THÈSE POUR OBTENIR LE GRADE DE DOCTEUR
DE L'UNIVERSITÉ DE MONTPELLIER

En Informatique

École doctorale : Information, Structures, Systèmes

Unité de recherche : LIRMM, UMR5506

Parameterized Complexity in Geometric Intersection Graphs

Présentée par Gaétan BERTHE
le 3 Juillet 2025

Devant le jury composé de

Marin BOUGERET Maître de conférences, HdR, LIRMM – Université de Montpellier	**Directeur de thèse**
Daniel GONÇALVES Chargé de recherche, HdR, LIRMM – CNRS	**Directeur de thèse**
Jean-Florent RAYMOND Chargé de recherche, LIP – CNRS	**Encadrant de thèse**
Christophe PAUL Directeur de recherche, HdR, LIRMM – CNRS	**Président du jury**
Paweł RZĄŻEWSKI Associate professor, École polytechnique de Varsovie	**Rapporteur**
Meirav ZEHAVI Full professor, Université Ben-Gourion du Néguev	**Rapporteur**
Nicolas BOUSQUET Chargé de recherche, HdR, LIRIS – CNRS	**Examinateur**
Jérémie CHALOPIN Directeur de recherche, HdR, LIS – CNRS	**Examinateur**
Vincent LIMOUZY Professeur, HdR, LIMOS – Université Clermont Auvergne	**Examinateur**

UNIVERSITÉ DE MONTPELLIER

Remerciements

Je voudrais commencer par remercier mes encadrants de thèse : Marin, Daniel et Jean-Florent, merci d'avoir été à mes côtés pendant ces trois années, vos conseils, votre pédagogie et votre soutien ont fait de ma thèse une expérience des plus épanouissantes !

I am grateful to my jury: Paweł Rzążewski and Meirav Zehavi for agreeing to review my manuscript, Nicolas Bousquet for following the progress of my thesis, providing valuable feedback on my work. Christophe Paul for accepting to be the president of the jury, and last but not least Jérémie Chalopin and Vincent Limouzy, for making the trip to attend my defense.

Je souhaite ensuite remercier les membres du LIRMM, en commençant par mes camarades thésards : Giannos et Vagelis (la Greek Mafia pour les intimes), pour m'avoir guidé dans la découverte de la vie de doctorant et de Montpelliérains. Yann pour avoir été un superbe co-bureau tout au long de cette thèse, et tes concerts inoubliables dans le parc du Peyrou. Laure pour ces week-ends à la découverte de Montpellier et ses environs, ainsi que toutes ces pauses à partager nos trouvailles sur Internet. Amadeus, pour m'avoir accompagné tout au long de cette aventure montpelliéraine, même dans la galère de trouver un logement avec un toit étanche à notre arrivée en thèse. À Hugo pour être toujours disponible, que ce soit pour apporter de l'aide, ou encore pour discuter autour d'une bière. À Simon pour me faire découvrir les derniers MOOC à la mode.

Je remercie aussi les différents stagiaires passés par l'équipe pendant ma thèse, en particulier Ariane et Thomas, pour leur aide précieuse au moment de ma soutenance.

Merci aux autres membres de l'équipe AlGCo (+ Matthieu), ainsi qu'à l'équipe administrative. Particulièrement, merci à Nicolas Serrurier-Gourvès pour tant de choses, tu nous manques terriblement.

Merci à la communauté graphe et algorithmique, travailler dans un cadre si dynamique et rempli de si belles personnes est une chance incroyable !

Une thèse ne se déroule pas uniquement au laboratoire, je remercie mes nombreux colocs qui se sont succédé au cours de cette thèse : Adrien, Maryse, Flore, Capucine, Anaëlle, Yeleen, Solenne et Mathis, pour tous ces moments de partage passés ensemble.

Je souhaite ensuite remercier mes amis rencontrés à l'ENS de Lyon. Si nos thèses et projets respectifs nous ont dispersés à travers la France et même l'Europe, cela ne nous a finalement pas tant éloignés que cela !

En particulier Orégane, Antoine, Baptiste et Alexandre, merci pour ces bons moments passés lors de mes retours à Lyon.

Guillaume pour toutes nos conversations sur mille et un sujets, que ce soit le cinéma, les séries, la littérature (plus rarement), les maths ou les galères quotidiennes. À Zoé et Joffrey, pour toutes nos folles aventures, être toujours perdu n'est pas si mal si c'est pour faire ce genre de rencontres !

À Thibault, Amine, Justine et Jérôme pour notre escapade en Laponie, ainsi qu'Amel, Corentin, Gabriel, Jules, Malo, Mista et Romain pour compléter cette belle équipe qu'est Houston.

La prépa commence à remonter, mais je continue d'avoir la chance de croiser des amis que je m'y suis faits. En particulier merci à Ruben et Koltok pour ces soirées passées à profiter de nos retrouvailles à Montpellier, et à Emma d'être une amie si précieuse, qui m'a appris et apporté tant ! Je n'oublie pas mes autres amis, et j'attends que nos chemins se recroisent avec hâte !

Merci enfin à ma famille : à mes parents, votre soutien inconditionnel et vos encouragements me permettent de poursuivre mes projets les plus fous !

À ma petite sœur Louise, pour me faire voyager avec toutes tes aventures à travers le monde.

À ma grande sœur Marie ainsi qu'à Clément, pour me montrer les belles choses que l'on peut construire avec de la motivation et en se sortant les mains des poches.

À mes neveux : Louis, pour me montrer que faire une thèse n'est pas la seule manière de grandir énormément en trois ans ; et Jules, pour m'avoir motivé sur la dernière ligne droite : Avec une avance d'une dizaine de jours, tu remportes la course entre ta venue au monde et la finalisation de ce manuscrit, bravo !

À mes grands-parents, pour m'avoir accompagné tout au long de mon enfance, et continuer encore aujourd'hui à me suivre dans ma progression.

À tous mes oncles et tantes, pour m'avoir accueilli à travers la France, ces moments passés à vos côtés sont des moments qui me sont chers.

Et pour finir, à mes cousins et cousines, j'ai énormément de chance de grandir à vos côtés !

Abstract

This thesis falls within the framework of the parameterized complexity theory, which focuses on expressing the resolution time for a problem not only in terms of the size of the instance, but in terms of other parameters too. Considerable attention has been given to reduce the parameter contribution, and to identify the situations where this contribution can be significantly lesser than the general case.

Here we consider vertex deletion problems, where the goal is to find, given a graph, a set of vertices of minimum size such that removing them from the graph gives some wanted properties. Here the considered parameter for this problem is the minimum size of such a solution set.

Two typical such examples are FEEDBACK VERTEX SET (resp. K_r-HITTING) which consists in deleting the minimum number of vertices in order to obtain a graph without cycle (resp. without size r clique). For many problems of such type, a resolution in subexponential time is possible for certain graph classes such as the planar graphs, which then differ from the general case where such running time is not possible (under classical complexity assumptions). This phenomenon is known to appear in classes defined as intersection graphs of geometric objects as well, such as the intersection graph of unit disks.

In this thesis, sufficient conditions are given for a geometric intersection graph class to admit subexponential algorithms for classic problems.

We also give negative results, which rule out the possibility of such algorithms for some specific problems and graph classes, providing insights into the limits of tractability.

Titre : Complexité paramétrée dans les graphes d'intersection géométrique

Abstract en français

Cette thèse s'inscrit dans le cadre de la théorie de la complexité paramétrée, qui vise à exprimer le temps de résolution d'un problème non seulement en fonction de la taille de l'instance, mais aussi en fonction d'autres paramètres. Une attention a été portée tout particulièrement à la réduction de la contribution du paramètre, ainsi qu'à l'identification des situations où cette contribution peut être significativement inférieure que dans le cas général. Nous nous intéressons ici en particulier aux problèmes de suppression de sommets, où l'objectif est de trouver, pour un graphe donné, un ensemble de sommets de taille minimale dont la suppression confère au graphe certaines propriétés souhaitées. Le paramètre que l'on considère pour ce problème est alors la taille d'un tel ensemble.

Deux exemples typiques de tels problèmes sont FEEDBACK VERTEX SET (resp. K_r-HITTING) qui consistent à supprimer des sommets pour obtenir un graphe sans cycle (resp. sans clique de taille r). Pour de nombreux problèmes de ce type, une résolution en temps sous-exponentiel est possible dans certaines classes, comme les graphes planaires, qui diffèrent donc du cas général où un tel temps d'exécution n'est pas possible (sous des hypothèses de complexité classique). Ce phénomène est également observé dans des classes de graphes d'intersection d'objets géométriques, tels que les graphes d'intersection de disques unitaires dans le plan.

Dans cette thèse, des conditions suffisantes sont établies pour qu'une classe de graphes d'intersection géométrique admette des algorithmes sous-exponentiels.

Nous fournissons également des résultats négatifs qui excluent l'existence de tels algorithmes pour certains problèmes et certaines classes de graphes, permettant d'affiner la frontière entre les instances simples et les instances complexes.

Contents

Résumé en français

Table des matières

Thèmes de la thèse

Cette thèse s'articule autour de trois thèmes principaux :

Théorie de la complexité. La théorie de la complexité vise à décrire l'efficacité des algorithmes pour résoudre des problèmes calculatoires. Le premier objectif est alors de concevoir de tels algorithmes, en essayant de trouver les meilleures stratégies pour résoudre chaque problème. Un objectif complémentaire est ensuite de justifier que l'on ne peut espérer faire mieux que le meilleur algorithme connu, et vise donc à déterminer ce qui rend un problème difficile à résoudre.

Graphes. Nous rencontrons des problèmes algorithmiques partout autour de nous, par exemple lorsque l'on planifie un événement, que l'on cherche un chemin le plus rapide possible, où que l'on résout un sudoku, etc. Lorsque l'on conçoit un algorithme pour résoudre l'un d'entre eux, il est préférable d'utiliser un langage universel, permetant d'adapter facilement les stratégies d'un problème à l'autre. La théorie des graphes fournit un tel langage. Un *graphe* est l'abstraction d'un réseau d'entités (appelées *sommets*) où certaines paires sont connectées (on dit qu'elles sont connectées par une *arête*). Par exemple, si l'on souhaite représenter les stations de métro d'une ville par un graphe, on peut utiliser les stations comme sommets et placer une arête entre chaque paire de stations consécutives le long d'une ligne de métro. Pour abstraire l'interaction dans un réseau social, les utilisateurs peuvent être représentés par les sommets d'un graphe, et leurs relations peuvent être représentées par des arêtes.

Selon le contexte, il est parfois intéressant de limiter notre attention aux graphes ayant des propriétés spécifiques. Ces propriétés peuvent être de nature combinatoire (par exemple, les graphes où chaque sommet est connecté à un nombre limité d'autres sommets, définissant les graphes de degré borné), topologique (par exemple, en demandant que l'on puisse dessiner le graphe sur une feuille de papier sans qu'aucune paire d'arêtes ne se croise, définissant les graphes planaires), algébrique, etc.

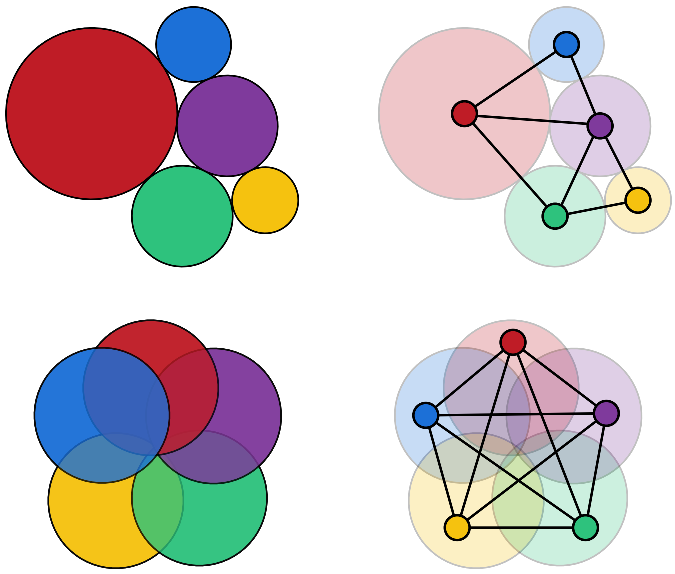

FIGURE 1 : **Dessus :** Graphe d'intersection de disques en contact. Observez que les arêtes ne se croisent pas. **Dessous :** Représentation d'un graphe de disques.

Intersection de formes géométriques. Certaines propriétés sont difficiles à exprimer, et il est parfois plus facile de décrire une construction permettant d'obtenir une classe de graphes partageant certaines propriétés. Un exemple célèbre consiste à considérer des disques en contact dans le plan (les intérieurs sont disjoints), puis de créer le graphe avec un sommet associé à chaque disque et avoir une arête entre les sommets des disques dont les bords se croisent. Il est facile de voir que l'on obtient un graphe planaire (voir figure 1), mais la réciproque est également vraie. Autrement dit, chaque graphe planaire peut être représenté comme le graphe correspondant à un ensemble bien choisi de disques en contact [Koe36]. Notez que nous pourrions également autoriser l'intersection de l'intérieur des disques, ce qui donnerait ce que l'on appelle les *graphes de disques* (voir figure 1).

Motivation

La théorie de la complexité vise à découvrir la frontière qui délimite les problèmes qui peuvent être résolus facilement (typiquement, en temps polynomial) et à identifier les raisons pour lesquelles certains problèmes se situent en dehors de ce territoire.

Toutefois, lors de la résolution d'un problème spécifique dans une certaine situation, les cas difficiles à résoudre peuvent ne pas se présenter. L'un des objectifs consiste alors à trouver la caractérisation la plus complète possible des instances pouvant être résolues en un temps donné. Cela peut se faire en considérant, par exemple, en plus de la taille du problème d'autres paramètres décrivant l'instance, qui peuvent aider à limiter le temps d'exécution d'un algorithme. C'est l'objectif de la théorie de la complexité paramétrée. Une autre solution consiste à restreindre les instances où un sous-ensemble d'instances, décrites par certaines propriétés combinatoires, qui peuvent être utilisées pour la description ou l'analyse de l'algorithme.

Un *problème de suppression de sommets* est un problème algorithmique dans lequel étant donné un graph avec n sommets, il faut décider si la suppression d'au plus k sommets donne au graphe certaines propriétés, telles que l'absence d'arêtes ou la planarité. En remarquant qu'il y a environ n^k sous-ensembles possibles de sommets à considérer, on obtient que pour une propriété facilement vérifiable, nous pouvons résoudre le problème en temps $n^{\mathcal{O}(k)}$. On peut se demander si une meilleure stratégie existe pour résoudre ce type de problème, réduisant le temps de résolution à $n^{\mathcal{O}(\sqrt{k})}$, par exemple.

Question 1. *Quels problèmes de suppression de sommets admettent un algorithme dont le temps d'exécution est de $n^{\mathcal{O}(k^{1-\varepsilon})}$ pour un certain $\varepsilon > 0$?*

On pourrait même espérer un résultat plus fort. Dans ce contexte, un algorithme *soluble à paramètre fixé (FPT)* a un temps d'exécution $f(k)n^{\mathcal{O}(1)}$ pour une fonction calculable f. Un tel temps d'exécution indique alors que, si l'on est capable de limiter le nombre k de sommets que l'on est autorisé à supprimer, le problème de suppression de sommets devient polynomial. Vient alors une deuxième question naturelle.

Question 2. *Quels problèmes de suppression de sommets admettent un algorithme FPT ?*

De nombreux problèmes de ce type sont connus, avec des temps d'exécution de la forme $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$. Pour ces problèmes de suppression de sommets, commence maintenant un objectif d'optimalité : quels sont les meilleurs fonction f

et exposant c tels que l'on peut résoudre le problème en temps $f(k) \cdot \mathcal{O}(n^c)$? Remarquons qu'il n'est pas toujours possible d'optimiser les deux simultanément. Dans cette thèse, nous nous intéresserons à l'amélioration de la fonction en k. En particulier, nous nous intéresserons aux *algorithmes FPT sous-exponentiels*, qui sont des algorithmes dont le temps d'exécution est $2^{\mathcal{O}(k^{1-\varepsilon})} n^{\mathcal{O}(1)}$ pour un certain $\varepsilon > 0$.

Il est théorisé que de nombreux problèmes n'admettent pas de tels algorithmes, cependant, il est possible de ne considérer qu'un certain ensemble de graphes satisfaisant certaines propriétés, appelé une *classe de graphes*. Une nouvelle question se pose alors :

Question 3. *Étant donné un problème de suppression de sommets, quelles sont les classes de graphes les plus générales admettant un algorithme FPT sous-exponentiel ?*

Ou au contraire, tracer la frontière depuis l'autre côté :

Question 4. *Étant donné un problème de suppression de sommets, quelles sont les classes de graphes les plus* ***restreintes*** *qui n'admette pas d'algorithme FPT sous-exponentiel ?*

Observez que pour les limites inférieures, plus une classe de graphes est restreinte, meilleur est le résultat.

Enfin, dans les classes de graphes où de tels algorithmes FPT sous-exponentiels existent, la question suivante est de trouver un algorithme avec le meilleur temps d'exécution possible.

Question 5. *Etant donné un problème de suppression de sommets* Π *et une classe de graphes* $\mathcal{G}$ *telle que* Π *peut être résolu en temps FPT sous-exponentiel pour les graphes de* $\mathcal{G}$*, quel est le plus petit* $\eta > 0$ *tel que* Π *soit résoluble en temps* $2^{\mathcal{O}(k^{\eta})} n^{\mathcal{O}(1)}$ *pour les graphes à* n *sommets de* $\mathcal{G}$ *?*

Cette thèse s'intéressera particulièrement à ces 3 dernières questions.

Plan de la thèse

Les chapitres sont pour la plupart indépendants, à l'exception du chapitre 2, où sont introduits les définitions et les résultats généraux apparaissant tout au long de ce manuscrit, du chapitre 3 qui contient une présentation des techniques de branchement utilisées comme prétraitement pour nos algorithmes, et du chapitre 4 qui introduit des résultats sur la complexité de voisinage utilisés dans les chapitres suivants.

Dans le chapitre 3, nous présentons plusieurs des méthodes déjà existantes pour obtenir des algorithmes sous-exponentiels dans les graphes d'intersection, telle que la théorie de la bidimensionalité, les méthodes de branchement ou les décompositions arborescentes par clique. De plus, ces méthodes ont amenées à une grande liste de résultats, pour différents problèmes algorithmique et classes de graphe. Nous présentons alors les principaux résultats du domaine.

Dans le chapitre 4, nous étudions certaines propriétés combinatoires du voisinage des sommets dans les graphes d'intersection. Plus particulièrement, si pour un graphe donné il est impossible de déduire des informations sur le voisinage d'un sommet en regardant les arêtes reliant les autres sommets entre eux, ce n'est pas forcément le cas pour les graphes d'intersections. En particulier, étant donné un ensemble de sommets A, il n'est pas toujours possible d'avoir, pour chaque sous-ensemble S de A, un sommet dont les sommets de A avec lesquels il partage une arête sont exactement les sommets de S. On peut alors s'intéresser, pour chaque ensemble A, au nombre maximum de voisinage distincts que l'on peut avoir sans contredire les propriétés de la classe de graphe. Cette notion est appelée complexité de voisinage et est le sujet de ce chapitre.

Dans le chapitre 5, nous présentons des bornes inférieures, existantes et nouvelles, au temps nécessaire pour résoudre certains problèmes de suppression de sommets dans les graphes d'intersection. Nous nous intéresserons plus précisément aux problèmes de couper les cycles et d'intersecter les cliques dans des graphes d'intersections de sommets. Ces bornes inférieures sont obtenues en construisant une instance du problème considéré qui traduit une instance que l'on pense être dur à résoudre.

Dans le chapitre 6, nous introduisons une nouvelle méthode appelée Almost-SQGM. Ce nouveau cadre est construit sur la méthode de la bidimensionnalité, présentée dans le chapitre 3, avec le même objectif d'obtenir des algorithmes FPT sous-exponentiels, mais qui peuvent être appliqués pour une gamme plus large de classes de graphes d'intersection. Nous présentons ensuite quelques applications de cette nouvelle méthode pour certaines classes de graphes spécifiques. Une grande force de cette méthode est qu'elle ne s'applique pas à un problème spécifique, mais une grande famille de problèmes algorithmique vérifiant des conditions qui sont respectées par un grand nombre de problèmes classiques.

Dans le chapitre 7, nous présentons des algorithmes FPT résolvant le problème FVS pour les graphes s-string et les graphes de pseudo-disques. Dans une première section, nous décrirons un algorithme général, qui n'utilise pas la représentation géométrique, et qui est FPT sous-exponentiel dans des classes de graphes très générales telles que les graphes s-string ou les graphes de pseudo-disques. Ensuite, nous nous concentrerons sur le graphe de pseudo-disques afin d'obtenir de meilleurs temps d'exécution, soit en généralisant certains des algorithmes existants pour les graphes de disques aux graphes de pseudo-disques, soit en utilisant une représentation géométrique et des arguments spécifiques pour les pseudo-disques.

Dans le chapitre 8, nous présentons d'abord un algorithme FPT sous-exponentiel résolvant, pour $r \geq 2$ fixé, le problème K_r-Hitting en temps FPT sous-exponentiel dans les classes de graphes creux (avec peu d'arêtes) et géométriques. La preuve prendra la forme d'un branchement sur une structure combinatoire, où une décision devra être prise sur où se trouve les sommets d'une solution dans cette structure. Si notre branchement doit traiter tous les choix possibles, la situation assure que chaque choix nous apporte beaucoup d'informations sur la solution finale, nous assurant qu'un petit nombre de tel choix est suffisant pour s'assurer qu'une solution existe respectant nos décisions. Ensuite, nous donnons des résultats supplémentaires pour le cas spécifique du problème K_3-Hitting, dans le cas où la complexité de voisinage est bornée.

Enfin, nous concluons dans le chapitre 9 et donnons quelques directions de recherche pour les sujets développés dans cette thèse.

Les définitions des problèmes algorithmiques mentionnés dans cette thèse se trouvent dans l'annexe.

Chapter 1
Introduction

Contents

Themes of the thesis

This thesis revolves around three main themes:

Computational complexity. Computational complexity theory aims at describing how efficient algorithms can be at solving computational problems. The first goal is then to design such algorithms, trying to find the best strategies for solving each problem. A complementary goal is then to justify that one cannot hope to do better than the best known algorithm, and so aims at capturing what makes a problem hard to solve.

Graphs. We encounter computational problems everywhere around us, such as when planning an event, finding the fastest way to go somewhere, solving a sudoku, etc. When developing an algorithm for solving one of them, one may prefer to use an universal language, making it easy to adapt strategies from one problem to another. Graph theory provides such a langage. A *graph* is the abstraction of a network of entities (called *vertices*) where some pairs are connected (said to be connected by an *edge*). For example, if one wants to represent the subway stations of a city by a graph, it may use as vertices the stations and put an edge between each pair of consecutive stations along a metro line. To abstract the interaction in a social network, the users can be represented by the vertices of a graph, and their relations can be represented with edges. Depending on the context it is sometimes interesting to restrict our attention to graphs with specifics properties. These properties can be of combinatorial nature (for example graphs where every vertex is connected to a bounded number of other vertices, defining *bounded degree graphs*), topological (for example asking that one can draw the graph on a piece of paper without any pair of edges crossing, defining *planar graphs*), algebraic, etc.

