] CHVÁTAL, V. et P. L. HAMMER. 1973, «Set-packing and threshold graphs», *Res. Rep., Comput. Sci. Dept., Univ. Waterloo, 1973*. 7

[15] COOK, S. A. 1971, «The complexity of theorem-proving procedures», dans *Proceedings of the third annual ACM symposium on Theory of computing*, p. 151–158. 11

[16] COUTURIER, J., P. HEGGERNES, P. VAN 'T HOF et D. KRATSCH. 2012, «Minimal dominating sets in graph classes : Combinatorial bounds and enumeration», dans *SOFSEM 2012 : Theory and Practice of Computer Science - 38th Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 21-27, 2012. Proceedings*, p. 202–213, doi :10.1007/978-3-642-27660-6_17. URL https://doi.org/10.1007/978-3-642-27660-6_17. 42

[17] COUTURIER, J., P. HEGGERNES, P. VAN 'T HOF et D. KRATSCH. 2013, «Minimal dominating sets in graph classes : Combinatorial bounds and enumeration», *Theor. Comput. Sci.*, vol. 487, doi :10.1016/j.tcs.2013.03.026, p. 82–94. URL https://doi.org/10.1016/j.tcs.2013.03.026. 24, 26, 27, 42, 46, 69, 73, 91

[18] COUTURIER, J., R. LETOURNEUR et M. LIEDLOFF. 2015, «On the number of minimal dominating sets on some graph classes», *Theor. Comput. Sci.*, vol. 562, doi :10.1016/j.tcs.2014.11.006, p. 634–642. URL https://doi.org/10.1016/j.tcs.2014.11.006. 24, 42, 44, 69

[19] DAMASCHKE, P., H. MÜLLER et D. KRATSCH. 1990, «Domination in convex and chordal bipartite graphs», *Inf. Process. Lett.*, vol. 36, nº 5, doi :10.1016/0020-0190(90)90147-P, p. 231–236. URL https://doi.org/10.1016/0020-0190(90)90147-P. 72

[20] D'AURIAC, J. A., N. COHEN, H. E. MAFTHOUI, A. HARUTYUNYAN, S. LEGAY et Y. MANOUSSAKIS. 2016, «Connected tropical subgraphs in vertex-colored graphs», *Discrete Mathematics & Theoretical Computer Science*, vol. 17, nº 3. URL http://dmtcs.episciences.org/2151. 3, 73

[21] DIRAC, G. A. 1961, «On rigid circuit graphs», dans *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, vol. 25, Springer, ISSN 1865-8784, p. 71–76, doi :10.1007/BF02992776. URL https://doi.org/10.1007/BF02992776. 5, 7, 26, 58

[22] FELLOWS, M., G. FRICKE, S. HEDETNIEMI et D. JACOBS. 1994, «The private neighbor cube», *SIAM Journal on Discrete Mathematics*, vol. 7, nº 1, doi :10.1137/s0895480191199026, p. 41–47. URL https://doi.org/10.1137/s0895480191199026. 44

[23] FERNAU, H., P. A. GOLOVACH et M.-F. SAGOT. 2018, «Algorithmic enumeration : Output-sensitive , input-sensitive , parameterized , approximative», . 69

[24] FOMIN, F. V., F. GRANDONI, A. V. PYATKIN et A. A. STEPANOV. 2008, «Combinatorial bounds via measure and conquer : Bounding minimal dominating sets and applications», *ACM Trans. Algorithms*, vol. 5, nº 1, doi :10.1145/1435375.1435384, p. 9 :1–9 :17. URL https://doi.org/10.1145/1435375.1435384. 21, 24, 25, 26, 41, 42, 45, 47, 69, 74

[25] FOMIN, F. V. et D. KRATSCH. 2010, *Exact Exponential Algorithms*, Texts in Theoretical Computer Science. An EATCS Series, Springer, ISBN 978-3-642-16532-0, doi :10.1007/978-3-642-16533-7. URL https://doi.org/10.1007/978-3-642-16533-7. 15, 39

[26] FREDMAN, M. L. et L. KHACHIYAN. 1996, «On the complexity of dualization of monotone disjunctive normal forms», *Journal of Algorithms*, vol. 21, nº 3, doi :10.1006/jagm.1996.0062, p. 618–628. URL https://doi.org/10.1006/jagm.1996.0062. 46

[27] GAREY, M. R. et D. S. JOHNSON. 1979, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, W. H. Freeman, ISBN 0-7167-1044-7. 72

[28] GAVRIL, F. 1972, «Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph», *SIAM Journal on Computing*, vol. 1, nº 2, doi :10.1137/0201013, p. 180–187. URL https://doi.org/10.1137/0201013. 44