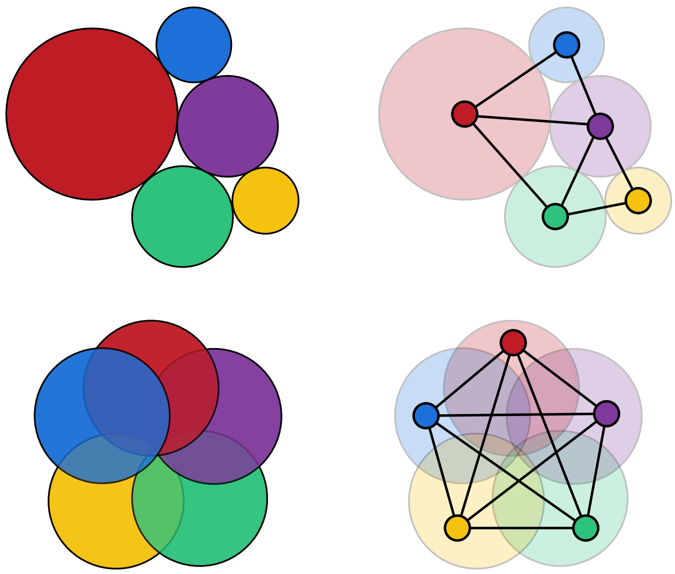

Figure 1.1: **Top:** Intersection graph of disks in contact. Observe the edges do not cross. **Bottom:** Representation of a disk graph.

Intersecting geometric shapes. Some properties are hard to express, and it is sometimes easier to describe a construction to obtain a class of graph, all sharing some properties. One famous example is considering disks in contact in the plane (interiors are disjoint), creating the graph with a vertex associated with each disk, and putting an edge between disks whose borders intersect. It is easy to see that one obtains a planar graph (see Figure 1.1), but the converse is also true. That is, each planar graph can be represented as the graph corresponding to a well-chosen set of disks in contact [Koe36]. Remark we could also authorize the interior of the disks to intersect, which would then give the so-called *disk graphs* (see Figure 1.1).

Motivation

Complexity theory aims at uncovering the border that delimits the problems that can be solved easily (typically, in polynomial time) and identifying the reasons why certain problems lie outside of this territory. However, when solving a specific problem in a particular situation the instances hard to solve may not arise. One goal lies in finding the most extensive characterization of instances solvable within a given time. This can be done for example by considering in addition to the size of the problem some other parameters describing the instance, that may help to bound the running time of an algorithm. This is the aim of parameterized complexity theory. Another solution is to restrict the instances to a subset of instances, described by some combinatorial properties, that can be used by the algorithm description or analysis.

A *vertex deletion problem* is a computational problem where given a n-vertex graph one must decide whether removing at most k vertices would give the graph certain properties, such as having no edges or being planar. Note that there are about n^k possible subsets of vertices to consider, so for an easily checkable property, we can solve the problem in time $n^{\mathcal{O}(k)}$. One might ask if there is a better strategy for solving this kind of problem, that would reduce the time to solve it to $n^{\mathcal{O}(\sqrt{k})}$, for example.

Question 1.1. *Which vertex deletion problems admit an algorithm with running time $n^{\mathcal{O}(k^{1-\varepsilon})}$ for some $\varepsilon > 0$?*

One may even hope for a stronger result. In this context, a *fixed-parameter tractable (FPT) algorithm* has running time $f(k)n^{\mathcal{O}(1)}$ for some computable function f. Such a running time then indicates that, if one is able to bound the number k of vertices we are allowed to remove, the vertex deletion problem then becomes polynomial. Then comes a second natural question.

Question 1.2. *Which vertex deletion problems admit an FPT algorithm?*

Many such problems are known, with running time of the form $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$. For those vertex deletion problems, now starts an optimality program: what are the best function f and exponent c such that it can be solved in time $f(k) \cdot \mathcal{O}(n^c)$? Observe that it may not be possible to optimize both simultaneously. In this thesis, we will be interested in improving the function in k. In particular we will be interested in *subexponential FPT algorithms*, which are algorithms with running time $2^{\mathcal{O}(k^{1-\varepsilon})}n^{\mathcal{O}(1)}$ for some $\varepsilon > 0$. Many problems are believed not to admit such algorithms, but one may want to consider a set of graphs satisfying certain properties, called a *graph class*. A new question is then:

Question 1.3. *Given a vertex deletion problem, what are the more general graph classes admitting a subexponential FPT algorithm?*

Or on the contrary, tracing the frontier from the other side:

Question 1.4. *Given a vertex deletion problem, what are the more* ***restricted*** *graph classes where it does not admit a subexponential FPT algorithm?*

Observe that for lower bounds, the more restricted a graph class is the better.

Finally, in the graph classes where such subexponential FPT algorithms do exist, a natural next question is to find an algorithm with the best possible running time.

Question 1.5. *Given a vertex deletion problem* Π *and a graph class* $\mathcal{G}$ *such that* Π *can be solved in subexponential FPT time for graphs in* $\mathcal{G}$*, how small can we take* $\eta > 0$ *such that* Π *can be solved in time* $2^{\mathcal{O}(k^\eta)} n^{\mathcal{O}(1)}$ *for the* n*-vertices graphs of* $\mathcal{G}$*?*

Outline of the thesis

The chapters are mostly independent, with the exception of Chapter 2, where are introduced definitions and general results appearing throughout this manuscript, Chapter 3 which contains a presentation of branching techniques used as preprocessing for our algorithms, and Chapter 4 which introduces results about neighborhood complexity used in the following chapters.

In Chapter 3, we present several existing methods for obtaining subexponential algorithms in intersection graphs and present some of the major results on the subject.

In Chapter 4, we investigate some combinatorial properties of the neighborhood of the vertices in intersection graphs.

In Chapter 5, we present existing as well as new time complexity lower bounds for some vertex deletion problems in intersection graphs.

In Chapter 6, we introduce a new method called Almost-SQGM. This new framework is built over the bidimensionality method, presented in chapter Chapter 3, with the same goal of obtaining subexponential FPT algorithms, but which can be applied for a broader range of intersection graph classes. We then present some applications of this new method for some specific graph classes.

In Chapter 7, present FPT algorithms solving the FVS problem for s-string graphs and pseudo-disk graphs. In a first section, we will describe a general algorithm that does not use the geometric representation and is subexponential FPT in very general graph classes such as s-string graphs or pseudo-disks graphs. Then, we will focus on the pseudo-disk graph in order to obtain better running times, by either generalizing some of the existing algorithms for disk graph to pseudo-disk graphs or by using a geometric representation and specific arguments for pseudo-disks.

In Chapter 8, first we present a subexponential FPT algorithm solving, for $r \geq 2$ fixed, the K_r-Hitting problem in subexponential FPT time in sparse and geometric graph classes. Secondly, we give additional results for the specific case of the K_3-Hitting problem.

Finally, we conclude in Chapter 9 and give some research directions for the subjects developed in this thesis.

The definitions of the computational problems mentioned in this thesis can be found in the appendix.

Instantiations of our algorithmic results are given in Table 1.1.

<table>
<tr><th>Graph class</th><th>Problem</th><th>Time complexity</th><th></th></tr>
<tr><td>Square</td><td rowspan="6">FVS</td><td>$2^{\widetilde{\mathcal{O}}(k^{9/10})} n^{\mathcal{O}(1)}$</td><td>Subsection 6.4.3</td></tr>
<tr><td rowspan="2">Contact segment</td><td>$2^{\widetilde{\mathcal{O}}(k^{7/8})} n^{\mathcal{O}(1)}$</td><td>Subsection 6.4.4</td></tr>
<tr><td>$2^{\widetilde{\mathcal{O}}(k^{5/6})} n^{\mathcal{O}(1)}$ $\star$</td><td>Subsection 7.4.5</td></tr>
<tr><td>s-string</td><td>$2^{\widetilde{\mathcal{O}}\left(s^{\mathcal{O}(1)} k^{52/53}\right)} n^{\mathcal{O}(1)}$</td><td>Section 7.1</td></tr>
<tr><td rowspan="3">Pseudo-disk</td><td>$2^{\widetilde{\mathcal{O}}(k^{6/7})} n^{\mathcal{O}(1)}$ $\star$</td><td>Subsection 7.4.3</td></tr>
<tr><td>$2^{\widetilde{\mathcal{O}}(k^{10/11})} n^{\mathcal{O}(1)}$</td><td rowspan="2">Section 7.3</td></tr>
<tr><td rowspan="4">TH</td><td rowspan="2">$2^{\widetilde{\mathcal{O}}(k^{3/4})} n^{\mathcal{O}(1)}$</td></tr>
<tr><td>Contact segment</td><td rowspan="3">Section 8.2</td></tr>
<tr><td>$K_{t,t}$-free string</td><td>$2^{\widetilde{\mathcal{O}}_t(k^{2/3})} n^{\mathcal{O}(1)}$</td></tr>
<tr><td>$K_{t,t}$-free d-DIR</td><td>$2^{\widetilde{\mathcal{O}}(k^{2/3}\sqrt{d}t^2 \log t)} n^{\mathcal{O}(1)}$</td></tr>
<tr><td>Pseudo-disk</td><td>K_r-Hitting</td><td>$2^{\widetilde{\mathcal{O}}_r\left(k^{(r+1)/(r+2)}\right)} n^{\mathcal{O}_r(1)}$</td><td>Section 8.1</td></tr>
</table>

Table 1.1: Examples of instantiations of the general cases considered if this thesis. The symbol $\star$ next to the complexity indicates that a geometric representation of the input graph is needed. The $\widetilde{\mathcal{O}}$ notation means up to a polylogarithmic factor.

Publications

The results presented in this thesis have led to four articles:

- **Subexponential Algorithms in Geometric Graphs via the Subquadratic Grid Minor Property: The Role of Local Radius,**
  with Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond.
  19th Scandinavian Symposium And Workshops On Algorithm Theory (SWAT 2024).
  Also on arXiv: arXiv:2306.17710.
  These results are presented in Chapter 5 and Chapter 6.

- **Feedback Vertex Set for pseudo-disk graphs in subexponential FPT time,**
  with Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond.
  50th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2024).
  Also on arXiv: arXiv:2410.23878.
  These results are presented in Section 7.4.

- **Kick the cliques,**
  with Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond.
  19th International Symposium on Parameterized and Exact Computation (IPEC 2024).
  Also on arXiv: arXiv:2407.01465.
  These results are presented in Chapter 8.

- **Pushing the frontiers of subexponential FPT time for Feedback Vertex Set,**
  with Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond.
  52nd EATCS International Colloquium on Automata, Languages, and Programming (ICALP 2025).
  Also on arXiv: arXiv:2504.17708.
  These results are presented in Section 7.1.

Chapter 2
Preliminaries

Contents

This chapter introduces the definitions and results used throughout the manuscript.

Basics. In this thesis, logarithms are binary. For a set S, we denote by $\mathcal{P}(S)$ its power set, and for $i \in \mathbb{N}$ we denote $\binom{S}{i}$ the set of subsets of size i of S. We assume the reader is familiar with the asymptotic notations $\mathcal{O}$, o, Ω, Θ. Additionally, setting $k \geq 1$, $f : \mathbb{N}^{k+1} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$, and for fixed $t_1, \ldots, t_k \in \mathbb{N}$ we denote $f(t_1, \ldots, t_k, n) = \mathcal{O}_{t_1,\ldots,t_k}(g(n))$ if the function $f'(n) : n \mapsto f(t_1, \ldots, t_k, n)$ satisfies $f'(n) = \mathcal{O}(g(n))$. Moreover, for $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ we denote by $f(n) = \tilde{\mathcal{O}}(g(n))$ if there exists a constant c such that $f(n) = \mathcal{O}(g(n) \log^c(n))$.

Given a set S a ***partition*** of S is a set $\mathcal{C} \subseteq \mathcal{P}(S)$ such that S is the disjoint union of $\mathcal{C}$.

2.1 Graphs

A ***graph*** G is a pair (V, E), with V a set denoted $V(G)$ and referred as the ***set of vertices*** of G, and $E \subseteq \binom{V}{2}$ denoted $E(G)$ and referred to as the ***set of edges*** of G. For simplicity, we may denote uv the edge $\{u, v\}$. For $u, v \in V(G)$ we say that u ***is adjacent to*** v if $uv \in E(G)$. Observe that with this definition the considered graphs are loopless (a vertex is not adjacent to itself), simple (there is at most one edge connecting two vertices), and undirected (u being adjacent to v is equivalent to v being adjacent to u). The ***order*** of a graph G is its number of vertices $|V(G)|$. In this thesis, all the considered graphs have finite order.

2.1.1 Basic graph concepts

A ***graph property*** is a property invariant by graph isomorphism. A ***subgraph*** of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, and G is then a ***supergraph*** of H. For $X \subseteq V(G)$, the ***subgraph induced*** by X, denoted $G[X]$, is the subgraph with vertex set X and edge set $\{e \in E(G) : e \subseteq X\}$. For $X \subseteq V(G)$ we denote $G - X$ the the graph induced by $V(G) \setminus X$. Let H be a graph, we say that G is H***-free*** if G does not contain H as a (not necessarily induced) subgraph, and H***-induced-free*** if G does not contain H as an induced subgraph.

Neighbors and degree. For a graph G, and $v \in V(G)$, we define the ***neighbors of*** v as the set $\{u \in V(G) : uv \in E(G)\}$ denoted $N_G(v)$, and the ***degree*** of v as $|N_G(v)|$ denoted $d_G(v)$. For both notations, we omit the subscript when it is clear from the context. We denote $\Delta(G) = \max_{v \in V(G)} d_G(v)$ the ***maximum degree*** of G. For $A \subseteq V(G)$, we use the notation $N(A) = (\cup_{v \in A} N(v)) \setminus A$. For $d \in \mathbb{N}$ we say that a graph G is d***-degenerate*** if every subgraph of G (including G itself) has a vertex of degree at most d. For a graph G, a set $S \subseteq V(G)$ is ***independent*** if there is no edge between two vertices of S, and ***dominating*** if for every $v \in V(G)$ either $v \in S$ or there is a vertex $u \in S$ such that $uv \in E(G)$.

Identification, contraction. For a graph G and u, v two distincts vertices, the ***identification*** of u and v is the operation resulting in the graph G' obtained from $G - \{u, v\}$ by adding a new vertex w with $N_{G'}(w) = N_G(u) \cup N_G(v)$. A ***contraction*** of an edge $uv \in E(G)$ is then the identification of the vertices u and v.

Paths, cycles, and distances. Given a graph G, a sequence of distinct vertices $(v_1, \ldots, v_k)$ with $k \geq 1$ is a ***path*** if for every $1 \leq i \leq k - 1$, the vertex v_i is adjacent to v_{i+1}. Moreover, if v_k is adjacent to v_1 such sequence is called a ***cycle***. The ***size*** of a path is its number of edges, and the ***size*** of a cycle is its number of edges as well. A graph such that there exists a path between every pair of vertices is said to be ***connected***. A ***connected component*** of a graph G is a set $S \subseteq V(G)$ such that $G[S]$ is connected and for every $S' \subseteq V(G)$ such that $S \subsetneq S'$ we have $G[S']$ not connected. A graph without cycle is a ***forest***, moreover, if the graph is connected we say it is a ***tree***. A set S such that $G - S$ is a forest is called a ***feedback vertex set***. A graph for which each connected component is a path is called a ***linear forest***. The ***distance*** between two vertices of a graph is the minimum size of a path linking them, and the ***diameter*** of a graph is the maximum distance between two of its vertices. The ***radius*** of a graph is the smallest integer $r \geq 0$ such that there exists a vertex v such that every vertex in the graph is at a distance at most r from v.

Classical graphs. A ***clique*** is a graph such that every pair of distinct vertices are adjacent. For $k \geq 2$ the clique on k vertices is denoted K_k. Given a graph G, a subgraph H of G that is a clique is said to be a ***complete subgraph***. When there is no ambiguity we also use ***clique*** to denote a subset of $V(G)$ inducing a complete subgraph. A graph G is ***bipartite*** if $V(G)$ can be partitioned into two sets A and B such that $E(G) \subseteq \binom{V}{2}$. The sets A and B are then said to be the ***parts*** of G (not necessarily unique). An equivalent definition of a bipartite graph is a graph whose cycles all have even size. A ***complete bipartite graph*** is a bipartite graph with parts A, B and every possible edge between A and B. For $i, j \in \mathbb{N}$ the complete bipartite graph with parts A, B such that $|A| = i$ and $|B| = j$ is denoted $K_{i,j}$. For $k \geq 1$ the (k, k)***-grid*** is the graph with vertex set $\{(i, j) : 1 \leq i, j \leq k\}$ and two vertices $(i, j), (i', j')$ are adjacent if and only if $|i' - i| + |j' - j| = 1$. For $d \geq 3$, d-dimensional grid graphs can be defined similarly, with d coordinates instead of 2.

Matching and bundles. A ***matching*** in a graph G is a set of edges $M \subseteq E(G)$ without common vertices. A ***maximum matching*** is then a matching that maximizes the size of M over all possible matchings in G. Such a matching is not hard to compute.

Theorem 2.1 ([Edm65])**.** *There is a polynomial time algorithm which, given a graph G, computes a maximum matching in polynomial time.*

A ***t-bundle*** [LPS+23] is a matching of size t plus a vertex connected to the $2t$ vertices of the matching. We say that B is a ***t-bundle of a graph*** G if $G[B]$ is a t-bundle plus possibly some extra edges. A set $S \subseteq V(G)$ is a ***t-bundle hitting set*** of G if $S \cap B \neq \emptyset$ for any t-bundle B of G.

2.1.2 Graph parameters

A ***graph parameter*** is a function from the set of all graphs to $\mathbb{N}$, invariant by isomorphism. One such example is the ***clique number*** of a graph G, denoted $\omega(G)$, which is the maximum order of a complete subgraph in G.

Treewidth. A ***tree decomposition*** of a graph G is a pair (T, β) where T is a tree and $\beta : V(T) \to \mathcal{P}(V(G))$ such that

- $\bigcup_{t \in V(T)} \beta(t) = V(G)$,
- for every edge e of G there is a $t \in V(T)$ such that $\beta(t)$ contains both endpoints of e, and
- for every $v \in V(G)$, the subgraph of T induced by $\{t \in V(t) : v \in \beta(t)\}$ is connected.

The sets $\beta(t)$ for $t \in V(T)$ are called the ***bags*** of the decomposition (T, β). The ***width*** of (T, β) is equal to $\max\{|\beta(t)| - 1 : t \in V(T)\}$ and the ***treewidth*** of G, denoted $\mathbf{tw}(G)$, is the minimum width over all the tree decompositions of G.

A result that will have important implications for us is that the treewidth can be approximated in exponential time parameterized by the treewidth:

Theorem 2.2 ([Kor22])**.** *There exists an algorithm that, given a n-vertex graph and an integer k, in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ either outputs a tree decomposition of G of size at most $2k + 1$ or correctly concludes that the treewidth of G is larger than k.*

The definition of tree decomposition can be extended to the weighted case: given a weight function $w : V(G) \to \mathbb{R}^+$, the ***weight*** of a bag is the sum of the weight of its vertices, the ***weighted width*** of a tree decomposition the maximum size of the bags. The ***weighted treewidth*** of G (for w) is the minimum weighted treewidth over all the tree decompositions of G.

2.1.3 Minors

Given two graphs H and G, a function $\phi : V(H) \to 2^{V(G)}$ is a ***minor model*** of H in G if

1. for every $x \in V(H)$, $G[\phi(x)]$ is connected and non-empty,
2. for every $x, y \in V(H)$, $\phi(x) \cap \phi(y) = \emptyset$, and
3. for every $\{x, y\} \in E(H)$, there exist $x' \in \phi(x)$ and $y' \in \phi(y)$ such that $\{x', y'\} \in E(G)$.

We say that H is a ***minor*** of G if there exists a minor model of H in G. Moreover, if there exists a model ϕ such that for every $x \in V(H)$ the graph $G[\phi(x)]$ has diameter at most r, we say that H is a ***depth-r minor*** of G.

Equivalently a graph H is a minor if it can be obtained by contracting edges of a subgraph G' of G. Additionally, a graph H that can be obtained by a sequence of edge contractions in G' an induced subgraph of G is called an ***induced minor*** of G.

2.1.4 Graph classes, sparsity, and separators

Graph classes. A ***graph class*** is a set of graphs, such that if G is contained in the set, then every graph G' isomorphic to G is also contained in the set. Given a certain graph operation, a graph class $\mathcal{G}$ is ***stable*** under this operation if applying the operation on any graph $G \in \mathcal{G}$ results in a graph $H \in \mathcal{G}$. A graph class is ***hereditary*** if stable under taking induced subgraphs, ***monotone*** if stable under taking subgraphs, ***(induced-)minor-closed*** if stable under taking (induced-)minors. For example the class of ***planar graphs*** is the set of graphs which can be represented in the plane such that none of their edges cross each other. An important property about this class is that it minor-closed, and so with the additional fact that K_5 and $K_{3,3}$ are not planar graphs, one obtains that if a graph is planar it does not contain either of these graphs as as a minor. Actually this property characterizes the class of planar graphs [Wag37]. Observe that for a given property Π, we can consider $\mathcal{G}_\Pi$ the class of graph having the property Π. Therefore, a graph class and a graph property are equivalent, the adjectives hereditary, monotone, and similar can then be used interchangeably for both.

Sparsity and separators. In this thesis, we use only a small portion of the vast theory of sparsity in graphs. For a wider view of the domain, we refer the reader to [NOdM12] for example.

A graph class $\mathcal{G}$ is ***sparse*** if there exists a constant c such that for any graph $G \in \mathcal{G}$ we have $|E(G)| \leq c|V(G)|$. For some function f, a graph class $\mathcal{G}$ is said to have ***expansion bounded*** by f if for every r, any depth-r minor H of any graph in $\mathcal{G}$ satisfies that $\frac{|E(H)|}{|V(H)|}$ is bounded by $f(r)$. If such a function exists we say that $\mathcal{G}$ has ***bounded expansion***, and if f can be taken to be a polynomial then $\mathcal{G}$ has ***polynomial expansion***. Observe that, taking $r = 0$, one obtains the result that a graph class having bounded expansion is sparse.

A ***balanced separator*** for a graph G is a subset $S \subseteq V(G)$ such that the connected components of $G - S$ have size at most $\frac{2}{3}|V(G)|$. A graph class $\mathcal{G}$ is said to have ***strongly sublinear separators*** if there exists $\varepsilon > 0$ such that any n-vertex graph $G \in \mathcal{G}$ has a balanced separator $S \subseteq V(G)$ with $|S| = \mathcal{O}(n^{1-\varepsilon})$. For example, in the case of planar graphs:

Theorem 2.3. *[LT79] Planar graphs on n vertices have a balanced separator of size* $\mathcal{O}(\sqrt{n})$.

For a hereditary graph class, having polynomially bounded expansion is equivalent to having strongly sublinear separators [PRS94, DN16, ER18]. Moreover, in such graph classes, the treewidth can be bounded:

Theorem 2.4 ([DN19])**.** *If a hereditary graph class $\mathcal{C}$ has strongly sublinear separators with function $f(n) = \beta n^{\delta}$, then the treewidth of any n-vertex graph in $\mathcal{G}$ is at most $15\beta n^{\delta}$.*

2.2 Computational problems

An ***algorithm*** is a finite sequence of mathematically rigorous instructions. A ***computational problem*** is a question or task that can be solved using an algorithm. A ***decision problem*** is a computational problem that asks if an instance I satisfies a given property. An instance satisfying the property is a ***Yes-instance*** (otherwise it is a ***No-instance***). Two instances I_1, I_2 are said ***equivalent*** if I_1 is a Yes-instance if and only if I_2 is a Yes-instance as well.