[29] GEORGII, E., S. DIETMANN, T. UNO, P. PAGEL et K. TSUDA. 2009, «Enumeration of condition-dependent dense modules in protein interaction networks», *Bioinformatics*, vol. 25, nº 7, doi :10.1093/bioinformatics/btp080, p. 933–940. URL http://dx.doi.org/10.1093/bioinformatics/btp080. 19

[30] GOLOVACH, P. A., P. HEGGERNES, M. M. KANTÉ, D. KRATSCH et Y. VILLANGER. 2016, «Enumerating minimal dominating sets in chordal bipartite graphs», *Discrete Applied Mathematics*, vol. 199, doi :10.1016/j.dam.2014.12.010, p. 30–36. URL https://doi.org/10.1016/j.dam.2014.12.010. 24

[31] GOLOVACH, P. A., P. HEGGERNES, M. M. KANTÉ, D. KRATSCH et Y. VILLANGER. 2017, «Minimal dominating sets in interval graphs and trees», *Discrete Applied Mathematics*, vol. 216, doi :10.1016/j.dam.2016.01.038, p. 162–170. URL https://doi.org/10.1016/j.dam.2016.01.038. 24

[32] GOLOVACH, P. A., P. HEGGERNES et D. KRATSCH. 2016, «Enumerating minimal connected dominating sets in graphs of bounded chordality», *Theor. Comput. Sci.*, vol. 630, doi :10.1016/j.tcs.2016.03.026, p. 63–75. URL https://doi.org/10.1016/j.tcs.2016.03.026. 69, 72, 73, 81, 90, 91, 96

[33] GOLOVACH, P. A., P. HEGGERNES, D. KRATSCH et R. SAEI. , «Enumeration of minimal connected dominating sets for chordal graphs», . 69, 73, 91

[34] GOLOVACH, P. A., D. KRATSCH, M. LIEDLOFF, M. RAO et M. Y. SAYADI. , «On maximal irredundant sets and (σ,ϱ)-dominating sets in paths», *WEPA 2016 : First Workshop on Enumeration Problems and Applications 21-22 Nov 2016 Aubiere (France)*. 44, 68, 69

[35] GOLOVACH, P. A., D. KRATSCH et M. Y. SAYADI. 2019, «Enumeration of maximal irredundant sets for claw-free graphs», *Theoretical Computer Science*, vol. 754, doi :https://doi.org/10.1016/j.tcs.2018.02.014, p. 3 – 15, ISSN 0304-3975. URL http://www.sciencedirect.com/science/article/pii/S0304397518300999, algorithms and Complexity. 44, 45, 69, 74

[36] GOLUMBIC, M. C. 2004, *Algorithmic graph theory and perfect graphs*, vol. 57, Elsevier, ISBN 978-0-444-51530-8. URL https://www.sciencedirect.com/bookseries/annals-of-discrete-mathematics/vol/57. 5

[37] GRAS, B. et M. LIEDLOFF. 2019, «Enumeration of minimal connected dominating sets in chordal bipartite graphs», dans *CTW 2019 : Proceedings of the 17th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, p. 53–56. URL https://www.ctw.ewi.utwente.nl/CTW2019ProceedingsFinal.pdf. 90, 91

[38] HARARY, F. 1963, «A characterization of block-graphs», *Canadian Mathematical Bulletin*, vol. 6, nº 0, doi :10.4153/cmb-1963-001-x, p. 1–6. URL https://doi.org/10.4153/cmb-1963-001-x. 7, 87

[39] HAYNES, T. W., S. T. HEDETNIEMI et P. J. SLATER. 1997, *Domination in graphs : Advanced topics*, Marcel Dekker, Inc., New York, ISBN 0824700341 (acid-free paper). 24

[40] HAYNES, T. W., S. T. HEDETNIEMI et P. J. SLATER. 1998, *Fundamentals of domination in graphs*, New York : Marcel Dekker, ISBN 9780429157769 (electronic bk.). 24

[41] HEDETNIEMI, S., R. LASKAR et J. PFAFF. 1985, «Irredundance in graphs : a survey», *Congr. Numer*, vol. 48, p. 183–193. 44

[42] HOWORKA, E. 1977, «A CHARACTERIZATION OF DISTANCE-HEREDITARY GRAPHS*», *The Quarterly Journal of Mathematics*, vol. 28, nº 4, doi :10.1093/qmath/28.4.417, p. 417–420, ISSN 0033-5606. URL https://doi.org/10.1093/qmath/28.4.417. 8

[43] IWATA, Y. 2012, «A faster algorithm for dominating set analyzed by the potential method», dans *Parameterized and Exact Computation*, Springer Berlin Heidelberg, p. 41–54, doi :10.1007/978-3-642-28050-4_4. URL https://doi.org/10.1007/978-3-642-28050-4_4. 24, 46

[44] JACOBSON, M. S. et K. PETERS. 1990, «Chordal graphs and upper irredundance, upper domination and independence», *Discrete Mathematics*, vol. 86, nº 1-3, p. 59–69. 44