2.2.1 Vertex deletion problems

Given a fixed graph property, the associated ***vertex deletion problem*** Π is the decision problem of deciding, given a graph G and an integer k, if there exists a set S of at most k vertices such that $G - S$ satisfies this property. The minimum integer k for such a set to exist is then denoted $OPT_{\Pi}(G)$.

Examples. The vertex deletion problems considered the more often in this thesis are:

- Feedback Vertex Set (FVS), where we want to remove vertices to obtain a forest,
- the Odd Cycle Transversal (OCT) problem, where we want a bipartite graph,
- and the H-Hitting problems for H a graph, where we want a H-free graph.

For more variations of these problems (such as H-minor-Hitting where we want to obtain a H-minor-free graph or $\mathcal{F}$-Hitting where we want to intersect all subgraphs isomorph to graphs in a family $\mathcal{F}$), we refer the reader to the appendix where formal definitions of the problems considered in this thesis are given.

Approximation. For a vertex deletion problem Π and $\alpha > 1$, an ***α-approximation*** of the problem Π is a set S such that $G - S$ satisfies the graph property defining the vertex deletion problem Π and $|S| \leq \alpha \cdot OPT_\Pi(G)$.

Approximations for the Feedback Vertex Set problem and the H-Hitting problems will be often used in this manuscript:

Theorem 2.5 ([BBF99, BG96])**.** *A 2-approximation of a minimum feedback vertex set can be constructed in polynomial time.*

Theorem 2.6. *A $|H|$-approximation of a minimum H-hitting set can be constructed in time $n^{\mathcal{O}(|H|)}$.*

Proof. If there is a set $C \subseteq V(G)$ such that $G[C]$ is isomorphic to H, taking all the vertices of C in the approximation and removing the vertices of C from G, and repeating this process until there is no such copy of H gives a H-hitting-set. The ratio of approximation comes from the fact that for an optimal solution S and C one of the found set with $G[C]$ isomorphic to H, we have $S \cap C \neq \emptyset$. □

2.3 Computational complexity

We assume that the reader is familiar with the computational complexity classes ***P***, ***NP***, the notions of ***polynomial reduction***, the ***k-SAT*** problem, and ***dynamic programming*** (DP), whose definitions can be found for example in [CFK+15].

To obtain lower bounds for the problems considered in this thesis, a stronger assumptions than $P \neq NP$ is needed: The ***Exponential Time Hypothesis*** (***ETH***) is the assumption implying that the 3-SAT problem with n variables cannot be solved in time $2^{o(n)}$ [IPZ01].

2.3.1 Parameterized complexity and kernels

Only a small fraction of parameterized complexity will be introduced in the thesis, we then refer the interested readers to [CFK⁺15] for a more thorough presentation.

A ***parameterized problem*** is a decision problem which takes as an instance a pair (I, k) with I the ***input*** of the problem and k an integer, the ***parameter***. In this thesis, we only consider parameterized problems where the input is a graph.

A ***kernelization algorithm***, or in short a ***kernel*** for a parameterized problem Π is an algorithm which, given an instance (G, k) of Π returns in polynomial time an instance (G', k') with $k' \leq f_1(k)$ and $|V(G')| \leq f_2(k)$ with f_1, f_2 some computable functions, and such that (G', k') is equivalent to (G, k). If f_1, f_2 can be chosen to be polynomial (resp. linear), we then say that Π have a ***polynomial kernel*** (resp. ***linear kernel***).

A graph parameterized-problem is said to be ***fixed-parameter tractable*** (***FPT***) if there is an algorithm that can decide any instance (G, k) in time $f(k)n^{\mathcal{O}(1)}$ for some computable function f, where $n = |V(G)|$. A central result in parameterized complexity is that admitting a kernel and being FPT is equivalent.

We say that a parameterized problem admits a ***subexponential FPT algorithm*** if there exists $\varepsilon < 1$ such that the problem can be solved in time $2^{\mathcal{O}(k^{\varepsilon})}n^{\mathcal{O}(1)}$, with n the size of the instance. The main theme of this thesis will be the question of the existence of such algorithm when restricted to certain graph classes.

2.4 Geometric definitions

In this section, we present general geometric results used in this manuscript.

Given $\mathcal{S}$ a family of sets $V \subseteq \mathbb{R}^d$, and $z \in \mathbb{R}^d$, the ***ply*** of z (with respect to $\mathcal{S}$) is the number of sets of $\mathcal{S}$ containing z and the ***ply*** of $\mathcal{S}$ is the the maximum ply of a point of $\mathbb{R}^d$ with respect to $\mathcal{S}$.

When referring to a ***region of the plane***, we mean a subsets of the plane homeomorphic to a closed disk.

Given a region $\mathcal{D}$ of the plane, we denote its boundary by $\partial\mathcal{D}$ and call ***internal points*** any point of $\mathcal{D}$ that does not belong to $\partial\mathcal{D}$. When considering a family $\mathcal{S}$ of regions of the plane, a ***cell*** of $\mathcal{S}$ refers to a maximal connected region of $\mathbb{R}^2 \setminus \bigcup_{\mathcal{D}\in\mathcal{S}} \partial\mathcal{D}$. The ***ply*** of a cell of $\mathcal{S}$ is the ply of its points.

2.4.1 Pseudo-disks

A set of regions of the plane $\mathcal{S}$ forms a ***system of pseudo-disks*** if, for any two intersecting regions $\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{S}$, their borders $\partial\mathcal{D}_1$ and $\partial\mathcal{D}_2$ either cross properly exactly twice or are tangent exactly once.

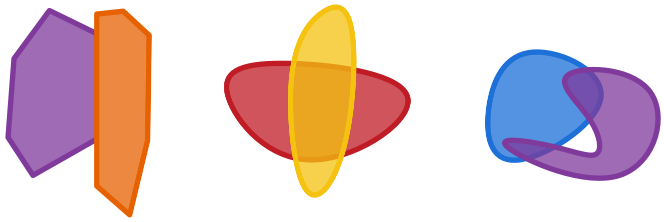

Figure 2.1: Three forbidden intersections between regions in systems of pseudo-disks.

A set of families of regions of the plane is said to have ***linear complexity*** if there exists a constant c such that for each family $\mathcal{F}$ of the set, the number of intersection points of the boundaries of the regions of $\mathcal{F}$ which are points of $\partial\left(\bigcup_{D\in\mathcal{S}} D\right)$ too is at most cn.

Theorem 2.7 ([KLPS86])**.** *The set of systems of pseudo-disks has linear complexity.*

A corollary of this result is a bound on the number of cells with ply at most a certain number.

Corollary 2.8. *[CS89] A system of n pseudo-disks has $\mathcal{O}(kn)$ cells of ply at most k.*

2.4.2 Fat and similarly sized objects

A region of the plane R is said to be ***α-fat*** if there are disks D_{in} and D_{out} with radius r_{in} and r_{out} such that $D_{in} \subseteq R \subseteq D_{out}$ and $r_{out}/r_{in} \leq \alpha$ (see Figure 2.2). For example, all disks are trivially 1-fat, and squares are $\sqrt{2}$-fat. The definition easily extends to higher dimensions by considering d-dimensional balls instead of disks.

The diameter of an object $S \subseteq \mathbb{R}^k$ is the maximum distance between two points of the object. A family of objects of $\mathbb{R}^k$ is said to be ***similarly sized*** when the ratio of the largest and smallest diameter of the objects in the family is bounded.

2.5 Intersection graph classes

Given a ground set U, and $S = (S_i)_i$ a finite family with $S_i \subseteq U$ for each i, the ***intersection graph*** of S is the graph G such that $V(G)$ contain a vertex u_i associated to each set S_i in S, and there is an edge between two vertices $u_i, u_j \in V(G)$ if $S_i \cap S_j \neq \emptyset$. When the sets are subsets of a ground set $\mathbb{R}^k$ for some $k \geq 1$, we then say that G is a ***geometric intersection graph***.

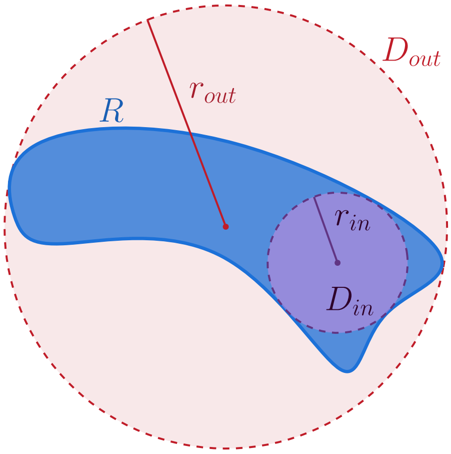

Figure 2.2: A region R which is r_{out}/r_{in}-fat.

In this thesis we will only consider the case where the vertices are path-connected subsets[1] of $\mathbb{R}^k$ for some k.

Sometimes, an algorithm taking as input an intersection graph will use the intersection model associated with this graph. On the contrary, an algorithm restricted to graphs of a specific intersection graph class is said to be ***robust*** if the input graph G is provided using one of the classical graph data structures, and so with no indication of the intersection model of G.

2.5.1 Intersection graphs of fat objects

Classical examples of intersection graphs of fat objects are the class of ***disk graphs***, the intersection graphs of closed disks in the plane, and the more restricted class of ***unit disk graphs*** for the intersection graph of unit disks, or equivalently, up to re-scaling, of disks all having the same diameter. More generally we define the ***d-dimensional (unit) ball graphs***, the intersection graphs of closed (unit) balls of the Euclidian distance in the d-dimensional Euclidian space. We denote ***square graphs*** the intersection graphs of squares in the plane, whose sides are axis parallel.

2.5.2 String graphs and their variants

Intersection graphs of arc-connected regions of the plane are in fact exactly the class of ***string graphs***, which are the intersection graphs of strings (i.e. Jordan

[1] A subset S of $\mathbb{R}^k$ is ***path connected*** if for every pair of points $u, v \in S$, there exists a continuous function $\varphi : [0,1] \to \mathbb{R}^k$ with $\varphi(0) = u$ and $\varphi(1) = v$.

arcs) in the plane. The result easily follows by the observation that a region of the plane can be filled by a curve covering its surface.

If instead of considering strings we restrict to the case of segments, we obtain the class of ***segment graphs***. Adding a restriction, if a segment graph can be represented with at most d different slopes, we call it a ***d-DIR graph***.[2]

Observe that when two non-parallel segments cross each other, they have only on point of crossing. This property in generalized, for any integer $s \geq 1$, with the class of ***s-string graphs***, the intersection graph of strings such that no two of them have more than s intersection points.

A stronger restriction is to consider the intersection graph of strings such that no three strings intersect on the same point and such that each string has at most d points shared with the other considered strings, this defines string graphs with ***edge degree*** at most d.

Another class of interest is the ***outerstring graphs***, the intersection graphs of strings all contained in a disk and each having one of its endpoints on the border of the disk.

2.5.3 Pseudo-disk graphs

Given a system of pseudo-disks $\mathcal{S}$, we denote by $G_{\mathcal{S}}$ the corresponding intersection graph, and this defines the class of pseudo-disk graphs. For any system of pseudo-disks, it is possible by doing minor perturbations which preserve being a system of pseudo-disks, to ensure that for any two intersecting elements $\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{S}$, either their borders do not intersect but in that case one is contained in the other, or their borders intersect on exactly two points while $\mathcal{D}_1 \cap \mathcal{D}_2$ contains internal points. Similarly, we can require that no point belongs to more than two boundaries (see Figure 2.3).

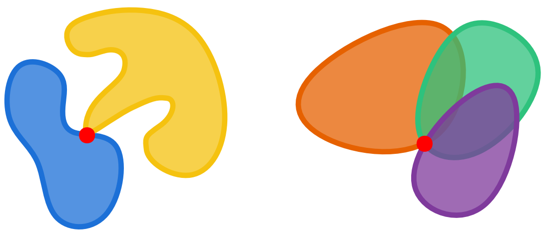

Figure 2.3: Two avoidable examples of intersections between regions in systems of pseudo-disks.

Note that pseudo-disk graphs are in particular string graphs and they form

[2]In general two d-DIR graphs may require different sets of slopes in their representation but in the case $d = 2$ it is known that the segments can be assumed to be axis-parallel, which we will do.

a common generalization of various classes of “fat” intersection graphs such as disk graphs, intersection graphs of axis-parallel squares, and more generally any intersection graph obtained from homothetic copies of a given convex set.

To bound the number of edges, a corollary of Corollary 2.8 is the following bound:

Theorem 2.9 ([LPS+24a]). *There exists a constant c such that any graph G on n vertices admitting a pseudo-disk representation with ply p has at most cpn edges.*

Observe that if a pseudo-disk graph G has a representation of ply p; we have $p \leq \omega(G)$, so we could replace p in the above bound by $\omega(G)$.

2.5.4 Contact graphs

For the classes of intersection graphs of regions of the plane, we can consider a ***contact*** version, where the additional condition is that the interior of the considered regions of the plane do not intersect, i.e. the intersection occurs only on the border of the regions.

A well studied case is the following.

Theorem 2.10 ([Koe36]). *The intersection graphs of disks in contact are exactly the planar graphs.*

More generally, ***map graphs*** are defined as the contact graph of regions of the plane.

As strings do not have an interior, in order to define a contact counterpart of the class and its subclasses require to tweak a bit the definition. We will instead require that the representation does not contain crossings. That is, two strings either intersect tangentially, or they intersect at an endpoint of one of them. Similarly, in a segment contact representation, any point belonging to two segments must be an endpoint of at least one of these segments. If a point belongs to several strings or segments, the above property must hold for any pair of them. This defines ***contact string graphs***, ***contact segment graphs***, ***contact d-DIR graphs*** and so on for other subclasses of string graphs.

2.5.5 Balanced separators and sparsity for geometric graphs

String graphs are not necessarily sparse as for example they include the cliques of any order. However, if they do not contain a big biclique as a subgraph then they are necessarily sparse, as shown by the following theorem.

Theorem 2.11 ([Lee17]). *There is a constant c such that for every integer $t > 0$, every $K_{t,t}$-free string graph on n vertices has at most $ct(\log t) \cdot n$ edges.*

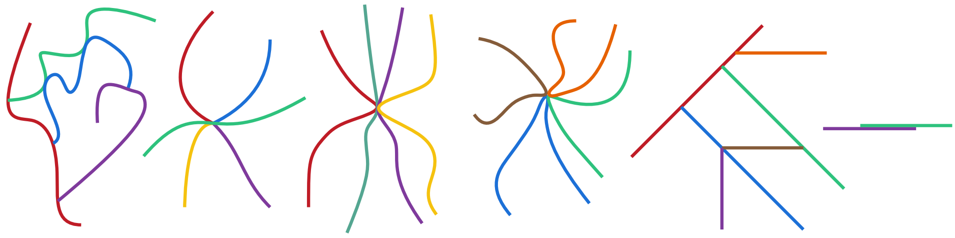

Figure 2.4: From left to right, four representations of contact string graphs, then a representation of 3-DIR contact segment graph, and finally on the right an example of an intersection between segments not allowed in a representation of a contact segment graph.

Moreover, the size of balanced separators is sublinear in the number of edges for string-graphs.

Theorem 2.12 ([Lee17]). *m-edge string graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$. And so by Theorem 2.4 it satisfies* $\mathbf{tw}(G) = \mathcal{O}(\sqrt{m})$.

Putting those two results together[3], we obtain the following bound on the treewidth of string graphs, which will be a key tool for obtaining subexponential FPT algorithms.

Corollary 2.13. *A n-vertices $K_{t,t}$-free string graph G satisfies* $\mathbf{tw}(G) = \mathcal{O}\left(\sqrt{nt}\log t\right)$.

Separators in disks, balls and pseudo-disk graphs. For the class disks, and more generally of d-dimensional ball with $d \geq 2$, it is known that bounding the ply is enough in order to obtain sublinear balanced separators.

Theorem 2.14 ([MTTV97]). *n-vertex intersection graphs of balls in $\mathbb{R}^d$ with ply p have balanced separators of size $\mathcal{O}(p^{1/d}n^{1-1/d})$. And so by Theorem 2.4 it satisfies* $\mathbf{tw}(G) = \mathcal{O}(p^{1/d}n^{1-1/d})$.

Observe that putting this result for $d = 2$ together with Theorem 2.10 gives another proof of Theorem 2.3.

For the class of pseudo-disk graphs, the results of string graphs still holds by inclusion. Moreover, in the case of bounded ply Theorem 2.12 and Theorem 2.9 give the following result, which is a generalization of Theorem 2.14 for $d = 2$:

Corollary 2.15. *Any pseudo-disk graph on n vertices admitting a pseudo-disk representation with ply p has treewidth $\mathcal{O}(\sqrt{pn})$.*

[3] Note that an error was found in the proof of the above results, but a claim by the author was made that the proof can be corrected in the case of string graphs (see [BR24]). Moreover, the previous bound [Mat14] gives similar results, up to logarithmic factors.

2.5.6 Inclusions between the considered graph classes

Some inclusions between intersection graph classes are trivial, such as a unit disk graph is a disk graph, or an interval graph is a segment graph. However, for some other inclusions, the inclusion is less trivial to prove, as for the aforementioned example of contact-disk graphs being exactly the planar graphs. In Subsection 7.4.5 it is shown that contact segment graphs are pseudo-disk graphs too. A nontrivial result that is of interest for us is that segment graphs are 1-string graphs, as two non parallel segments cross at most once and the parallel case can be prevented [KM94]. Another inclusion that can be observed is that square graphs are pseudo-disk graphs. A part of the lattice formed by the inclusions between the intersection graph classes considered in this thesis is represented in Figure 2.5.

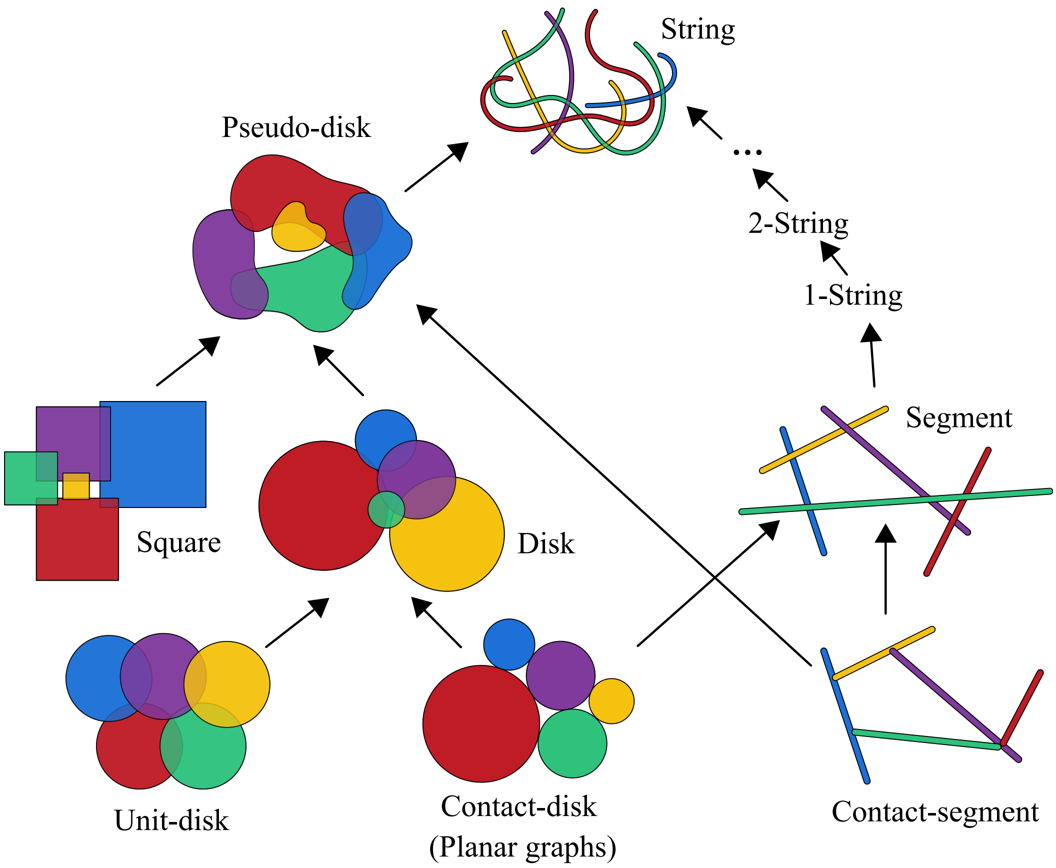

Figure 2.5: Inclusions between some of the geometric graph classes mentioned in this thesis.

Chapter 3
Techniques and results for subexponential algorithms in intersection graphs

Contents

This chapter presents a wide range of existing tools for obtaining subexponential algorithms that solve computational problems on graphs when restricted to intersection graphs or other graph classes of interest. For each of these techniques, we will discuss some of their applications, but also some of their limitations, in order to introduce some of our results presented in the following chapters.

3.1 Standard techniques

3.1.1 Branching

In this section, we present the standard branching techniques used throughout the manuscript. For a more throughout analysis of the topic we refer the interested reader to the chapter "Bounded search tree" of [CFK+15].

The case of Independent Set in string graphs

Let us start with an example where the branching tree is not a path. We will consider the algorithm devised in [FP11] for solving Independent Set in string graphs, following the same presentation as in [BR19].

Let us consider the following branching rule for solving an instance (G, k) with G a n_0-vertex graph. Given a vertex $v \in V(G)$, observe that there exists a solution containing v if and only if the graph $G - \{v \cup N(v)\}$ has a solution of size $k - 1$, and a solution not containing v exists if and only if $G - \{v\}$ has a solution of size k. Then by solving the instances $(G - \{v \cup N(v)\}, k - 1)$ and $(G - \{v\}, k)$, we can solve the instance (G, k).

In the case of a string graph G with n vertices, applying this branching on a vertex of degree at least $d = \lceil n^{1/3} \rceil$ and by denoting $F(N)$ the maximal number of subinstances generated for graphs with at most n vertices, we obtain, denoting $c = d + 1$:

$$\begin{aligned}
F(n) &\leq F(n-c) + F(n-1) \\
&\leq F(n-c) + F(n-c) + F(n-2) \\
&\leq 2F(n-c) + F(n-2) \\
&\leq 3F(n-c) + F(n-3) \\
&\dots \\
&\leq cF(n-c) + F(n-c) \\
&= (c+1)F(n-c) \\
&\leq (c+1)^2 F(n-2c) \\
&\dots \\
&= (c+1)^{\mathcal{O}(n/c)}.
\end{aligned}$$

The number of instances generated for solving the initial instance is then bounded by $F(n_0) = (c+1)^{\mathcal{O}(n_0/c)} = 2^{\mathcal{O}(n_0^{2/3} \log n_0)}$. This bound can be viewed as a bound on the number of leaves in the branching tree. Since a binary tree has fewer internal nodes than leaves this bounds the overall number of instances considered during the branching. The internal nodes correspond to a call to the branching procedure, which can be performed in $n^{\mathcal{O}(1)}$ time. The generated instances (G', k') (corresponding to the leaves) satisfy $\Delta(G) < d$, so for each instance (G', k') and denoting $n = |V(G')| \leq n_0$, we have $|E(G')| \leq d|V(G')| \leq dn$. Therefore, by Theorem 2.4 we have $\mathbf{tw}(G') = \mathcal{O}(\sqrt{dn}) = \mathcal{O}(n^{2/3})$. As Independent Set can be solved in time $2^{\mathcal{O}(\mathbf{tw}(G'))} n^{\mathcal{O}(1)}$ [CFK+15], we obtain an overall running time of $2^{\mathcal{O}(n_0^{2/3} \log n_0)} \cdot 2^{\mathcal{O}(n^{2/3})} = 2^{\mathcal{O}(n_0^{2/3} \log n_0)}$.

A general case for branching

Let us now describe a more general setting where instead of the number of vertices, the procedure will decrease some parameter k, using a branching rule which, for some fixed c and r, satisfies the following properties:

- If the parameter is negative, the branching stops.
- If a branching rule applies, the parameter decreases by at least c in r of its branches, and there is at most one remaining branch in which the parameter decreases by at least 1.

For the Independent Set branching example presented above we can then take $r = 1$ and $c = \lceil n^{1/3} \rceil + 1$.

Denoting $F(k)$ the maximal number of subinstances generated for a parameter at most k then satisfies:

$$\begin{aligned} F(k) &\leq rF(k-c) + F(k-1) \\ &\leq (cr+1)F(k-c) \\ &= 2^{\mathcal{O}\left(\frac{k}{c}\log(cr+1)\right)}. \end{aligned}$$

Therefore, taking k_0 the parameter of the initial instance, n the number of vertices of its graph, and assuming $c = k_0^{\varepsilon_1}$ with $0 < \varepsilon_1 < 1$ and $r = k^{\mathcal{O}(1)}$, we then get $F(k_0) = 2^{\mathcal{O}\left(k_0^{1-\varepsilon_1} \log k_0\right)}$ subinstances. Now assuming that each branching can be done in subexponential FPT time $2^{\mathcal{O}(k_0^{\varepsilon_2})} n^{\mathcal{O}(1)}$ and each instance where we do not branch can be solved in $2^{\mathcal{O}\left(k_0^{\varepsilon_3}\right)} n^{\mathcal{O}(1)}$ the overall running time for solving the initial instance is then

$$2^{\mathcal{O}\left(k_0^{1-\varepsilon_1} \log(k_0)\right)} \cdot \left(2^{\mathcal{O}\left(k_0^{\varepsilon_2}\right)} n^{\mathcal{O}(1)} + 2^{\mathcal{O}\left(k_0^{\varepsilon_3}\right)} n^{\mathcal{O}(1)}\right) = 2^{\mathcal{O}\left(k_0^{\varepsilon_4} \log k_0\right)} n^{\mathcal{O}(1)}$$

with $\varepsilon_4 = \max(1 - \varepsilon_1, \varepsilon_2, \varepsilon_3)$.

Let us now describe examples of applications of this branching technique for Feedback Vertex Set and Triangle Hitting. For most of the results, we only give an informal description of the branchings, as the formal statement with the exact complexity will be given in the following chapters.

Cliques

Here we present a folklore branching technique that, when considering the problem Feedback Vertex Set or Triangle Hitting, generates a set of instances with small clique number.

The branching will make use of the following observation.

Lemma 3.1. *Given a graph G, a feedback vertex set or a triangle hitting set S, and $C \subseteq V(G)$ such that $G[C]$ is a clique, then $|C \setminus S| \leq 2$.*

Proof. Suppose on the contrary that there is $C' \subseteq C$ with $|C'| \geq 3$ and $S \cap C' = \emptyset$, then $G[C']$ is a clique of size at least 3, and so contains a triangle. However, $G[C']$ is a subgraph of $G - S$, which is a triangle-free graph. □

Before explaining the branching and its analysis, we need to explain how to find big cliques in the considered graph. The problem is known to be hard when considering any graph, even to approximate. Note that in geometric graph classes, its difficulty is not well understood, with for example a polynomial algorithm only recently found in the restricted case where we only consider disks of at most 2 distinct radii [KM24]. Regarding approximation, an EPTAS is known for disk graphs [BBB+21], later generalized to the intersection graphs of convex sets with additional geometric properties [BGM20]. However, for pseudo-disk graphs the existence of such EPTAS is an open question, even when considering convex pseudo-disks [Gre22]. To avoid this problem, we will make use of an approximation.

Lemma 3.2. *Given a graph G and integers $3 \leq p \leq k$, either G does not have a feedback vertex (respectively a triangle hitting set) of size at most k or we can find in time $2^{\mathcal{O}(p \log k)} n^{\mathcal{O}(1)}$ a clique of size p in G if it exists.*

Proof. We start by computing a 2-approximation S for the FVS problem in G in polynomial time using Theorem 2.5 (respectively Theorem 2.6 and a 3-approximation S for the TH problem). If $|S| > 2k$ (respectively $|S| > 3k$) we know that G does not have a feedback vertex set (respectively a triangle hitting set) of size at most k. Otherwise, suppose G contains a clique C of size p in G. We know that $|C \setminus S| \leq 2$ by Lemma 3.1. Guessing $p - 2$ vertices of C in S and then the two remaining vertices in $V(G)$ takes time $|S|^{p-2} n^2 = 2^{\mathcal{O}(p \log k)} n^{\mathcal{O}(1)}$. □

We are now ready to describe the branching which given an instance (G, k) of Triangle Hitting or Feedback Vertex Set and an integer $t \geq 3$, returns a set $\mathcal{I}$ of instances (G', k') with $k' \leq k$ and G' a K_t-free graph.

First, the branching process searches for a clique of size t using Lemma 3.2. If it does not find such a clique then the graph is K_t-free as wanted, and we can add the instance in the final set $\mathcal{I}$. Otherwise, given H such that $G[H]$ is a clique of size t, by Lemma 3.1 we know that a potential solution S have $|H \setminus S| \leq 2$. The branching is then for each pair$\{u, v\} \in \binom{2}{H}$ to consider the instances $(G - (H - \{u, v\}), k - (t - 2))$.

Bicliques

In the case of bicliques, we only consider Feedback Vertex Set as having a large biclique does not imply having a large triangle hitting set.

Lemma 3.3. *Given a graph G, S a feedback vertex set and $H \subseteq V(G)$ such that $G[H]$ is isomorphic to $K_{t,t}$ for some $t \geq 2$ with sides A and B, then either $|A \setminus S| \leq 2$ or $|B \setminus S| \leq 2$.*

Following the same ideas as the case of cliques, we can then create a set of $K_{t,t}$-free instances.

Bundles

In this section, we present the "virtual branching" trick of [LPS+22] which handles large triangle bundles. As discussed later (in Subsection 7.1.1), the presence of large triangle bundles is a source of difficulty in instances of problems like Feedback Vertex Set and Triangle Hitting. Thus, we consider here a setup where we are given an instance (G, k) of one of these two problems, a $O(1)$-approximation M of the considered problem (where $|M| = \mathcal{O}(k)$), a value p (typically $p = k^{\epsilon}$), and the objective here is to get rid off any p-bundle (v, H) where the matching H is such that $V(H) \cap M = \emptyset$, and the vertex v (connected to the $2p$ vertices of H) is in M.

Observe that, compared to the case of cliques, the difficulty is that we cannot perform a classical branching. Indeed, we could consider one "small" branch where we take v (and k decreases by one), and one "heavy branch" where we avoid v, but in the heavy branch, even if the solution has to use p vertices to take one vertex in each edge of H, we does not know which vertex to pick in each edge (and branching to decide which one would not be subexponential).

The virtual branching of [LPS+22] resolves this problem by considering the two following branches:

- either take v (leading to instance $(G \setminus v, k - 1, M \setminus v)$),

- or absorb H (leading to instance (G, k, M'), with $M' = M \cup V(H)$)

Notice that in the second branch, to the price of a slight increase in the size of M, the parameter virtually decreases by $|H|$, in the sense that a solution avoiding v must hit each edge of H. Therefore, we consider these edges of H as "marked", and maintain that the size of the matching of marked edges in M' must be at most k. More formally, this procedure leads to the following statement.

Given an instance graph (G, k) of Feedback Vertex Set or Triangle Hitting and an integer p, there exists a $2^{\mathcal{O}\left(\frac{k}{p}\log k\right)} n^{\mathcal{O}(1)}$-time algorithm that produces a collection $\mathcal{C}$ of size $2^{\mathcal{O}\left(\frac{k}{p}\log k\right)}$ of tuples (G', k', M') such that:

- (G', k') is such that $k' \leq k$, and G' is an induced subgraph of G,
- $|M'| = \mathcal{O}(k)$ and there is no p-bundle of the form a vertex $v \in M'$ and p edges of $G[N(v) \setminus M']$.
- (G, k) is a Yes-instance of the original problem if and only if there exists $(G', k', M') \in \mathcal{C}$ such that (G', k') is a Yes-instance of the original problem.

Notice that in the instances in $\mathcal{C}$, large triangle bundles may still be present in the graph, but are confined into a small set M'. Notice also that once it is done, for each (G', k', M') obtained, by adding for each $v \in M'$ a maximum matching of edges in $G - M'$ (of size at most p thanks to the branching) we then obtain a set $\tilde{M}$ of size $\mathcal{O}(pk)$ such that $N_{G'}(\tilde{M})$ is an independent set. Stated differently each triangle of G' has at most one vertex outside of $\tilde{M}$. We now give a more formal statement of the obtained procedures after applying the procedure for the cliques and bundles.

Lemma 3.4 ([LPS+22]). *Let $\mathcal{G}$ be a hereditary graph class where the maximum clique can be α-approximated for some constant factor $\alpha \geq 1$ in polynomial time. There exists a $2^{\mathcal{O}\left(\frac{k}{p}\log k\right)} n^{\mathcal{O}(1)}$-time algorithm that, given an instance (G, k) of FVS and an integer $p \in [6\alpha, k]$, where $G \in \mathcal{G}$, returns a collection $\mathcal{C}$ of size $2^{\mathcal{O}\left(\frac{k}{p}\log k\right)}$ of tuples (G', k', M) such that:*

1. *For any $(G', k', M) \in \mathcal{C}$, (G', k') is an instance of FVS where G' is an induced subgraph of G, $\omega(G') \leq p$, and $k' \leq k$;*
2. *$|M| = \mathcal{O}(pk)$, and every triangle of G' has at least two vertices in M; and*
3. *(G, k) is a Yes-instance of FVS if and only if there exists $(G', k', M) \in \mathcal{C}$ such that (G', k') is a Yes-instance of FVS.*

The idea behind this technique appeared to be very useful to us as we pushed it in the two following ways.

- In Chapter 7 we consider Feedback Vertex Set, and the previous p-bundle (p, H) is replaced by what we call a $\widetilde{K_{p,p}}$, which informally is a minor model of a $K_{p,p}$ whose one side is in M and the other one outside.
- In Chapter 8 we consider K_r-Hitting, and the previous p-bundle (v, H) is replaced by tuple $(X_M, Y^1, \dots, Y^p)$ where the center $X_M \subseteq M$ plays the role of v, and the Y^i are disjoint subsets (called r-petal) with $Y^i \cap M = \emptyset$ such that $G[X \cup Y^i]$ is the clique of size r, and play the role of H.

3.1.2 Bounding the treewidth

Most of the problems considered in this thesis can be solved for n-vertex graph in time $2^{\widetilde{\mathcal{O}}(\mathbf{tw}(G))} n^{\mathcal{O}(1)}$. Thus, obtaining a bound on the treewidth of the form $\mathbf{tw}(G) = \mathcal{O}(n^{1-\varepsilon})$ or $\mathbf{tw}(G) = \mathcal{O}(k^{1-\varepsilon})$ yields a subexponential time algorithm or a subexponential FPT algorithm, respectively. We present in this section classical methods for obtaining such bounds on the treewidth.

Bidimensionality and extensions

We start this review of methods by the so-called bidimensionality theory. This framework was developed in [DFHT05] in order to obtain subexponential FPT algorithms and has many advantages:

- It applies to a wide range of problems,
- and to a wide range of graph classes,
- and in many cases the obtained running times are ETH tight.

First, we define the graph classes where the framework applies. The definition will make use of the ***grid number*** of a graph G, denoted $\boxplus(G)$, which is the graph parameter corresponding to the biggest t such that the (t, t)-grid is a minor of G.

Definition 3.5 ([DFHT05, FLS18])**.** *Given $c < 2$, a graph class $\mathcal{G}$ has the* **sub-quadratic grid minor property** *for c (***SQGM** *for short), denoted $\mathcal{G} \in SQGM(c)$, if $\mathbf{tw}(G) = \mathcal{O}(\boxplus(G)^c)$ for all $G \in \mathcal{G}$. We write $\mathcal{G} \in SQGM$ if there exists $c < 2$ such that $\mathcal{G} \in SQGM(c)$.*

Planar graphs and H-minor-free for any H are examples of graph classes having the SQGM property, both having the $SQGM(1)$ property [DH08]. However, most of the geometric intersection graph classes considered in this thesis do not have the SQGM property. The class of unit disk graphs, for example, does not have the SQGM, as this graph class contains the graphs K_t for every t, but as we have $\mathbf{tw}(K_t) = t - 2$ and $\boxplus(K_t) \leq \sqrt{|V(K_t)|} = \mathcal{O}(\sqrt{t})$, this rules out the existence of a constant $c < 2$ such that the treewidth of G a unit disk graph is $\mathcal{O}(\boxplus(G)^c)$.

Remark 3.6. *In general, every graph G satisfies $\mathbf{tw}(G) = \mathcal{O}(\boxplus(G)^c)$ for some $c < 10$ [CT21]. However, the additional property $c < 2$ in the SQGM definition will be a key property in order to obtain subexponential FPT algorithms with the bidimensionality framework.*

We now define the problems for which the bidimensionality framework applies.

Definition 3.7 ([DFHT05])**.** *A graph parameter p is* **minor-bidimensional** *if:*

- *For G a graph and H a minor of G, we have $p(H) \leq p(G)$.*
- *There exists $\delta > 0$ such that $p(R) \geq \delta r^2$ for the (r,r)-grid R.*

For such a parameter p, the parameterized problem which, given a graph G, asks if $p(G) \leq k$ is then said to be a **minor-bidimensional** *problem.*

Observe that Feedback Vertex Set and Vertex Cover are examples of minor-bidimensional problems, but not Odd Cycle Transversal or K_3-Hitting.

We are now ready to state the cornerstone of the bidimensionality framework.

Theorem 3.8 ([DFHT05])**.** *Given a bidimensional problem Π that can be decided in time $2^{\widetilde{\mathcal{O}}(\mathbf{tw}(G))}n^{\mathcal{O}(1)}$, and a graph class $\mathcal{C}$ having the SQGM(c) property, the problem Π can be solved for the graphs in $\mathcal{C}$ in time $2^{\widetilde{\mathcal{O}}(k^{c/2})}n^{\mathcal{O}(1)}$.*

Proof. $\mathcal{C}$ having the $SQGM(c)$ property, $\mathbf{tw}(G) \leq q\boxplus(G)^c$ for some constant q. As Π is minor-bidimensional, there exists a constant $\delta > 0$ such that for the (r,r)-grid H_r, $OPT_\Pi(H_r) \geq \delta r^2$. Using Theorem 2.2, we can decide if $\mathbf{tw}(G) > \frac{qk^{c/2}}{\delta}$, in which case $OPT_\Pi(G) > k$ and we have a No-instance, or $\mathbf{tw}(G) = \mathcal{O}\left(\frac{qk^{c/2}}{\delta}\right) = \mathcal{O}\left(k^{c/2}\right)$, in which case the instance can be decided using the $2^{\widetilde{\mathcal{O}}(\mathbf{tw}(G))}n^{\mathcal{O}(1)}$-time algorithm. □

The above theorem has limitations, with for example the previous remark that unit disk graphs, and many other geometric intersection graph classes, do not satisfy the SQGM property. Another limitation is that natural problems such as K_r-Hitting for $r \geq 3$ are not bidimensional. Despite this, extensions of the bidimensionality theory were devised in follow-up works, which enabled the design of subexponential FPT algorithms for a wide range of problems and geometric graph classes.

One such idea, which applies to unit disk graphs, is to observe that if the only obstacle to the existence of strongly sublinear separators is the presence of large cliques, the branching step described in Subsection 3.1.1 can be applied as a preprocessing. This strategy yields the following results.

Theorem 3.9 ([FLS18]). *The* Feedback Vertex Set *problem can be solved for map graphs and unit disk graphs in time* $2^{\widetilde{\mathcal{O}}(k^{3/4})}n^{\mathcal{O}(1)}$.

Theorem 3.10 ([GKT14]). *The* Feedback Vertex Set *problem for bounded edge degree string graphs or H-free intersection graph of convex similarly sized fat objects can be solved in time* $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$.

Similarly to the above results, in Chapter 6 we extend the bidimensionality framework to new intersection graph classes by characterizing sufficient additional conditions in order to obtain strongly sublinear separators.

Note that this section is only a very quick overview of the bidimensionality theory. For example, we restricted ourselves to graph parameters bidimensional for the minor relation, but the method can be applied to the contraction relation as well, which would give algorithms for other problems such as Dominating Set. See [DFHT05] and [BT22] for example for a presentation of this variant.

Moreover, another strength of this framework is its capacity to be extended in many directions, such as obtaining approximation algorithms [FLS18] or kernels [FLST20].

Cliques separators and structural decomposition

A geometric intersection graph class containing arbitrary large cliques does not admit small separators, and so there is no hope of bounding the treewidth of the considered graphs by a function sublinear in the number of vertices. A recurrent idea, particularly in the case of unit disk graphs, is to instead find separators which are union of a small number of cliques of arbitrary size. Then, strategies similar to the dynamic programming on a tree-decomposition can be applied, but this time by taking advantage of the clique decomposition. Example of applications of this method can be found in [FLP+19] where a $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$ algorithm is devised solving Feedback Vertex Set in unit disk graphs (improving the previous running time of Theorem 3.9). This running time was once again improved in [AO21], removing the $\log k$ factor in the exponent and obtaining an ETH-tight running time.

In [dBBKB+20] a framework is designed around this idea. More precisely, the idea is to use weighted tree decomposition with a well-chosen weight function and a partition of the vertices of the graph, such that:

1. Geometric intersection graph classes of interest, such as intersection of convex fat objects, have a weighted tree-width sublinear in the number of vertices.

2. It is possible to solve the considered problems in time exponential in the weighted-treewidth.

The following results were obtained using this framework.

Theorem 3.11 ([dBBKB+20]). *In dimension d, Independent Set (and so Vertex Cover) for convex fat objects when a representation is given can be solved in time* $2^{\mathcal{O}(n^{1-1/d})}$. *Moreover, there is a robust algorithm for intersection graphs of similarly sized fat objects with the same running time for solving the problems Independent Set (and so Vertex Cover), Dominating Set, and FVS.*

And in dimension 2, for a larger class:

Theorem 3.12 ([dBKBMT23]). *Independent Set (and so Vertex Cover) and FVS can be solved in time* $2^{\widetilde{\mathcal{O}}(n^{2/3})}$ *for pseudo-disk graphs.*

Observe however that these results are not FPT, but in the case of Vertex Cover where a linear kernel outputting an induced subgraph of the initial graph is known [CFJ04], we can then obtain FPT algorithms from these results.

Corollary 3.13. *Vertex Cover can be solved in time:*

- $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ *for intersection graphs of convex fat objects if a representation is given, and similarly sized fat objects even without representation.*

- $2^{\widetilde{\mathcal{O}}(k^{2/3})}n^{\mathcal{O}(1)}$ *for pseudo-disks given a representation.*

An important lemma

In this paragraph, we discuss the following lemma.

Lemma 3.14. *([LPS+22]) Let G be a disk graph that has some realization of ply p, and let* $M \subseteq V(G)$ *be such that for all* $v \in M$, $N_G(v) \setminus M$ *is an independent set, there does not exist a vertex in* $V(G) \setminus M$ *whose neighborhood is contained in M, and* $G - M$ *has treewidth w. Then, the treewidth of G is* $\mathcal{O}(max\{\sqrt{|M| \cdot w} \cdot p^{2.5}, w \cdot p\})$.

Recall that in the setup we consider, we are often given an instance (G, k) (typically of Feedback Vertex Set) and set M which is a feedback vertex set with $|M| = \mathcal{O}(k)$. By definition of M, we have $\mathbf{tw}(G \setminus M) \leq 1$, implying a trivial bound $\mathbf{tw}(G) \leq 1 + |M|$. Thus, the crucial point in Lemma 3.14 is to obtain a bound of the form $\mathbf{tw}(G) \leq f(|M|)$, where f is sublinear. Notice also that this lemma is typically applied after performing the preliminary branching on cliques, implying that p and w are $\mathcal{O}(k^\epsilon)$, and thus this lemma tells us that $\mathbf{tw}(G)$ is sublinear in k, implying subexponential parameterized algorithm using state of the art dynamic programming. Finally, let us comment on the hypothesis that for all $v \in M$, $N_G(v) \setminus M$ is an independent set. This may appear like a technical condition, but it in fact it corresponds to the property of item 2 from Lemma 3.4 obtained thanks to the virtual branching, and it is a necessary condition (see Subsection 7.1.1). This result was used in [LPS⁺22] for obtaining robust FPT algorithms solving Feedback Vertex Set and Odd Cycle Transversal for disk graphs. Observe that Lemma 3.4 was designed as a preprocessing for this lemma, and so the tuples (G, k, M) generated by the preprocessing satisfy the conditions of Lemma 3.14, with the exception of the condition of not having a neighborhood included in M, and which will be discussed below.

The analysis of the running time of the same algorithms was later improved in [ACO23]:

Theorem 3.15 ([LPS⁺22, ACO23])**.** *There exist:*

- *a robust algorithm solving Feedback Vertex Set for n-vertex disk graphs in time $2^{\widetilde{\mathcal{O}}(k^{9/10})}$,*
- *and a robust randomized algorithm solving Odd Cycle Transversal for n-vertex disk graphs in time $2^{\widetilde{\mathcal{O}}(k^{19/20})}$.*

The generalization of these algorithms to pseudo-disk graphs will be discussed in Section 7.3.

Moreover, the authors of [ACO23] get even better running times by considering variant of Lemma 3.14, which can be applied to a graph associated to the representation of the graph as disks[1] and the framework of Theorem 3.12.

Theorem 3.16 ([ACO23])**.** *Given a n-vertex disk graph with a representation, TH can be solved in time $2^{\widetilde{\mathcal{O}}(k^{2/3})}n^{\mathcal{O}(1)}$, FVS in time $2^{\widetilde{\mathcal{O}}(k^{7/8})}n^{\mathcal{O}(1)}$ and OCT in time $2^{\widetilde{\mathcal{O}}(k^{15/16})}n^{\mathcal{O}(1)}$.*

[1] This graph is called *arrangement graph*, and will be defined in Chapter 6.

Neighborhood complexity in algorithms

Here, we will consider the following graph property.