[45] JING-HO, Y., C. JER-JEONG et G. J. CHANG. 1996, «Quasi-threshold graphs», *Discrete Applied Mathematics*, vol. 69, nº 3, doi :https://doi.org/10.1016/0166-218X(96)00094-7, p. 247 – 255, ISSN 0166-218X. URL http://www.sciencedirect.com/science/article/pii/0166218X96000947. 7

[46] KANTÉ, M. M., V. LIMOUZY, A. MARY et L. NOURINE. 2014, «On the enumeration of minimal dominating sets and related notions», *SIAM Journal on Discrete Mathematics*, vol. 28, nº 4, doi :10.1137/120862612, p. 1916–1929. URL https://doi.org/10.1137/120862612. 24, 44, 69, 72

[47] KANTÉ, M. M., V. LIMOUZY, A. MARY, L. NOURINE et T. UNO. 2013, «On the enumeration and counting of minimal dominating sets in interval and permutation graphs», dans *Algorithms and Computation*, Springer Berlin Heidelberg, p. 339–349, doi :10.1007/978-3-642-45030-3_32. URL https://doi.org/10.1007/978-3-642-45030-3_32. 24

[48] KANTÉ, M. M., V. LIMOUZY, A. MARY, L. NOURINE et T. UNO. 2015, «A polynomial delay algorithm for enumerating minimal dominating sets in chordal graphs», dans *Graph-Theoretic Concepts in Computer Science - 41st International Workshop, WG 2015, Garching, Germany, June 17-19, 2015, Revised Papers*, p. 138–153, doi :10.1007/978-3-662-53174-7_11. URL https://doi.org/10.1007/978-3-662-53174-7_11. 24

[49] KARP, R. M. 1972, «Reducibility among combinatorial problems», dans *Complexity of computer computations*, Springer, p. 85–103. 11

[50] KORTE, B. et J. VYGEN. 2018, *Combinatorial Optimization*, Springer Berlin Heidelberg, doi :10.1007/978-3-662-56039-6. URL https://doi.org/10.1007/978-3-662-56039-6. 7, 87

[51] KRATSCH, D., M. LIEDLOFF et M. Y. SAYADI. 2017, «Enumerating minimal tropical connected sets», dans *SOFSEM 2017 : Theory and Practice of Computer Science - 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*, p. 217–228, doi :10.1007/978-3-319-51963-0_17. URL https://doi.org/10.1007/978-3-319-51963-0_17. 73, 84, 85, 91

[52] KRZYWKOWSKI, M. 2013, «Trees having many minimal dominating sets», *Inf. Process. Lett.*, vol. 113, nº 8, doi :10.1016/j.ipl.2013.01.020, p. 276–279. URL https://doi.org/10.1016/j.ipl.2013.01.020. 24

[53] LASKAR, R. et J. PFAFF. 1983, «Domination and irredundance in split graphs. technicial report 430, clemson university, dept. math», *Sciences*. 44

[54] LOKSHTANOV, D., M. PILIPCZUK et S. SAURABH. 2018, «Below all subsets for minimal connected dominating set», *SIAM Journal on Discrete Mathematics*, vol. 32, nº 3, doi : 10.1137/17m1138753, p. 2332–2345. URL https://doi.org/10.1137/17m1138753. 21, 72, 73, 90, 91

[55] MOON, J. W. et L. MOSER. 1965, «On cliques in graphs», *Israel J. Math.*, vol. 3, p. 23–28, ISSN 0021-2172. 10, 20, 94

[56] MÜLLER, H. et A. BRANDSTÄDT. 1987, «The np-completeness of steiner tree and dominating set for chordal bipartite graphs», *Theoretical Computer Science*, vol. 53, nº 2-3, p. 257–265. 72

[57] PFAFF, J., R. LASKAR et S. HEDETNIEMI. 1983, «Np-completeness of total and connected domination and irredundance for bipartite graphs», dans *Tech. Rept. 428*, Clemson University Clemson, SC. 72

[58] PUECH, J. 1998, «Irredundance perfect and p6-free graphs», *Journal of Graph Theory*, vol. 29, nº 4, p. 239–255. 47

[59] ROTE, G. 2019, «The maximum number of minimal dominating sets in a tree», dans *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, p. 1201–1214, doi :10.1137/1.9781611975482.73. URL https://doi.org/10.1137/1.9781611975482.73. 24, 42, 69

[60] SCHMIDT, J. 2009, *Enumeration : Algorithms and complexity*, mémoire de maîtrise, Leibniz Universität Hannover. 19

[61] SKJØRTEN, I. B. 2017, *Faster enumeration of minimal connected dominating sets in split graphs*, mémoire de maîtrise, The University of Bergen. 42, 73, 91

[62] TARJAN, R. 1972, «Depth-first search and linear graph algorithms», *SIAM journal on computing*, vol. 1, nº 2, p. 146–160. 88