Definition 3.17. *If for a graph class $\mathcal{G}$ there is a constant c such that for every $G \in \mathcal{G}$ and every $X \subseteq V(G)$, $|\{N(v) \cap X : v \in V(G)\}| \leq c|X|$, then we say that $\mathcal{G}$ has* **linear neighborhood complexity** *with* **ratio** *c.*

This notion will be of great importance in the coming chapters, and related structural results will be described in more detail in Chapter 4. Therefore, let us now only sketch the role of neighborhood complexity in the problems we consider (typically FEEDBACK VERTEX SET and TRIANGLE HITTING). Given an instance (G, k) we generally start by defining a constant approximation M, where $|M| = O(k)$ and $G \setminus M$ has by definition a simple structure. Then, the objective is to simplify $G \setminus M$ to obtain a simple graph G' with a bound of the form $\mathbf{tw}(G') = o(|M|)$. Typically, to bound the set I of isolated vertices in $G \setminus M$, we can partition $I = \bigcup_{X \subseteq M} I_X$, where $I_X = \{v \in I \mid N(v) \cap M = X\}$. Then, it is generally possible in FVS or TH to only keep a constant number of vertices in each I_X, reducing I to a set I', where $|I'| = \mathcal{O}(|\{N(v) \cap X, X \subseteq M\}|)$. Thus, if the graph has a linear neighborhood complexity with ratio c, we obtain $|I'| = \mathcal{O}(c|M|)$.

Contraction decomposition theorems

A ***Contraction Decomposition Theorem*** is a statement of the form: given a graph G of some graph class, for any p, one can partition $E(G)$ into $E_1, \ldots, E_p$ such that contracting the edges in each E_i in G yields a graph of treewidth at most $f(p)$ for some $f : \mathbb{N} \to \mathbb{N}$. In [BLL+22] such a theorem was proved for unit disk graphs, and was used for proving the following theorem.

Theorem 3.18 ([BLL+22])**.** *There exists a robust algorithm solving ODD CYCLE TRANSVERSAL for n-vertex unit disk graphs in time $2^{\mathcal{O}(\sqrt{k}\log k)} n^{\mathcal{O}(1)}$.*

These results have many other algorithmic applications, but they are not covered in this thesis.

Geometric properties

An alternative approach to bounding the treewidth is to find new properties of geometric graph classes, such as this recent result for outer-string graphs.

Theorem 3.19 ([AOX24])**.** *The treewidth of an n-vertex $K_{t,t}$-free outerstring graph is $\mathcal{O}(t(\log t)\log n)$.*

For comparison, recall that for string graphs we have the worse bound $\mathbf{tw}(G) = \mathcal{O}\left(\sqrt{t(\log t) \cdot n}\right)$ (Corollary 2.13). This implies two algorithms subexponential in the parameter for Vertex Cover and Feedback Vertex Set.[2]

Corollary 3.20 ([AOX24]). *For n-vertex outerstring graphs,*

- *Vertex Cover can be solved in time* $2^{\widetilde{\mathcal{O}}\left(\sqrt{k}\right)} n^{\mathcal{O}(1)}$*,*
- *and Feedback Vertex Set can be solved in time* $n^{\widetilde{\mathcal{O}}\left(\sqrt{k}\right)}$*.*

3.2 Results of interest

3.2.1 The $\mathcal{F}$-Hitting problem with $\mathcal{F}$ finite

A recent major result for the development of subexponential FPT algorithms is the following theorem.

Theorem 3.21 ([LPS+24b]). *Let $\mathcal{F}$ be a finite family of graphs, $\eta, \mu > 0$ and $0 < \rho < 1$, and $\mathcal{G}$ a graph class such that all the induced subgraphs H of a graph $G \in \mathcal{G}$ have a balanced separator of size $\eta\omega(H)^{\mu}|V(H)|^{\rho}$. Then the $\mathcal{F}$-Hitting problem can be solved in subexponential FPT time for graphs in $\mathcal{G}$.*

The result is in fact even stronger, as it is about the weighted variant of the problem, and the running time is more precisely $2^{\mathcal{O}(k^c)} n + \mathcal{O}(m)$ for a graph with n vertices and m edges in $\mathcal{G}$, with $c < 1$ depending on $\mathcal{F}, \eta, \mu$ and ρ.

For many usual geometric graph classes $\mathcal{G}$ like pseudo-disks, this theorem answers by the affirmative to whether there exist subexponential FPT algorithms solving $\mathcal{F}$-Hitting for $\mathcal{G}$ when the family $\mathcal{F}$ of graphs to hit is finite.

However, as $K_{t,t}$ for $t \geq 2$ is a graph with $\omega(K_{t,t}) = 2$ and no balanced separators of size sublinear in $V(K_{t,t})$, the result does not hold for classes containing large bicliques. Such examples are 2-DIR graphs (in fact, we prove in Chapter 5 that there is no subexponential FPT algorithms for Triangle Hitting in 2-DIR graphs) or outerstring graphs, where Vertex Cover (K_2-Hitting) can be solved in subexponential FPT time (Corollary 3.20) but where the existence of subexponential FPT algorithms solving $\mathcal{F}$-Hitting for any fixed finite family $\mathcal{F}$ is open.

Moreover, this result does not aim for optimality for the constant $c < 1$ in the k^c exponent of the running time, and in fact, the constants obtained with this method are close to 1. So, obtaining for a fixed $\mathcal{F}$ algorithms solving $\mathcal{F}$-Hitting

[2] In the original article both algorithms are stated FPT but there was a flaw in the proof, as confirmed to us by the authors.

with $\mathcal{F}$ finite is still a question of interest with the aim of obtaining better running times, or considering graph classes where Theorem 3.21 does not apply.

3.2.2 Win-win strategy using balanced separators

Using the branching method defined in Subsection 3.1.1 and the existence of strongly sublinear separators, many results were obtained, with the following win-win strategy. Either the input graph G contains a large "structure" (like $K_{t,t}$, K_t, a vertex of degree t, for $t = n^\varepsilon$), and in that case we can perform a subexponential branching, or otherwise $\mathbf{tw}(G) = \mathcal{O}(n^{1-f(\varepsilon)})$. We give some examples below.

Theorem 3.22 ([NOP+21])**.** *For a linear forest H, the Feedback Vertex Set problem can be solved for H-induced-free graphs on n vertices in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.*

Theorem 3.23 ([BR19])**.** *3-Coloring, Independent Set, Feedback Vertex Set, and Maximum Induced Matching can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$ for n-vertex string graphs.*

The algorithm for 3-Coloring was later generalized, this time for the weighted homomorphism problem, a problem defined below which can encode many problems.

Definition 3.24. *Let H be a fixed graph (with possible loops[3]), and consider the following computational problem called WHom(H). Given a graph G, a function $w : (V(G) \times V(H)) \cup (E(G) \times E(H)) \to \mathbb{R}$, and an integer k. For a homomorphism $h : G \to H$ the weight of h is defined as $\sum_{x \in V(G) \cup E(G)} w(x, h(x))$. We ask if there exists a homomorphism from G to H whose total weight is at least k.*

If the graph H does not contain a square as a subgraph, the problem can be solved in subexponential time:

Theorem 3.25 ([OR20])**.** *Let H be a fixed graph that does not have two vertices with two common neighbors, then the WHom(H) problem can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$ for n-vertex string graphs.*

An example of problems that can be encoded by WHom(H) is k-Coloring by taking $H = K_k$ and a well-chosen weight function. From the above theorem, one then obtains that there exists a subexponential time algorithm for solving 3-Coloring. This result was in fact already known with the same running time [BR19].

A similar bound is known for P_t-induced-graphs:

Theorem 3.26 ([GOR+19])**.** *Let t fixed and H be a fixed graph which does not have two vertices with two common neighbors, then the WHom(H) problem can be solved in time $2^{\mathcal{O}(\sqrt{n})}$ for P_t-induced-free graphs with n vertices.*

[3] A loop is an edge whose endpoints are the same vertex.

Chapter 4
Neighborhood complexity

Contents

This chapter introduces four ways of describing for a graph how complex the close neighborhood around the vertices can be. Examples of known and new results will be given for those "neighborhood complexities", which will have algorithmic applications presented in the following chapters.

4.1 The r-neighborhood complexity

This first section presents a well-studied parameter known as r-neighborhood.

For a graph G and a vertex $v \in V(G)$, we denote $N_G^r[v]$ the set of vertices at distance at most r from v. Observe that $N_G^1[v] = N_G(v) \cup \{v\}$.

Definition 4.1. *A graph class $\mathcal{G}$ has* **bounded r-neighborhood complexity** *by $f : \mathbb{N} \to \mathbb{N}$ if for all $G \in \mathcal{G}$, every non-empty $A \subseteq V(G)$ and every $r \geq 1$, we have $|\{N_G^r[v] \cap A : v \in A\}| \leq f(r)|A|$.*

The monotone graph classes with bounded r-neighborhood complexity are now perfectly characterized:

Lemma 4.2 ([RVS19])**.** *A monotone graph class has bounded expansion if and only if it has bounded bounded r-neighborhood complexity by some function f.*

However, it remains a field of research to determine how small we can take the function f. An example of progress in this direction is the following result:

Theorem 4.3 ([JR24])**.** *Planar graphs have r-neighborhood complexity bounded by a function f with $f(r) = \mathcal{O}(r^4)$.*

4.2 Linear neighborhood complexity

In this section, we focus on the case of 1-neighborhood complexity. We already mentioned this parameter in Chapter 3 due to its relevance in algorithmic applications. We restate its definition:

Definition 4.4. *If for a graph class $\mathcal{G}$ there is a constant c such that for every $G \in \mathcal{G}$ and every $X \subseteq V(G)$, $|\{N(v) \cap X : v \in V(G)\}| \leq c|X|$, then we say that $\mathcal{G}$ has* **linear neighborhood complexity** *with* **ratio** *c.*

In particular, we will be interested in the relation between the biggest clique or biclique excluded as a subgraph and the neighborhood complexity ratio. For example in disk-graphs, improving from a ratio of $\mathcal{O}(t^6)$ proved in [LPS+22], we have:

Theorem 4.5 ([ACO23])**.** *K_t-free disk-graphs have linear neighborhood complexity with ratio $\mathcal{O}(t^2)$.*

Observe that the ratio is not always polynomial, for example:

Theorem 4.6. *For every c, $K_{t,t}$-free outerstring graphs do not have linear neighborhood complexity with ratio $\mathcal{O}(t^c)$.*

Proof. For $t > 3$, we describe the construction on the right of Figure 4.1. We construct an outerstring graph G as follows: consider a clique of size t obtained as an outerstring graph by taking t strings with one extremity on a circle and then permuting the order of the strings in order to obtain the $t!$ permutations. It is then not hard to see that it is possible to add 2^t disjoint strings with one endpoint on the circle all crossing distinct subsets of the t initial strings. The graph G is then the intersection graph of those $t + 2^t$ strings. Observe that G has only t vertices with degree at least t, and so does not have $K_{t-1,t-1}$ as a

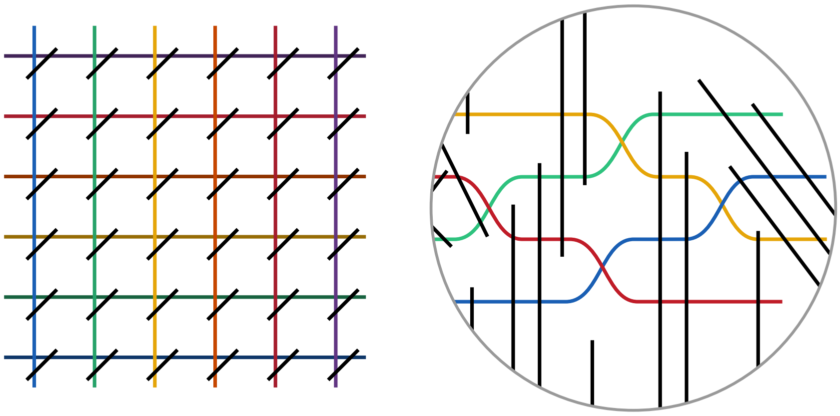

Figure 4.1: **Left:** Example showing that the class of segment graphs does not have linear neighborhood complexity. By taking as set A the $2t$ segments forming a $K_{t,t}$ (in color in the figure), there is a set of t^2 segments (in black) with distinct neighborhoods in A. **Right:** The class of outerstring graphs does not have linear neighborhood complexity, as a set A of t outerstrings crossing each other in a way their top to bottom order goes through many permutations (the strings in color in the figure, with $t = 4$), it is possible to generate 2^t distinct neighborhoods in A (in black).

subgraph. By taking $X \subseteq V(G)$ the vertices associated to the t first strings, we have $|\{N(v) \cap X : v \in V(G)\}| = t + 2^t$. A bound of the form $\mathcal{O}((t-1)^c)$ for some c is then impossible. Note that we can create examples of arbitrarily large size by taking disjoint copies of this construction. □

We now prove two other examples of graph classes with linear neighborhood complexity, that will be useful in the following chapters.

Lemma 4.7. *For every integer $t > 0$, the class of $K_{t,t}$-free d-DIR graphs has linear neighborhood complexity with ratio $\mathcal{O}(dt^3 \log t)$.*

Proof. Let G be a $K_{t,t}$-free d-DIR graph, and fix $M \subseteq V(G)$. Observe that $|\{N(v) \cap M : v \in V(G)\}| \leq |M| + |\{N(v) \cap M : v \in V(G) \setminus M\}|$, and thus we only have to prove that $|\{N(v) \cap M : v \in V(G) \setminus M\}| = \mathcal{O}(|M| \cdot dt^3 \log t)$.

Whenever two vertices of $G - M$ have the same neighborhood in M, we delete one of them. For simplicity, we also delete all vertices with no neighbors in M and call G' the obtained graph. We now focus on bounding the order of G'; observe that this would straightforwardly translate to a bound on the number of neighborhoods in M of vertices of $G - M$.

For a d-DIR representation of G', we consider that one of its directions is the horizontal. Therefore, in the following, we say that a vertex of G' is ***horizontal*** if its segment in the representation is. For convenience, the vertices with 0-length segments are considered as horizontal vertices.

Let M_H denote the horizontal vertices of M and $M_C = M \setminus M_H$. Moreover, we set $A = V(G') \setminus M$ and define in the same way A_H. As the horizontal direction was chosen arbitrarily, a bound of the size of A_H multiplied by d would be a bound to the size of A. Towards this goal we further partition A_H into two subsets: A_{H_1} and A_{H_2} with $A_{H_1} = \{v \in A_H : N(v) \cap M_H \neq \emptyset\}$ and $A_{H_2} = A_H \setminus A_{H_1}$.

Claim 4.8. $|A_{H_1}| = \mathcal{O}(|M| \cdot t^2 \log t)$.

Proof of Claim 4.8. We will bound $|A_{H_1}|$ by using special points of the plane contained in the horizontal segments of M. Let P_H be the set of the endpoints of the M_H segments and the points of intersection of a M_H segment with a M_C segment.

We denote by m_M the number of edges of the graph $G'[M]$. The size of P_H is at most $2|M_H| + m_M$ as intersections between segments correspond to edges in $G'[M]$, whose number is bounded by $\mathcal{O}(|M| \cdot t \log t)$ in Theorem 2.11.

There are at most $|P_H|$ segments of A_{H_1} which do not intersect any point of P_H: those are the ones strictly included in a subsection of the horizontal segments of M split at the P_H points. For the remaining segments, which each intersect at least one point of P_H, there cannot be more than $2t|P_H|$ of them as otherwise by the pigeonhole principle there would be a point of P_H contained in at least $2t$ segments. Then K_{2t} would be a subgraph of G, and so $K_{t,t}$, contradicting our hypothesis. Thus, we get $|A_{H_1}| \leq (2t+1)|P_H| \leq (2t+1)(2|M_H|+m_M) = \mathcal{O}(t^2(\log t)|M|)$. $\square$

We now want to bound the size of A_{H_2}, the horizontal vertices of $G - M$ which do not intersect a horizontal segment of M.

Claim 4.9. $|A_{H_2}| = \mathcal{O}(t^3(\log t)|M|)$.

Proof of Claim 4.9. Let us first remark that we can safely ignore the horizontal vertices of M as they have no neighbors in A_{H_2}. To simplify the proof, we add a perturbation to the segments of the representation of G', keeping the property that this is a d-DIR representation of G' but such that an intersection involves at most 2 different directions, and such that all the endpoints and the intersections between segments of M_C have distinct ordinate. To achieve such perturbation, we extend the length of every segment of the representation of G' by a small amount. Moreover, for each line that supports at least one segment, we move slightly the whole line to a near parallel line (ensuring the associated intersection graph remains the same). Once this perturbation is done, we define a set of special points in the plane, P_C, which are the endpoints of the M_C segments and their intersection points with each other. The perturbation ensures that the P_C points have distinct ordinates.

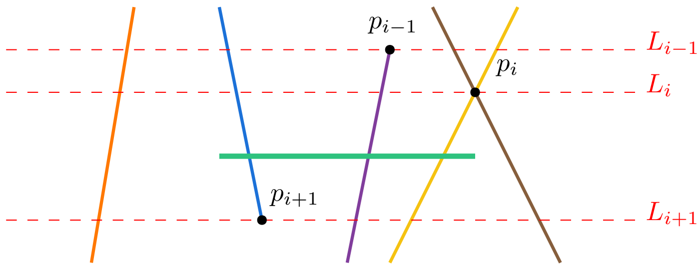

Figure 4.2: Example for the construction of the family $(Z_i)_i$ in the proof of Claim 4.9. The green segment would be part of the set Z_i.

Let us denote $p_1, \dots, p_k$ the points of P_C ordered from top to bottom, and for $1 \leq i \leq k$ we denote L_i the horizontal line containing the point p_i. We now partition A_{H_2} according to their positions relative to the $(L_i)_i$ lines: for $1 \leq i \leq k$ we denote by Z_i the set of segments of A_{H_2} whose neighborhood in M can be achieved by a horizontal segment above L_{i+1} but not by a segment on or above the line L_i. (see Figure 4.2)

Observe that:

- $(Z_i)_i$ is a partition of A_{H_2}, as a segment that can be represented on one line L_i will have a direction, either up or down, where it can be moved by a small amount without changing its neighborhood.
- In the zone between lines L_i and L_{i+1}, there would be no endpoints of segments of M_C, and segments of M_C do not cross each other, which means they will have a constant left to right order. We will denote $S_i = \{s_1, \dots, s_l\}$ those segments enumerated from left to right .
- For a segment of Z_i, its neighborhood is an interval in the sequence $(s_i)_i$.

We now want to prove that there exist two segments in S_i such that every segment in Z_i intersects at least one of them. We differentiate three cases depending on the point p_i:

- If p_i is an intersection of segments of different directions, the perturbation ensures that at most two directions are involved. Therefore there is two segments in S_i, each containing the point p_i and one from each involved direction, such that a segment in Z_i crosses exactly one of those two segments, as otherwise a representation of this segment with this neighborhood in M could be raised above the line L_i.
- If p_i is a top endpoint of a segment s, then each segment z of Z_i would intersect this segment s as otherwise the representation of z below L_i could be raised up above L_i.

- If p_i is a bottom endpoint of a segment s, if this endpoint was at the left of each segment in S then all the segments that could be done below L_i could still be represented with the same neighborhood above L_i and so $Z_i = \emptyset$, the same is true if the endpoint is on the right of each segment of S_i. Therefore, we can assume that the endpoint is between two segments of S. Then a segment of Z_i will necessarily intersect these two segments as otherwise the representation of z below L_i could be raised up above L_i.

Now that we have this property, let s_j be the segment of S_i that is crossed by at least half of Z_i. There are at most t^2 segments that do not cross the segment s_{j-t}, nor s_{j+t} as this is the number of intervals in $[j-t,\dots,j,\dots,j+t]$ that contain j. There are at most $t-1$ segments intersecting s_{j-t} (if it exists), as otherwise, there would be a $K_{t,t}$. Similarly, there are at most $t-1$ segments intersecting s_{j+t}. Therefore, $|Z_i| \leq 2(t^2+2t-2) = \mathcal{O}(t^2)$.

As the $(Z_i)_i$ was a partition of A_{H_2}, and the number of special points $|P_C|$ is at most $2|M_C| + m_M = \mathcal{O}(|M| t \log t)$, we obtain $|A_{H_2}| = \mathcal{O}(|M| t^3 \log t)$. □

To conclude,

$$|\{N(v) \cap M : v \in V(G) \setminus M\}| \leq 1 + d(|A_{H_1}| + |A_{H_2}|) = \mathcal{O}(|M| \cdot dt^3 \log t).$$

□

If we restrict ourselves to the class of *contactsegment* graphs, we do not need to forbid big bicliques in order to have linear neighborhood complexity:

Lemma 4.10. *The class of contact segment graphs has linear neighborhood complexity.*

Proof. Let G be a contact segment graph, and fix $M \subseteq V(G)$. As in the previous lemma, let us only prove that $|\{N(v) \cap M : v \in V(G) \setminus M\}| = \mathcal{O}(|M|)$. Whenever two vertices of $G - M$ have the same neighborhood in M, we delete one of them. We also delete all vertices whose neighborhood in M can be obtained with a zero-length segment: this kind of neighborhood either contains at most one vertex of M (there are $|M| + 1$ such neighborhoods) or is a set of segments in contact on the same point. This point has to be an endpoint of at least one segment of M, so there are at most $2|M|$ such neighborhoods. We call G' the graph obtained after deleting those vertices. We split the set $A = V(G') \setminus M$ in two parts such that A_1 is the set of segments of A whose interior is in contact with an endpoint of a segment of M, and $A_2 = A \setminus A_1$. Observe that an endpoint of a segment can be in contact with only one interior of segments without creating a crossing, so we have $|A_1| \leq 2|M|$. Now it remains to bound A_2.

Claim 4.11. $|A_2| \leq 24|M|$.

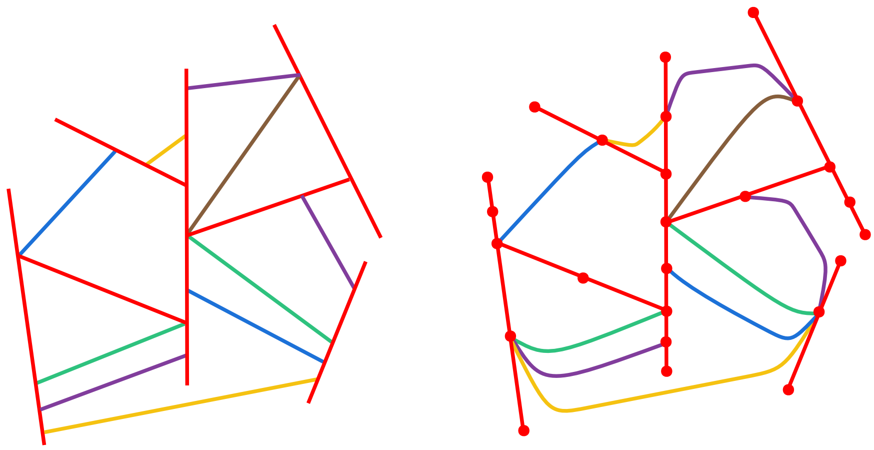

Figure 4.3: Example of the construction used in the proof of Claim 4.11. The contact segment representation of G' is on the left, with the segments of M in red. On the right is the constructed planar graph H, with the edges not in red being the representation of the segments of A_2.

Proof. Let us define an embedding in the plane of a planar graph H such that $|H| \leq 8|M|$ and $|E(H)| \geq |A_2|$. The embedding is defined from a contact segment representation of $G' - A_2$, as depicted in Figure 4.3. Firstly we add one vertex on every position of the endpoints of the segments of M, and denote V_1 this set of vertices, where $|V_1| \leq 2|M|$. Then, we add edges between vertices of V_1 as follows: for each segment v of M, we add a path starting at an endpoint of v and then (following their order along v) all the endpoints of segments of M on segment v, until reaching the other endpoint of v. We draw the edges of this path following the segment representation of v. We denote this set of edges by E_1. Observe that at this step the graph is planar, implying that $|E_1| \leq 6|M|$ by the Euler's formula.

The next step of the construction is to subdivide every edge $e \in E_1$ by adding a vertex v_e on the center e. We denote V_2 this set of vertices. Informally, we associate a vertex of H with every section of segments between endpoints of M. Observe that $|V_1| + |V_2| \leq 8|M|$.

Let us now associate to each $v \in A_2$ a new edge $e(v)$ in H, and also explain how we can draw these new edges without crossings. Let $v \in A_2$. Observe that as v is not in contact with M in its interior, its neighborhood in M is entirely decided by its two endpoints, and both of them are part of a segment of M as otherwise the neighborhood in M of v could be achieved with a zero-length segment, and so $v \notin V(G')$. Let s be the segment representing v and $\{p_1(s), p_2(s)\}$ be the endpoints of s. Let slightly shortening s to obtain $\tilde{s}$, and for $i \in \{1, 2\}$ let $p_i(\tilde{s})$ be the endpoint of $\tilde{s}$ corresponding to $p_i(s)$. We now associate a vertex $v_i \in V(H)$ to each $p_i(s)$ and we will define $e(v) = \{v_1, v_2\}$. Moreover, to draw $e(v)$, we will also define three segments $s_1(v), s_2(v)$ and $s_3(v)$, and draw $e(v)$ as $s_1(v) \cup s_2(v) \cup s_3(v)$. Firstly, we define $s_3(v) = \tilde{s}$. Then, let us distinguish two cases:

1. If $p_i(s)$ is also an endpoint p of a segment of M, then v_i is the vertex of H corresponding to the endpoint of this segment in M, and $s_i(v) = \{p, p_i(\tilde{s})\}$
2. Otherwise, $p_i(s)$ is in the interior of exactly one segment s of M, and more precisely inside an edge $e \in E_1$. Then, we define $v_i = v_e$. Let $p(v_e)$ be the point associated to v_e. We define $s_i(v) = \{p(v_e), p_i(\tilde{s})\}$.

This concludes the definition of H. As required, we get that H is planar, $|H| \leq 8|M|$, $|E(H)| \geq |A_2|$. This last property gives $|E(H)| \leq 3|H|$ by the Euler's formula, and so $|A_2| \leq 24|M|$, which is the wanted result. □

In total, $|\{N(v) \cap M : v \in V(G) \setminus M\}| \leq 1 + |M| + |A_1| + |A_2| = \mathcal{O}(|M|)$. □

4.3 Number of small neighborhoods

This section introduces another point of view for describing the diversity of the neighborhoods of the vertices of a graph G relative to a subset A. It can be seen as the neighborhood complexity where only the neighborhoods in A of size at most some integer k are considered.

Theorem 4.12 ([ADEP21, Kes20]). *Let $\mathcal{F}$ and $\mathcal{P}$ be two finite systems of pseudo-disks in the plane. Consider the hypergraph $H(\mathcal{P}, \mathcal{F})$ whose vertices are the pseudo-disks in $\mathcal{P}$ and the edges are all the sets $\{D \in \mathcal{P} : D \cap S \neq \emptyset\}$, where S is a pseudo-disk in $\mathcal{F}$. Then the number of edges of cardinality at most $k \geq 1$ in $H(\mathcal{P}, \mathcal{F})$ is $\mathcal{O}(|\mathcal{P}|k^3)$.*

4.4 Subgraph-neighborhood

In this section, instead of considering the neighborhoods of the vertices relative to a set A, we consider the neighborhoods of disjoint connected subgraphs H relative to a set A. This new definition is used in Section 7.1, where a new framework is developed for providing subexponential FPT algorithms for the FVS problem on some subclasses of string graphs.

Definition 4.13. *We say that a graph class $\mathcal{G}$ has* **bounded subgraph-neighborhood complexity** *(with parameters α, f_1, f_2) if there exist a value α and two polynomial functions f_1, f_2 such that the following conditions hold. For every r, every $K_{r,r}$-free graph $G \in \mathcal{G}$, every set $A \subseteq V(G)$ and every family $\mathcal{H}$ of disjoint non-adjacent*[1] *subgraphs of $G - A$:*

[1]Two vertex subsets are ***non-adjacent*** in a graph if there is no edge from one to the other.

1. $|\{N_A(H) : H \in \mathcal{H}\}| \leq f_1(r)|A|^\alpha$*, where* $N_A(H)$ *denotes the neighbors of the vertices of* H *in* A*, and*

2. $|\{N_A(H) : H \in \mathcal{H}\}| \leq f_2(r, p, m)|A|$*, where* p *and* m *denote the maximum over all* $H \in \mathcal{H}$ *of* $|N_A(H)|$ *and* $|H|$ *respectively.*

In fact, in Section 7.1 we will use the definition with the restriction that the considered connected subgraphs are trees. In this case, we hence say that the graph class has ***bounded subgraph-neighborhood complexity***. However, in order to be as general as possible, this section deals with bounded subgraph-neighborhood complexity, which is a stronger result than having bounded tree-neighborhood complexity.

4.4.1 Subgraph-neighborhood of pseudo-disk graphs

Lemma 4.14. *There exists a constant* c *such that the class of pseudo-disk graphs has bounded subgraph-neighborhood complexity with parameters* $\alpha = 4$*,* $f_1(r) = c$ *and* $f_2(r, p, m) = cp^3$*.*

Proof. Let $r \geq 2$, G be a $K_{r,r}$-free pseudo-disk graph, $(\mathcal{D}_v)_{v \in V(G)}$ be a representation of G as pseudo-disks, $A \subseteq V(G)$, and $\mathcal{H}$ a family of disjoint non-adjacent connected subgraphs of $G - A$. We would like to apply Theorem 4.12 on the set A and the family of subsets of the plane obtained by taking for each connected subgraph $H \in \mathcal{H}$ the union $\mathcal{D}_H = \cup_{v \in H} \mathcal{D}_v$. However, this union may not be homeomorphic to a disk. It is easy to resolve this issue by instead considering a subset $\mathcal{D}'_H$ of $\mathcal{D}_H$ intersecting the same set of pseudo-disks in the family $(\mathcal{D}_u)_{u \in V(G-A)}$ where small cuts are made in order to have $\mathbb{R}^2 \setminus \mathcal{D}'_H$ connected, and more precisely having $\mathcal{D}'_H$ homeomorphic to a disk. Recalling that $\mathcal{H}$ is a set of disjoint non-adjacent connected subsets and that $\mathcal{D}'_H \subseteq \mathcal{D}_H$ we have that the shapes $(\mathcal{D}'_H)_{H \in \mathcal{H}}$ are disjoint, and so trivially pseudo-disks. Now applying Theorem 4.12 with the families $(\mathcal{D}_v)v \in A$ and $(\mathcal{D}'_H)_{H \in \mathcal{H}}$ directly gives that there exists a constant c_2 such that if for all $H \in \mathcal{H}$ we have $d_A(H) \leq p$ then $|\{N_A(H) : H \in \mathcal{H}\}| \leq c_2 p^3 |A|$ as wanted for the second bound. Observing that we can take $p = |A|$ as an upper bound for $d_A(H)$ with $H \in \mathcal{H}$ gives the first wanted bound. □

4.4.2 Subgraph-neighborhood of s-strings graphs

Lemma 4.15. *There exist constants* c_1, c_2 *such that the class of* s*-string graphs has bounded subgraph-neighborhood complexity with parameter* $\alpha = 4, f_1(r) = c_1 s^4 r \log r$ *and* $f_2(r, p, m) = c_2(sr \log r)^4 (p + m)^3$*.*

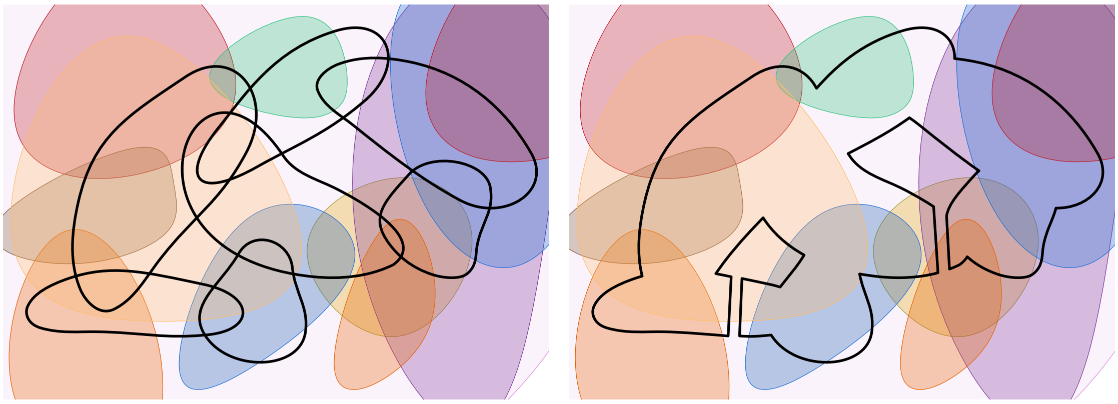

Figure 4.4: Transformation of a union of pseudo-disks to a region homeomorphic to a disk, while keeping the same neighborhood in A. The filled pseudo-disks belong to A.

Proof. The proof is very similar to the proof of Lemma 4.14 with some additional tweaks. Let G be a $K_{r,r}$-free s-string graph for some $r \geq 2$ and $(\mathcal{S}_v)_{v\in V(G)}$ be an s-string representation of G. For $H \in \mathcal{H}$, again we consider the region $\mathcal{D}_H = \bigcup_{v\in V(H)} \mathcal{S}_v$. As we did for pseudo-disks, we modify $\mathcal{D}_H$ by cutting small sections where no intersections take place (not even with a string of A) and obtain a new set $\mathcal{D}'_H \subseteq \mathcal{D}_H$ that is connected and such that there is exactly one arc-connected region in $\mathbb{R}^2 \setminus \mathcal{D}'_H$ and which intersects the same strings in A as $\mathcal{D}_H$ (see Figure 4.5). Moreover, we add a small thickness to $\mathcal{D}'_H$ while preserving the wanted property about the number of regions in $\mathbb{R}^2 \setminus \mathcal{D}'_H$ in order to obtain a geometric shape homeomorphic to a disk. Observe that as there are no edges between vertices of distinct members of $\mathcal{H}$, by taking the thickness small enough the connected sets of the form $\mathcal{D}'_H$ (for $H \in \mathcal{H}$) are disjoint and so they trivially form a system of pseudo-disks.

Now we would like to construct another system of pseudo-disks representing the vertices of A. We do this by cutting the strings in $(\mathcal{S}_u)_{u\in A}$ whenever two of them cross each other. Observe that by Theorem 2.11 there exists a constant c such that by denoting $d_r = cr\log r$ we have at most $d_r|A|$ intersecting pairs, and remember that each pair intersects each other at most s times as we consider a s-string representation. Therefore, we have at most $sd_r|A|$ cuts which gives an upper bound of $(sd_r + 1)|A|$ for the number of sections of strings between intersection points. We denote C as the set of such sections. Again we define a family $(\mathcal{R}_v)_{v\in C}$ by considering the sections (which are disjoint by construction) and giving them some thickness while preventing the sections from intersecting each other. This is clearly a system of pseudo-disks as its elements are disjoint. By

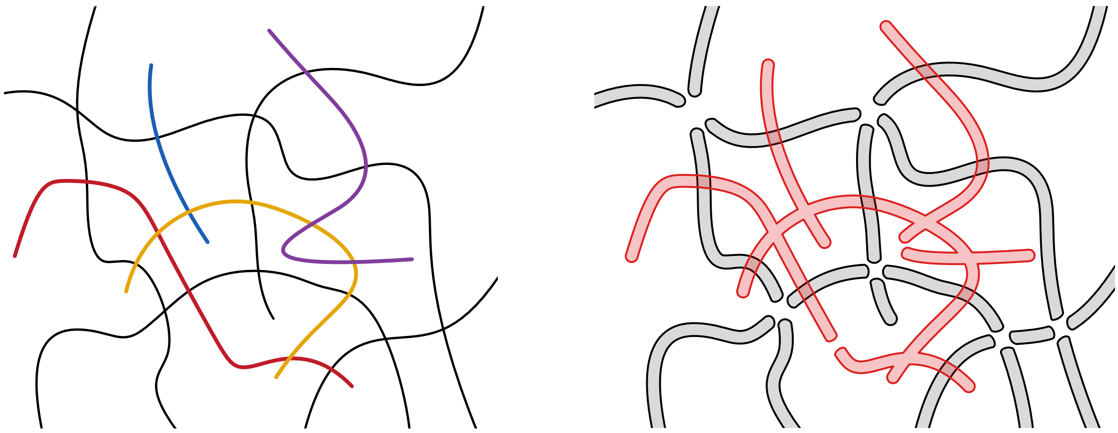

Figure 4.5: Transformation of a union of s-strings to a region homeomorphic to a disk, while keeping the same neighborhood in A. The strings of A are depicted in black.

Theorem 4.12, the number of distinct sets $N_C(\mathcal{D}'_H) = \{v \in C : \mathcal{D}'_H \text{ intersects } \mathcal{R}_v\}$, for $H \in \mathcal{H}$, is $\mathcal{O}(x^3|C|)$ where x is the maximum, over every $H \in \mathcal{H}$, of the number of pseudo-disks in $(\mathcal{R}_v)_{v \in C}$ crossed by D'_H. This upper bound gives the second result of this lemma by observing that if a pair of pseudo-disks $\mathcal{D}'_{H_1}, \mathcal{D}'_{H_2}$ intersect exactly the same sets of $\mathcal{R}_v$, then $N_A(H_1) = N_A(H_2)$. We obtain the first result of the lemma by using the trivial bound $x \leq |C|$. $\square$

Chapter 5
Lower bounds

Contents

This chapter presents lower bounds for the complexity of solving vertex-deletion problems in geometric intersection graph classes, or other classical problems.

In particular, new lower bounds for K_r-HITTING, OCT, and FVS in subclasses of segment graphs are presented.

Notice that for the problems we consider where the input is a graph G and an integer $k \leq n$ (where $n = |V(G)|$), we only present non-parameterized lower bound stating that, under a complexity assumption, there is no algorithm running in time $2^{o(f(n))}$. These lower bound are stronger than their counterpart in the parameterized world, as given the functions f considered, it also implies that there is no $2^{o(f(k))}n^{\mathcal{O}(1)}$ algorithm as well.

5.1 Known lower bounds

5.1.1 For the class of all graphs

For most of the vertex deletion problems, an algorithm with running time $2^{o(n)}$ for n-vertices graphs would be in contradiction with ETH, as shown by the next theorem. Before stating the result we need an additional definition: A graph property Π is ***nontrivial*** if there are infinitely many graphs satisfying Π, and infinitely many graphs not satisfying Π as well.

We can now state the lower bound, which comes from the polynomial reduction presented in [LY80].

Theorem 5.1 ([LY80]). *For a fixed hereditary nontrivial graph property Π, the vertex deletion problem associated with Π cannot be solved in time $2^{o(n)}$ unless ETH fails.*

Remark 5.2. *In the case where the property Π is trivial, it is easy to see that the vertex deletion problem can be solved in polynomial time.*

Corollary 5.3. *Feedback Vertex Set, Odd Cycle Transversal, and more generally the $\mathcal{F}$-Hitting problem with $\mathcal{F}$ nonempty and such that there are infinitely many $\mathcal{F}$-free graphs cannot be solved in time $2^{o(n)}$ for the class of all graphs, unless ETH fails.*

5.1.2 H-induced-free graph class

Before stating lower bounds for the complexity of problems restricted to geometric graphs, we give examples of lower bounds for graph classes defined with other restrictions than geometric ones.

Theorem 5.4 ([NOP+21]). *If H is not a linear forest, Feedback Vertex Set cannot be solved for H-induced-free graphs with n vertices in time $2^{o(n)}$.*

For the case of H-induced-free graphs with H a linear forest, we have seen in Theorem 3.22 that FVS admits subexponential non-parameterized algorithms.

5.1.3 Fat objects

Theorem 5.5 ([FLS18]). *For $d \geq 3$, there is no algorithm solving Feedback Vertex Set in unit d-dimensional ball in subexponential FPT time, unless ETH fails.*

Theorem 5.6 ([dBBKB+20]). *For every $d \geq 2$, the following problems cannot be solved in time $2^{o(n^{1-1/d})}$ under ETH:*

- *Independent Set for d-dimensional unit ball graphs,*
- *Feedback Vertex Set and r-Dominating Set for induced subgraphs of a d-dimensional grid.*

Theorem 5.7 ([Mar05, Mar06])**.** *Independent Set and Dominating Set for unit disk graphs and unit square graphs are $W[1]$-complete.*

5.1.4 Thin objects

Theorem 5.8. *The following problems cannot be solved in time $2^{o(n)}$ under ETH:*

- *Dominating Set for segment graphs [Mar06],*
- *Independent Dominating Set, Connected Dominating Set in segment graphs [BR19].*

The problem Independent Set is $W[1]$-hard for segments, $W[1]$-complete for unit segment graphs, but the problem is FPT algorithms in d-DIR graphs [Mar06, KK06]. However, for d-DIR graphs the known FPT complexities are more than exponential in k.

For the WHom(H) in string graphs, the H for which Theorem 3.25 proves the existence of subexponential algorithms are in fact exactly the H where such algorithms exists, assuming ETH.

Theorem 5.9 ([OR20])**.** *Let H be a fixed graph having two vertices with two common neighbors. Then the WHom(H) problem cannot be solved in time $2^{o(n)}$ for segment graphs, unless the ETH fails.*

This lower bound completes a similar dichotomy of subexponential algorithms solving WHom(H) for P_t-induced-free graphs as well. However, now the sublinear exponent is $\sqrt{n}$ as in Theorem 3.26 [GOR+19, OR20]. Note that for the graph H where a subexponential algorithm solving WHom(H) exists for both P_t-induced-free and string graphs, the best known running time can be very different. For example, Independent Set and 3-Coloring cannot be solved in $2^{o(\sqrt{n})}$ for segment graphs unless ETH fails, but can be solved in time $n^{\mathcal{O}_t(\log^2(n))}$ for P_t-induced-free graphs [PPR21, GL20].

5.2 Lower bounds for TH, OCT and FVS

In this section, we show that assuming the ETH, there is no algorithm solving TH and OCT in time $2^{o(n)}$ on n-vertex 2-DIR graphs and more generally in time

$2^{o(\sqrt{\Delta n})}$ in 2-DIR graphs with maximum degree Δ. We note that the result for OCT was already proved in segment graphs in [OR20] as a consequence of Theorem 5.9. In our second negative result, we prove that assuming the ETH, the problems TH, OCT, and FVS cannot be solved in time $2^{o(\sqrt{n})}$ on n-vertex $K_{2,2}$-free contact 2-DIR graphs. Notice that our $2^{o(\sqrt{n})}$ lower-bounds match those known for the same problems in planar graphs [CJ03].

Let us first start by stating our first result on TH and OCT.

Theorem 5.10. *Under the ETH, TH and OCT cannot be solved in time $2^{o(n)}$ on n-vertex 2-DIR graphs.*

Before proceeding to the proof, we need to introduce some definitions for the gadgets used in our reductions.

Definition 5.11. *For $k \geq 2$, a k-**polygon** P is a 2-DIR graph composed of $4k$ axis-parallel segments in the plane such that $V(P) = H \cup V \cup C$ with:*

1. *H is a set of k disjoint horizontal segments of non-zero-length;*
2. *V is a set of k disjoint vertical segments of non-zero-length;*
3. *every segment of H intersects exactly two segments of V, and vice-versa;*
4. *C consists of zero-length segments located at each intersection point between a segment of H and a segment of V; and*
5. *the intersection graph of the segments in P is connected.*

Notice that in the definition above, $|C| = 2k$. See Figure 5.1 for a depiction of a 3-polygon.

Lemma 5.12. *Let P be a k-polygon. Then P does not have any triangle hitting set of size $k-1$, and has exactly two triangle hitting sets of size k: the non-zero horizontal segments and the non-zero vertical segments.*

Proof. Let P be a k-polygon, and let $\{H, V, C\}$ be the partition of $V(P)$ named as in the definition of k-polygon. Notice that every vertex of C together with its two neighbors (one vertical, one horizontal, by definition) forms a triangle, and that every triangle P has this form. Therefore, P contains $2k$ triangles. As every segment is part of at most two triangles, any triangle hitting set has size at least k, which shows the first part of the statement. Suppose now that X is a triangle hitting set of size k. In order to intersect the $2k$ triangles of P, each of the k segments in X has to take part in two triangles, so $X \cap C = \emptyset$, and no triangle can be hit twice, so S is an independent set. The induced subgraph over $V \cup H$ is a cycle of size $2k$, whose independent sets of size k are H and V. Thus, $X = V$ or $X = H$, as claimed. □

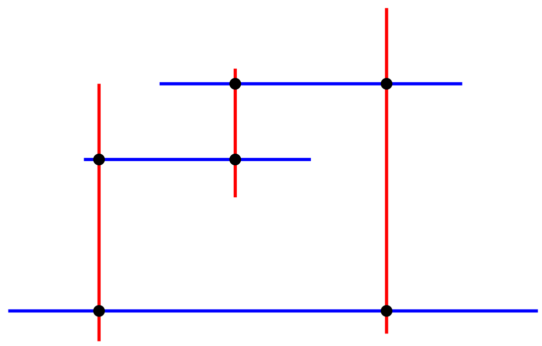

Figure 5.1: An example of a 3-polygon, where the segments in H are blue, the ones in V are red, and the length 0 segments in C are represented by black dots.

Proof or Theorem 5.10. The proof is a reduction from 3-SAT.

Let φ be a 3-SAT instance with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Without loss of generality, we may assume that each variable appears in some clause and that φ has no clause with only one literal (otherwise it could be easily simplified). For our reduction we also want to avoid clauses with 3 literals all positive or all negative. To do so, for any clause of the form $x_a \vee x_b \vee x_c$ we define an additional variable y_i and we replace the clause with the equivalent clauses $x_a \vee x_b \vee \overline{y_i}$ and $x_c \vee y_i$, and similarly for clauses containing only negative literals. Notice that after performing these operations the number of clauses and variables increased by $\mathcal{O}(m)$.

Let us now construct a 2-DIR graph G from the formula φ. In this graph, each variable x_i is represented by a k_i-polygon with k_i to be specified later. (We only describe the non-zero segments, as the position of the zero-length segments is uniquely determined by those, by definition.) To every clause C, we associate a point of the plane z_C. We construct the variable polygons such that, for every clause C and variable x, each of the following is satisfied:

1. if the literal x (respectively $\bar{x}$) appears in C, then some vertical (respectively horizontal) segment of the polygon of the variable x ends at z_C;

2. if C contains only two literals, then a new vertex with zero-length segment is added at position z_C;

3. if x does not appear in C, then z_C does not belongs to any segment of the polygon of x; and

4. if two horizontal (respectively vertical) segments intersect, their intersection consists in a unique point of the form $z_{C'}$ for some clause C'.

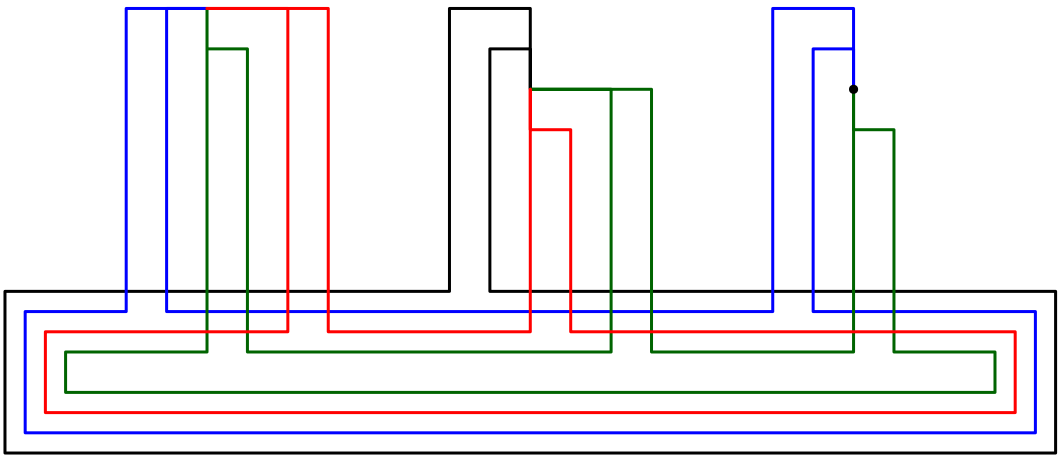

Figure 5.2: The construction for the formula $(\overline{x_2} \vee x_4 \vee \overline{x_3}) \wedge (x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee x_4)$. The zero-length segments at each corner of the k-polygons are not represented, while that added for the clause with two variables is depicted with a black dot.

Figure 5.3: The two remaining possibilities for the clause gadget. The length 0 segments at each corner of the k-polygons are not represented. The black dots represent the length 0 segments we add for clauses with 2 literals.

Also, we want the number of segments in each polygon and its number of intersections with each other polygon to be linearly bounded from above by the number of clauses the corresponding variable appears in. Such a configuration can for instance be obtained by initially drawing the polygons of the variables as concentric rectangles and then, for every clause C, picking a point z_C outside of the outermost triangle and connecting corresponding the polygons to it. See Figure 5.2 for an illustrative example and Figure 5.3 for a depiction of the connection to the z_C's.

For $1 \leq i \leq n$ we define k_i such that the variable gadget for the variable x_i constructed as defined above is a k_i-polygon. Let $k = \sum_{i=1}^{n} k_i$. We define a ***good triangle hitting set*** as a triangle hitting set of size k.

Claim 5.13. *If a good triangle hitting set exists, it is a minimum triangle hitting set,* *and a minimum odd cycle transversal set. Moreover, it only contains non-zero-length segments and all the segments from a polygon it contains have the same orientation.*

Proof. Let S be a good triangle hitting set of G. By Lemma 5.12, for each variable x_i, its corresponding k_i-polygon needs k_i segments in order to hit all its triangles. As $k = \sum_{i=1}^{n} k_i$ and no segment is part of two distinct polygons, we can conclude that exactly k_i segments of the considered polygon are in S, and S is a minimum triangle hitting set. Again by Lemma 5.12 we get that those k_i segments have non-zero-length and either consist of all vertical segments of the polygon, or of all its horizontal segments. Moreover, this implies that S does not contain the zero-length segments added for clauses with two literals. It remains to prove that S is a minimal odd cycle transversal set. Minimality is obtained by the minimality of S as a triangle hitting set and observing that an odd cycle transversal is a triangle hitting set. Let us consider a cycle in G with odd size and show that S contains at least one segment of this cycle. To do so it is enough to prove the cycle necessarily contains 2 segments of the same corner, as in this case, it would contain the two non-zero segments of this corner, one of which necessarily belongs to S. By contradiction suppose that no pair of segments in the cycle intersect in a corner. Then the intersection between two adjacent segments of the cycle is done in their interior (i.e., not at their endpoints) and so they do not have the same orientation: one is horizontal, the other is vertical. Hence the segments of the cycle are alternatively horizontal and vertical and the cycle has even size, a contradiction. □

Claim 5.14. *G has a good triangle hitting set if and only if φ is satisfiable.*

Proof. Direction "⇒". Let S be a good triangle hitting set of G. We define a truth assignment for the variables by setting the variable x_i at false if the segments of the corresponding polygon being in S are the horizontal segments, and true if they are the vertical ones. For a clause C of φ, let Δ_C denote the set of the 3 vertices containing z_C (which trivially induce a triangle). Because S is a hitting set and by Claim 5.13, we have $S \cap \Delta_C \neq \emptyset$. Let x_i be a variable such that its k_i-polygon contains a segment in $S \cap \Delta_C$. Suppose this segment is vertical, this means that the variable x_i appears in C in a positive literal, and that in the constructed assignment x_i is true, and the clause C is satisfied. We can obtain the same conclusion in the case of a vertical segment, so each clause is satisfied with the defined truth assignment, and so φ is satisfiable.

Direction "⇐". For an assignment of the variables x_i such that φ is satisfied, we construct a set S of size k with the same encoding as before: S contains the vertical segments of the k_i-polygon for the variables x_i assigned to true, and the horizontal ones otherwise. Let us now show that S is a hitting set.

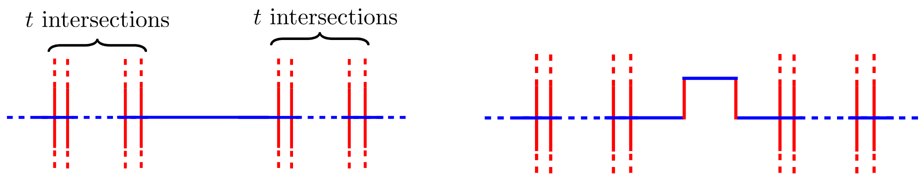

Figure 5.4: The gadget used in the proof of Theorem 5.16. A horizontal segment of a polygon before and after the addition of the crenellation in order to bound the number of crossings by $t+4$.

All the triangles contained in the polygons encoding the variables are hit, as they always contain a horizontal segment and a vertical segment of the same polygon. The remaining triangles are the one formed by each clause C on the point z_C. As φ is satisfied by the considered truth assignment, there is a literal which evaluate to true. Let x_i be the variable contained in this literal. If the literal is positive, then x_i was assigned to true, so all the horizontal segments of its polygon are in S; and the segment in contact with z_C is horizontal. Therefore, the triangle at the point z_C is hit by S. The case of a negative literal has the same conclusion. Thus, all triangles of G are hit by S. □

Claim 5.15. $|G| = \mathcal{O}(m)$.

Proof. As argued above for every variable, the number of segments of its polygon is linearly bounded by the number of clauses it appears in. Each clause contains at most 3 variables and there are at most m extra vertices (for clauses of size 2) so the total number of segments is $\mathcal{O}(m)$.

□

We described the construction of a graph G and integer k from φ such that (G,k) is a positive instance of TH or OCT if and only if φ is satisfiable. Besides, the construction can clearly be done in time polynomial in $n+m$ and the graph has $n' = \mathcal{O}(m)$ vertices. Therefore any algorithm solving TH or OCT in time $2^{o(n')}$ for input graphs of size n' could be used to solve 3-SAT in time $2^{o(m)}$ on formulae with m clauses, which would refute the ETH. □

We observe that the instances constructed as in Figure 5.2 contain large complete bicliques. One way to forbid such bicliques is to restrict the number of crossings in our constructions. We do so by two different approaches. The lower bounds we obtain are weaker in terms of value, however, they apply more generally as they hold for more restricted graph classes. The bound in the following result provides an interpolation between graphs of bounded degree and the general case (obtained by taking $d = n$) where we get the same bound as in Theorem 5.10.

Theorem 5.16. *Under ETH, the problems TH and OCT cannot be solved in time $2^{o\left(\sqrt{dn}\right)}$ on 2-DIR graphs with maximum degree $d \geq 6$.*

Proof. In this proof, we construct the same 2-DIR graph G from the proof of Theorem 5.10, but adapt it so that a side of a polygon is not crossed by more than t segments. We do so by creating a graph G' where crenelations, as depicted in Figure 5.4, are added on the top segments of the base rectangle of each variable gadget, as they are the only segments with unbounded degree in the construction. Observe that after the modification of a segment, each newly created horizontal segment has degree at most $t+4$, as a segment outside of a clause gadget intersects 2 length 0 segments. Taking $t = d - 4$ ensures that the degree of the created segments is at most d (a crenelation increases the degree by 2 on each side), and the degree of the unmodified segments is at most $6 \leq d$, so the maximum degree of G is at most d.

Claim 5.17. $|G'| = \mathcal{O}\left(\frac{|G|^2}{d}\right)$.

Proof. In our construction of G', for a segment intersected by c segments, we add $\mathcal{O}(\frac{c}{d})$ crenelations, each containing a constant number of segments. We have $c \leq |G|$ so the number of additional segments is $\mathcal{O}\left(\frac{|G|^2}{d}\right)$. □

Because our construction still satisfies the properties of Theorem 5.10, Claim 5.14 holds. Therefore any algorithm solving TH or OCT in time $2^{o(\sqrt{dn'})}$ for input graphs of size n' and maximum degree d could be used to solve 3-SAT on formulae with m clauses and in time $2^{o(m)}$, which would refute the ETH. □

We now provide in the next theorem a construction that is $K_{2,2}$-free, with maximum degree 6, where segments do not cross (i.e., that is a contact 2-DIR graph) and that applies also to FVS, to the price of a weaker lower bound.

Theorem 5.18. *Under ETH, the problems TH, OCT, and FVS cannot be solved in time $2^{o(\sqrt{n})}$ on n-vertex $K_{2,2}$-free contact 2-DIR graphs with maximum degree 6.*

Before starting the proof of Theorem 5.18, we need some intermediate results. The ***incidence graph*** of a 3-SAT formula is the (bipartite) graph whose vertices are clauses and variables and where edges connect variables to the clauses they appear in. The restriction of the 3-SAT problem to formulae with a planar incidence graph is called Planar-3-SAT.

Theorem 5.19 ([Lic82]). *There is no algorithm that solves Planar-3-SAT on a formula with n variables and m clauses in time $2^{o(\sqrt{n+m})}$, unless the ETH fails.*

Lemma 5.20. *There is an algorithm that, given a planar bipartite graph G, returns in polynomial time a representation of G as a contact graph of rectangles, where two intersecting rectangles intersect on a non-zero-length segment.*

Proof. Let G be a planar bipartite graph. In linear time it can be represented as a contact 2-DIR graph according to [CKU98], with the contact occurring only between segments of different directions. We will now transform this representation to obtain a representation of G as a contact graph of rectangles. To do this, we first prevent the contact of endpoints of two segments, by slightly extending one if such a contact point exists. Once this transformation is done, we then shorten all the segments at both sides by a small ϵ, and thicken them by the same amount. The segments are now interior disjoint rectangles, and two segments that were in contact are now two rectangles sharing an ϵ-length segment on their border. □

In order to avoid all-positive or all-negative clauses as in the proof of Theorem 5.10, we show below that those (if any) can be replaced without destroying planarity.

Lemma 5.21. *There is an algorithm that, given a planar 3-SAT formula with n variables and m clauses, returns in time polynomial in $n + m$ an equivalent planar 3-SAT formula with $n + m$ variables and $2m$ clauses where in addition no clause contains 3 positive literals or 3 negative literals.*

Proof. Let ϕ be the input formula with n variables and m clauses and let G be its planar incidence graph. For a variable or clause z of φ, we denote by $v(z)$ the corresponding vertex in G. If there is a clause C of the form $x_a \vee x_b \vee x_c$ then we replace it by the two clauses $C_1 = x_a \vee x_b \vee \bar{y}$ and $C_2 = y \vee x_c$, where y is a fresh variable. Clearly, the obtained formula φ' is equivalent and has $n + 1$ variables and $m + 1$ clauses. Also, the incidence graph G' of φ' can be obtained from G by renaming $v(C)$ into $v(C_1)$ (for the clause C_1) and replacing the edge $v(x_c)v(C_1)$ by the path $v(C_1)v(y)v(C_2)v(x_c)$, where $v(C_2)$ and $v(y)$ are new vertices representing C_2 and y. As a subdivision of a planar graph, G' is planar. A symmetric replacement can be done for clauses where all literals are negative. Each replacement decreases the number of all-positive or all-negative clauses so after m replacement steps at most the obtained formula is free of such clauses (and equivalent to the initial one). Finding a clause to replace and doing so can be performed in polynomial time, hence so does the replacement of all all-positive and all-negative clauses, as claimed. □

Proof of Theorem 5.18. In this proof we adapt our construction from Theorem 5.10 so that the polygons intersect each other only at points of the form z_C, for C a clause. toward this goal we consider an instance φ of the problem PLANAR-3-SAT;

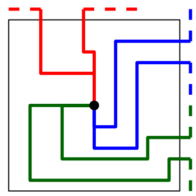

Figure 5.5: An example of extensions of three polygons toward a point Z_C in order to encode a clause as described in the proof of Theorem 5.18.

by Lemma 5.21 we may assume that φ has no all-positive or all-negative clause; also we can easily simplify clauses of size one.

The incidence graph G_φ of φ is a bipartite planar graph, so using Lemma 5.20 it can be represented as a contact graph of rectangles. Let us now describe how this representation is used to construct a 2-DIR instance of the cycle-hitting problems we consider. Initially, for each variable we create a 2-polygon that follows the associated rectangle (in the contact representation) and for each clause C we choose a point z_C of the plane located at the center of its rectangle. For each clause, we want to add a few sides to the polygons of its (2 or 3) variables so that these polygons intersect in z_C only. As they do not intersect elsewhere, this ensures that the obtained graph is a contact 2-DIR. We describe the construction for clauses with 3 variables; that for two variables is a simpler version of it. Recall that for a clause C, the rectangles of its variables are in contact with the rectangle of C and do not intersect mutually. For each variable in C, we add a non-zero-length segment with one endpoint being z_c, vertical if the literal containing the variable is positive, horizontal otherwise. As the clause does not contain all-positive or all-negative literals, it is always possible to do so with the three segments intersecting in z_C only. The choices of the sides of z_c on which we put the segments are made so that the circular order of the segments around z_C is the same as the circular order of the rectangles of the variables around that of C. We claim that extending the polygons of the variables in order to connect each with the newly created segment without crossings can be done by adding a constant number of segments to each, as depicted in Figure 5.5. In order to finish the construction of the graph G, for each non-zero-length segment, which is then a side of one of the polygons, we add in the portion of the segment between the two intersections with the two adjacent sides of the concerned polygon a ***crenellation***, which is the simple gadget depicted in Figure 5.6. We make this crenellation with segments in contact, and small enough so that it does not intersect other segments. This corresponds to subdividing 4 times each edge of the cycles corresponding to polygons.

Claim 5.22. *A cycle in G which is not a triangle contains* 5 *consecutive sides of the same polygon. And so G is $K_{2,2}$ free.*

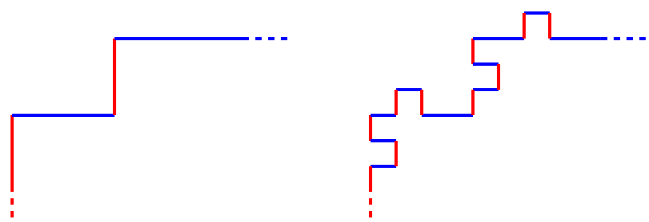

Figure 5.6: Portion of a polygon before and after the adding of the crenellation on each side of the polygon.

Proof. Let Q be a cycle of size at least 4. A z_C point is contained in only 3 segments and the segments containing a z'_C point with $C' \neq C$ are at a distance of at least 4 because of the crenellation. This ensures there is in Q a segment s which does not contain a z_C point, so which is a part of a polygon and more precisely of a crenellation. However, if a cycle hits a segment from a crenellation that does not contain a z_C point, all the paths that the cycle can follow over the crenellation go from one of its sides to another, crossing the 5 adjacent non-zero-length segments of the crenellation. Observe that $K_{2,2}$ is a cycle of size 4, and we just proved that a cycle that is a subgraph of G is either of size 3 or at least 5, so G is $K_{2,2}$ free. □

Observe that the graph G obtained by the above construction satisfies the same properties as the graph constructed in the proof of Theorem 5.10. Additionally,

1. G is a contact 2-DIR (as argued above);
2. G is $K_{2,2}$-free, as proved in Claim 5.22.
3. the polygons corresponding to two variables intersect only in a point of the form z_C, for C a clause where they both appear.
4. G has degree at most 6, where this value could be attained by a segment in contact with z_C, see for example Figure 5.5.

Now that the graph is defined, let us show how it relates to the cycle-hitting problems. As in the previous proof, we denote k_i the number such that the variable gadget for the variable x_i is a k_i-polygon and $k = \sum_{i=1}^{n} k_i$.

Claim 5.23. *For a set $S \subseteq V(G)$ of size at most k, the three following properties are equivalent:*

1. *S is a feedback vertex set,*
2. *S is a odd cycle transversal,*
3. *S is a triangle hitting set.*

Proof. We trivially have (1)⇒(2)⇒(3) so it is enough to prove (3)⇒(1). Let S be a triangle hitting set of G of size at most k. Hitting all the triangles of the polygons of G requires already k vertices, so S has size exactly k and so is a good triangle hitting set as defined in the proof of Theorem 5.10. Claim 5.22 ensures that any cycle of length greater than 3 contains two adjacent non-zero-length segments from the same polygon, and so with different directions. As S is a good triangle hitting set, we know that one of those two segments will be part of S. This proves that S not only hits all the triangles, but it is a set hitting every cycle of G. □

Because our construction still satisfies the properties of the construction done in the Theorem 5.10, the Claim 5.14 is still satisfied. Therefore, an algorithm solving FVS, TH or OCT in time $2^{o(\sqrt{n})}$ would also solve PLANAR-3-SAT with the same complexity, which according to Theorem 5.19 contradicts the ETH. □

5.3 Generalization to the K_r-HITTING problem

For $r \geq 4$, observe that by replacing each of the zero-length segments in the previous constructions by $r-2$ overlapping zero-length segments, solving the K_r-Hitting in the newly obtained graph is equivalent to solving TH in the original graph. We then extend the lower bound for TH, i.e. K_3-HITTING, to the K_r-HITTING problem for $r \geq 4$. The results obtained in this chapter are then summarized in Table 5.1.

<table>
<tr><th colspan="4">Lower bounds (under ETH, with $r \geq 3$)</th></tr>
<tr><th>Restriction</th><th>Class</th><th>Problem</th><th>Bound</th></tr>
<tr><td>None</td><td rowspan="3">2-DIR</td><td rowspan="2">K_r-HITTING, OCT</td><td>$2^{\Omega(n)}$</td></tr>
<tr><td>Maximum degree $\Delta \geq 6$</td><td>$2^{\Omega(\sqrt{\Delta n})}$</td></tr>
<tr><td>$K_{2,2}$-free contact, max. deg. 6</td><td>K_r-HITTING, OCT, FVS</td><td>$2^{\Omega(\sqrt{n})}$</td></tr>
</table>

Table 5.1: Summary of the new lower bounds proved in this chapter. For all the results we can assume a representation of the considered graphs is given in the instance.

Chapter 6
Beyond bidimensionality

Contents

The widely-used bidimensionality framework, already presented in Subsection 3.1.2, provides subexponential FPT algorithms solving a wide range of problems, so called bidimensional problems, for a broad range of graph classes, said to have the SQGM property. In this chapter, is introduced the Almost-SQGM framework (that will be denoted ASQGM from now on). This framework is similar to bidimensionality as it provides subexponential FPT algorithms solving a subset of bidimensional problems for instances in graph classes satisfying weaker conditions than the SQGM property (hence the name of Almost-SQGM).

We first identify sufficient conditions, for any graph class $\mathcal{C}$ included in string graphs, to admit subexponential FPT algorithms for any problem in $\mathcal{P}$, a family of bidimensional $\mathcal{F}$-Hitting problems. Informally, these conditions boil down to the fact that for any $G \in \mathcal{C}$, the local radius of G is polynomial in the clique number of G and in the maximum matching in the neighborhood of a vertex. To demonstrate the applicability of this generic result, we bound the local radius for two special classes: intersection graphs of axis-parallel squares and of contact graphs of segments in the plane. This implies that any problem $\Pi \in \mathcal{P}$ (in particular, FVS) can be solved in:

- $2^{\mathcal{O}(k^{7/8}\log k)}n^{\mathcal{O}(1)}$-time in contact segment graphs,
- $2^{\mathcal{O}(k^{9/10}\log k)}n^{\mathcal{O}(1)}$ in intersection graphs of axis-parallel squares.

We also provide results for TH by solving it in:

- $2^{\mathcal{O}(k^{3/4}\log k)}n^{\mathcal{O}(1)}$-time in contact segment graphs,
- $2^{\mathcal{O}(\sqrt{d}t^2(\log t)k^{2/3}\log k)}n^{\mathcal{O}(1)}$-time in $K_{t,t}$-free d-DIR graphs (intersection of segments with at most d possible slopes).

Organization of the chapter. In Section 6.1, we introduce the class of problems $\mathcal{P}$, a family of bidimensional problems where one has to find a set of size at most k hitting a fixed family of graphs. This class contains in particular FVS, and Pseudo Forest Del (resp. P_t-Hitting) where given a graph G, the goal is to remove a set S of at most k vertices of G such that each connected component of $G - S$ contains at most one cycle (resp. does not contain a path on t vertices as a subgraph). We point out that these three problems are also in the list of problems mentioned in [LPS+23] that admit EPTAS in disk graphs. Then, in Section 6.2 we formalize the concept of ASQGM property, extending the work of [BT22]. In Section 6.3, we provide the preliminary steps of this method (see Corollary 6.4), and the core of the framework is presented in Section 6.4.

There, we provide a sufficient conditions for graph class to admit subexponential FPT algorithms for any problem in $\mathcal{P}$. These conditions mainly boil down to having $ASQGM(\omega, \mu^{\mathrm{N}^\star})$, where $\mu^{\mathrm{N}^\star}$ is, informally, the maximum size of matching in the neighborhood of a vertex. Then, we use the framework of [BT22] to show that string graphs have $ASQGM(\omega, \mathrm{lr})$. Thus, in order to obtain a subexponential FPT algorithm for a problem $\Pi \in \mathcal{P}$ in a given subclass of string graphs, the only challenge is to bound the local radius by a polynomial of ω and $\mu^{\mathrm{N}^\star}$. At the end of Section 6.4, we provide such bounds for square graphs (intersection of axis-parallel squares) and contact segment graphs.

6.1 Problems in $\mathcal{P}$

Definition 6.1. *We denote by $\mathcal{P}$ the class of all vertex deletion problems Π such that:*

1. Π *is bidimensional (see Definition 3.7 for the definition);*

2. *there is an integer* $c_\Pi > 0$ *such that for any solution* S *in a graph* G*, and any* c_Π*-bundle* B *of* G*,* $S \cap B \neq \emptyset$*; and*

3. Π *can be solved on a graph* G *in time* $\mathbf{tw}(G)^{\mathcal{O}(\mathbf{tw}(G))}$.

Claim 6.2. *FVS, Pseudo Forest Del and* P_t*-Hitting for* $t \leq 5$ *belong to* $\mathcal{P}$.

Proof. It is well known that these three problems are bidimensional. For the second condition, one can check that c_Π is equal to 1 for FVS (as a 1-bundle is a triangle) and equal to 2 for Pseudo Forest Del and P_t-Hitting when $t \leq 5$. For the last condition, as FVS corresponds to hit all K_3 as minor and Pseudo Forest Del correspond to hit all $\{H_0, H_1, H_2\}$ as a minor (with H_i is formed by two triangles sharing i vertices), these two problems can be solved in $\mathbf{tw}(G)^{\mathcal{O}(\mathbf{tw}(G))}$ by [BST19]. For P_t-Hitting the result holds by [CMPP17]. □

6.2 Overview of the ASQGM framework

Recall from Chapter 3 that classical geometric graph classes, such as disk graphs for example, cannot have the SQGM property. Indeed, if G is a clique of size a (which is a disk graph, as it is the intersection graph of a disks overlapping), then $\mathbf{tw}(G) = a - 1$ but $\boxplus(G) \leq \sqrt{|G|} = \sqrt{a}$. To overcome this, let us introduce the following notion where the bound on treewidth is allowed to depend on an additional parameter besides $\boxplus(G)$.

Definition 6.3. *Given a graph parameter* p *and a real* $c < 2$*, a graph class* $\mathcal{G}$ *has the* **almost subquadratic grid minor property** *(***ASQGM** *for short) for* p *and* c *if there exists a function* f *such that* $\mathbf{tw}(G) = \mathcal{O}(f(p(G))\boxplus(G)^c)$*. The class* $\mathcal{G}$ *has ASQGM*(p) *if there exists* $c < 2$ *such that* $\mathcal{G}$ *has the ASQGM property for* p *and* c*. The notation is naturally extended to more than one parameter.*

This notion was used implicitly in earlier work (e.g., [GKT14]) but we chose to define it explicitly in order to highlight the contribution f of the parameter p to the treewidth, which is particularly relevant when it can be shown to be small (typically, polynomial). Let us now explain how ASQGM can be used to obtain subexponential parameterized algorithms on geometric graphs.

It was shown in [FLS18] that FVS can be solved in time $2^{\mathcal{O}\left(k^{3/4}\log k\right)}n^{\mathcal{O}(1)}$ in map graphs, a superclass of planar graphs where arbitrary large cliques may exist, as follows. Let $\omega(G)$ denote the order of the largest clique in a graph G. The first ingredient is to prove that map graphs have *ASQGM*(ω), and more precisely that $\mathbf{tw}(G) = \mathcal{O}(\omega(G)\boxplus(G))$. Then, if $\omega(G) \geq k^\varepsilon$ for some ε, the presence of such large clique allows to have subexponential branchings (as a solution of FVS must take almost all vertices of a clique). When $\omega(G) < k^{\varepsilon}$, then the ASQGM property gives that $\mathbf{tw}(G) \leq k^{\varepsilon}\, ⊞(G) \leq k^{\frac{1}{2}+\varepsilon}$ (as before we can immediately answer no if $⊞(G) > \mathcal{O}(\sqrt{k})$). By appropriately choosing ε the authors of [FLS18] obtain the mentioned running time. The same approach also applies to unit disk graphs and has since been improved to $2^{\sqrt{k}\log k} n^{\mathcal{O}(1)}$ in [FLP+19] using a different technique, and finally improved to an optimal $2^{\sqrt{k}}(n+m)$ in [AO21] for similarly sized fat objects (which typically includes unit squares, but not disks, squares, nor segments).

There is also a line of work aiming at establishing ASQGM property for different classes of graphs and parameters, with for example [GKT14] proving that (1) string graphs have ASQGM when the parameter p is the number of times a string is intersected (assuming at most two strings intersect at the same point), and that (2) intersection graphs of "fat" and convex objects have ASQGM when the parameter $p(G)$ is the minimal order of a graph H not subgraph of G (generalizing the degree when H is a star).

6.3 Preliminary branching steps

Our algorithms make use of two of the preprocessing branchings presented in Subsection 3.1.1, the first one is the branching on large cliques, and the second is on large bundles. Again an analysis of these steps can be found in [LPS+22] for FVS in disk graphs. Here we extend them to any problem in $\mathcal{P}$ and any graph class where the maximum clique can be approximated in polynomial time. These two reduction steps are summarized in the following corollary, which is the routine that we will use in our algorithms. The proof is not included as it can be obtained by closely following the proof of Lemma 3.4 in [LPS+22] and described in Subsection 3.1.1.

Corollary 6.4. *Let $\Pi \in \mathcal{P}$. Let $\mathcal{G}$ be a hereditary graph class where the maximum clique can be α-approximated for some constant factor $\alpha \geq 1$ in polynomial time. There exists a $2^{\mathcal{O}\left(\frac{k}{p}\log k\right)} n^{\mathcal{O}(1)}$-time algorithm that, given an instance (G, k) of Π and an integer $p \in [6\alpha c_{\Pi}, k]$, where $G \in \mathcal{G}$, returns a collection $\mathcal{C}$ of size $2^{\mathcal{O}\left(\frac{k}{p}\log k\right)}$ of tuples (G', k', M) such that:*

1. *For any $(G', k', M) \in \mathcal{C}$, (G', k') is an instance of Π where G' is an induced subgraph of G, $\omega(G') \leq p$, and $k' \leq k$;*
2. *M is a c_{Π}-bundle hitting set of G' with $|M| = \mathcal{O}(pk)$, and for any $v \in M$, $\mu(G'[N(v) \setminus M]) < c_{\Pi}$; and*

3. *(G,k) is a yes-instance of Π if and only if there exists $(G',k',M) \in \mathcal{C}$ such that (G',k') is a yes-instance of Π.*

6.4 Positive results via ASQGM

6.4.1 From $ASQGM(\omega, \mu^{\mathrm{N}^\star})$ to subexponential algorithms.

In this section we provide subexponential parameterized algorithms for problems of $\mathcal{P}$ in any class that has the $ASQGM(\omega, \mu^{\mathrm{N}^\star})$ property, with $\mu^{\mathrm{N}^\star}$ a subneighborhood function as defined below.

Definition 6.5. *Given a graph G, a* **subneighborhood function** *of G is any function $\mathrm{N}^\star : V(G) \to 2^{V(G)}$ such that for any $v \in V(G)$, $\mathrm{N}^\star(v) \subseteq N(v)$. Moreover, if for any $u \in V(G)$, $|\{v \in V(G) : u \in \mathrm{N}^\star(v)\}| \leq c$ for some $c \in \mathbb{N}$ then we say that $\mathrm{N}^\star$ has c-***bounded occurrences**.

Given a subneighborhood function $\mathrm{N}^\star$, we define $\mu^{\mathrm{N}^\star}(v)$ as the maximum number of edges of a matching in $G[\mathrm{N}^\star(v)]$. We denote by $\mu^{\mathrm{N}^\star}(G)$ the maximum of $\mu^{\mathrm{N}^\star}$ over $V(G)$.

For example in square graphs, we will fix a representation $\mathcal{S}$, and define $\mathrm{N}^\star(v)$ as the set of neighbors of v whose square is smaller than the one of v.

The main theorem from this subsection is the following. Recall that $\mathcal{P}$ encompasses fundamental algorithmic problems such as FVS, Pseudo Forest Del and P_t-Hitting for $t \leq 5$ (Claim 6.2).

Theorem 6.6. *Let Π be a problem of $\mathcal{P}$ and $\mathcal{C}$ be a hereditary graph class such that:*

- *maximum clique can be $\mathcal{O}(1)$-approximated in polynomial time in $\mathcal{C}$;*
- *for any $G \in \mathcal{C}$, there exists a subneighborhood function $\mathrm{N}^\star$ that has $\mathcal{O}(\omega(G)^{c_1})$-bounded occurrences for some $c_1 \in \mathbb{N}$; and*
- *$\mathcal{C}$ has the $ASQGM(\omega, \mu^{\mathrm{N}^\star})$ property, i.e., there exists a multivariate polynomial P such that for all $G \in \mathcal{C}$, we have $\mathbf{tw}(G) = \mathcal{O}(P(\omega(G), \mu^{\mathrm{N}^\star}(G)) \cdot \boxplus(G))$.*

Then, Π admits a parameterized subexponential algorithm on $\mathcal{C}$. More precisely, for $\varepsilon > 0$ such that $P(k^\varepsilon, k^{(c_1+2)\varepsilon}) = \mathcal{O}(k^{\frac{1}{2}-\varepsilon})$, Π admits a parameterized subexponential algorithm on $\mathcal{C}$ running in time $2^{\mathcal{O}(k^{1-\varepsilon}\log(k))}$. This algorithm does not need a representation except if one is required for finding the $\mathcal{O}(1)$-approximation of a maximum clique.

Lemma 6.7. *Let Π be a problem of $\mathcal{P}$. Consider a graph G and $\mathrm{N}^\star$ a c-bounded occurrences subneighborhood function of G. Let $M \subseteq V(G)$ be a c_Π-bundle hitting set of G such that for any vertex $v \in M$, $\mu(G[N(v)] - M) < c_\Pi$. Then for every positive integer $\tau \geq c_\Pi$, there exists a set $B \subseteq V(G)$ of size $|B| = \frac{c|M|}{\tau - c_\Pi + 1}$ such that $\mu^{\mathrm{N}^\star}(G - B) \leq \tau$.*

Proof. Let τ a positive integer with $\tau \geq c_\Pi$, and let us define $B = \{v \in V(G) : \mu^{\mathrm{N}^\star}(v) \geq \tau\}$ the set of vertices with "big" $\mu^{\mathrm{N}^\star}$ in G. Let us first prove that for any $v \in B$, $|\mathrm{N}^\star(v) \cap M| \geq \mu^{\mathrm{N}^\star}(v) - c_\Pi + 1$. Let $E' \subseteq E(G)$ be a maximum matching in $G[\mathrm{N}^\star(v)]$ with $|E'| = \mu^{\mathrm{N}^\star}(v)$. Observe that we cannot have c_Π edges $e \in E'$ such that $V(e) \cap M = \emptyset$ as if $v \notin M$, then vertices of E' together with v would form a c_Π-bundle not hit by M, a contradiction, and if $v \in M$, this would contradict the hypothesis $\mu(G[N(v)] - M) < c_\Pi$. Thus, there is at least $|E'| - c_\Pi + 1$ edges of E' intersecting M, leading to the desired inequality. Thus, we get

$$|B|\tau \leq \sum_{v \in B} \mu^{\mathrm{N}^\star}(v) \leq \sum_{v \in B} (|\mathrm{N}^\star(v) \cap M| + c_\Pi - 1).$$

Moreover, as for any $v \in V(G)$ there are at most c vertices u such that $v \in \mathrm{N}^\star(u)$, we get $\sum_{v \in B} |\mathrm{N}^\star(v) \cap M| \leq c|M|$ by the pigeonhole principle (if the inequality was false, then there would exists $v \in M$ with $|\{u : v \in \mathrm{N}^\star(u)\}| > c$). This leads to $|B| = \frac{c|M|}{\tau - c_\Pi + 1}$. $\square$

We are now ready to describe the general algorithm to solve Π.

Proof of Theorem 6.6. Given an instance (G, k) of Π, we first use Corollary 6.4 with $p = k^\varepsilon$ to obtain in time $2^{\mathcal{O}(k^{1-\varepsilon} \log(k))}$ the set of $2^{\mathcal{O}(k^{1-\varepsilon} \log(k))}$ triples (G_2, M, k_2) with $k_2 \leq k$, $|M| = \mathcal{O}(k^{1+\varepsilon})$, and $\omega(G_2) \leq k^\varepsilon$.

In order to solve Π on (G, k), it is now enough to solve it on these instances (G_2, k_2). Observe that applying the Lemma 6.7 to such (G_2, k_2, M) triple with $\tau \geq c_\Pi$ gives a set B of size at most $\frac{c|M|}{\tau - c_\Pi + 1} = \mathcal{O}\left(\frac{\omega(G_2)^{c_1} k^{1+\varepsilon}}{\tau - c_\Pi + 1}\right) = \mathcal{O}\left(\frac{k^{1+\varepsilon+\varepsilon c_1}}{\tau - c_\Pi + 1}\right)$ such that $G_3 = G_2 \setminus B$ satisfies $\mu^{\mathrm{N}^\star}(G_3) \leq \tau$.

By assumption on the *ASQGM* property we then have $\mathbf{tw}(G_3) = \mathcal{O}(P(k^\varepsilon, \tau) \boxplus(G))$. Moreover, $\mathbf{tw}(G_2) \leq \mathbf{tw}(G_3) + |B| = \mathcal{O}(P(k^\varepsilon, \tau) \boxplus(G)) + \mathcal{O}\left(\frac{k^{1+\varepsilon+\varepsilon c_1}}{\tau - c_\Pi + 1}\right)$ as removing a vertex decreases the treewidth by at most 1. We set $\tau = k^{(c_1+2)\varepsilon}$. By assumption we have $P(k^\varepsilon, k^{(c_1+2)\varepsilon}) = \mathcal{O}(k^{\frac{1}{2}-\varepsilon})$. As Π is bidimensional, there exists c_1 such that if $\boxplus(G) > c_1\sqrt{k}$, then (G, k) is a no-instance.

Thus, as $\mathbf{tw}(G_2) = \mathcal{O}\left(k^{\frac{1}{2}-\varepsilon} \boxplus(G)\right) + \mathcal{O}\left(\frac{k^{1+\varepsilon+\varepsilon c_1}}{\tau}\right) = \mathcal{O}\left(k^{\frac{1}{2}-\varepsilon} \boxplus(G)\right) + \mathcal{O}(k^{1-\varepsilon})$, observe that if $\boxplus(G) \leq c_1\sqrt{k}$, then there exists a constant c such that $\mathbf{tw}(G_2) \leq$

$ck^{1-\varepsilon}$. Thus, we use the treewidth approximation of Theorem 2.2 on G_2 with $\ell = ck^{1-\varepsilon}$ to obtain in $2^{\mathcal{O}(\ell)}n^{\mathcal{O}(1)}$ either a $2\ell+1$ treewidth decomposition, or conclude that $\mathbf{tw}(G_2) > \ell$. In the latter case, this implies that $\boxplus(G) > c_1\sqrt{k}$, and thus we can conclude that (G, k) is a no instance. Otherwise, by the definition of problems in $\mathcal{P}$ we can solve Π in time $\mathbf{tw}(G_2)^{\mathcal{O}(\mathbf{tw}(G_2))}$, which gives the claimed overall time complexity of $2^{\mathcal{O}(k^{1-\varepsilon}\log(k))} \times \mathbf{tw}(G_2)^{\mathcal{O}(\mathbf{tw}(G_2))} = 2^{\mathcal{O}(k^{1-\varepsilon}\log(k))}$. □

6.4.2 From $ASQGM(\omega, \mathrm{lr})$ to $ASQGM(\omega, \mu^{\mathrm{N}^\star})$.

To be able to use Theorem 6.6, we need to deal with graph classes that have the $ASQGM(\omega, \mu^{\mathrm{N}^\star})$ property. This section provides a general framework for obtaining this property via the local radius. The local radius was originally introduced by Lokshtanov et al. [LPS+23] for disks graphs in the context of approximation algorithms. Here we first extend this definition to string graphs. To that end, we will see string graphs as graphs admitting a ***thick representation***. In such a representation every vertex v of the considered graph G corresponds to a subset $\mathcal{D}_v$ of the plane that is homeomorphic to a disk, two intersecting such regions have an intersection with non-empty interior, and the number of maximal connected regions $\mathbb{R}^2 \setminus \bigcup_{v\in V(G)} \partial\mathcal{D}_v$ is finite.

To turn a string representation into a thick one, it simply suffices to thicken each string by a small enough amount so that no new intersections occur. On the other hand, note that any thick representation can be turned into a string representation by replacing each connected subset of the plane $\mathcal{D}_u$ by a string that almost completely fills its interior. Note that a thick representation is not necessarily a pseudo-disk representation as here, the intersection of two regions, $\mathcal{D}_u \cap \mathcal{D}_v$, may not be connected, or it may also be that $\mathcal{D}_u \setminus \mathcal{D}_v$ is not connected. Thick representations allow us to extend the definition of local radius to all string graphs. The next definition is illustrated Figure 6.2.

Definition 6.8. *Let G be a string graph and $\mathcal{S}$ be a thick representation of it. Let $\mathcal{X}$ be the set of all maximal connected region $\mathcal{R}$ of $\mathbb{R}^2 \setminus \bigcup_{D\in\mathcal{S}} \partial D$, contained in at least one object of $\mathcal{S}$. We define the* **arrangement graph** *of $\mathcal{S}$, denoted $A_\mathcal{S}$, by:*

- *adding one vertex of each region of $\mathcal{X}$, and*
- *adding an edge between two vertices if the boundaries of their regions share a common arc.*

Moreover, for each $v \in G$, we denote $\mathcal{R}_\mathcal{S}(v) \subseteq \mathcal{X}$ the set of regions included in $\mathcal{D}_v$ (recall that $\mathcal{D}_v$ is the region associated to v), and $V_\mathcal{S}(v) \subseteq V(A_\mathcal{S})$ the set of vertices associated to the regions of $\mathcal{R}_\mathcal{S}(v)$ (implying $|V_\mathcal{S}(v)| = |\mathcal{R}_\mathcal{S}(v)|$). Finally, we denote $A_\mathcal{S}(v) = A_\mathcal{S}[V_\mathcal{S}(v)]$.

Definition 6.9 (from [LPS⁺23], extended here to string graphs)**.** *Let G be a string graph.*

- *Given a thick representation $\mathcal{S}$ of G,*
  - *for any $v \in V(G)$, we define $\mathrm{lr}_{\mathcal{S}}(v)$ as the radius of the graph $A_{\mathcal{S}}(v)$*
  - *we define $\mathrm{lr}_{\mathcal{S}}(G) = \min_{v \in V(G)} \mathrm{lr}_{\mathcal{S}}(v)$*
- *the local radius $\mathrm{lr}(G)$ of G is the minimum over all thick representation $\mathcal{S}$ of G of $\mathrm{lr}_{\mathcal{S}}(G)$.*

In order to show ASQGM we use the framework of Baste and Thilikos [BT22] (originally designed for the classic SQGM property), that we recall now.

Definition 6.10 (contractions [BT22])**.** *Given a non-negative integer c, two graphs H and G, and a surjection $\sigma : V(G) \to V(H)$ we write $H \leq^c_\sigma G$ if*

- *for every $x \in V(H)$, the graph $G[\sigma^{-1}(x)]$ has diameter at most c and*
- *for every $x, y \in V(H)$, $xy \in E(H) \iff G[\sigma^{-1}(x) \cup \sigma^{-1}(y)]$ is connected.*

We say that H is a c-diameter contraction of G if there is a surjection σ such that $H \leq^c_\sigma G$ and we write this $H \leq^c G$. Moreover, if σ is such that for every $x \in V(H)$, $|\sigma^{-1}(x)| \leq c'$, then we say that H is a c'-size contraction of G, and we write $H \leq^{(c')} G$. If there exists an integer c such that $H \leq^c G$, then we say that H is a contraction of G.

Definition 6.11 ((c_1, c_2)-extension [BT22])**.** *Given a class of graph $\mathcal{G}$ and two non-negative integers c_1 and c_2, we define the* **(c_1, c_2)-extension** *of $\mathcal{G}$, denoted by $\mathcal{G}^{(c_1,c_2)}$, as the class containing every graph H such that there exist a graph $G \in \mathcal{G}$ and a graph J that satisfy $G \leq^{(c_1)} J$ and $H \leq^{c_2} J$ (see Figure 6.1).*

$$\begin{tikzcd}
& J \arrow[dl, "c_1\text{-size contraction}"'] \arrow[dr, "c_2\text{-diameter contraction}"] & \\
G \in \mathcal{G} & & H \in \mathcal{G}^{(c_1,c_2)}
\end{tikzcd}$$

Figure 6.1: A graphical representation of the definition of $\mathcal{G}^{(c_1,c_2)}$, adapted from [BT22].

Lemma 6.12 (implicit in the proof of [BT22, Theorem 15])**.** *For every integers c_1, c_2 and $G \in \mathcal{P}^{(c_1,c_2)}$, with $\mathcal{P}$ the class of planar graphs, we have $\mathbf{tw}(G) = \mathcal{O}(c_1 c_2 \boxplus(G))$.*

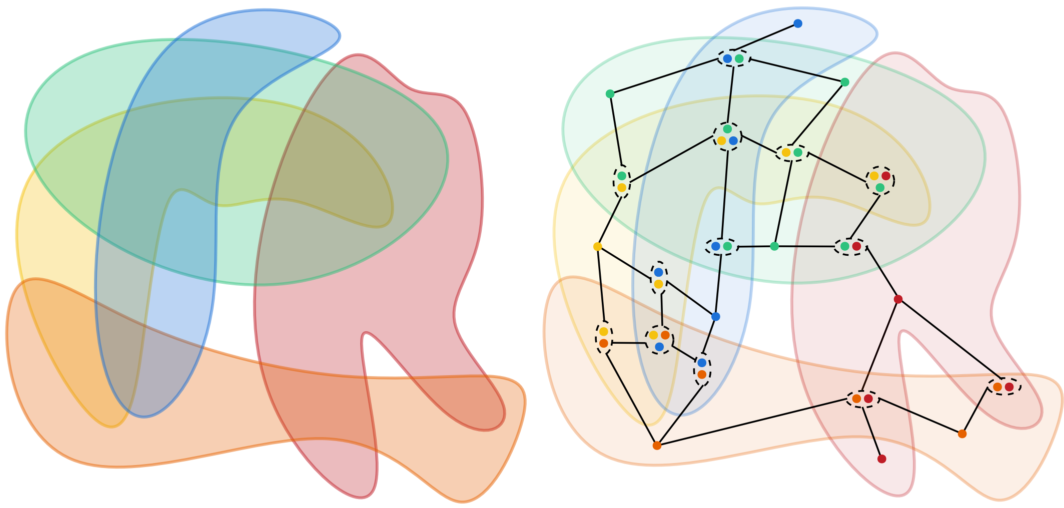

Figure 6.2: Left: thick representation of a string graph G. Right: Illustrates both $A_{\mathcal{S}}$ and the graph J used in the proof of Theorem 6.13. To visualise $A_{\mathcal{S}}$, consider that each black dotted ellipse is a single vertex (we have $|V(A_{\mathcal{S}})| = 23$). Moreover, if v is the vertex represented in red, we have $|V_{\mathcal{S}}(v)| = 6$ and $\mathrm{lr}_{\mathcal{S}}(v) = 2$. To visualise J: for each maximal connected region $\mathcal{R}$ of $\mathbb{R}^2 \setminus \bigcup_{D\in\mathcal{S}} \partial D$, the clique $K_{\mathcal{R}}$ with more than one vertex is represented by a black dotted ellipse around the clique. For readability only one edge is represented between two cliques instead of the complete bipartite graph.

The main result of this section is the following.

Theorem 6.13. *String graphs have the ASQGM(ω, lr) property, more precisely for a string graph G we have* $\mathbf{tw}(G) = \mathcal{O}(\omega(G) \cdot \mathrm{lr}(G) \cdot \boxplus(G))$.

Proof. Let G be a string graph, and $\mathcal{S}$ a thick representation such that $\mathrm{lr}_{\mathcal{S}}(G) = \mathrm{lr}(G)$. Let us define a graph J as follows, Figure 6.2 is a representation of the construction. For any maximal connected region $\mathcal{R}$ of $\mathbb{R}^2 \setminus \bigcup_{D\in\mathcal{S}} \partial D$, we add to J a clique $K_{\mathcal{R}}$ of size $\mathrm{ply}(\mathcal{R})$. Then, for any pair of regions $\{\mathcal{R}_1, \mathcal{R}_2\}$ that share a common arc, we add all edges between $K_{\mathcal{R}_1}$ and $K_{\mathcal{R}_2}$. For any $v \in V(G)$, we associate a set $X(v) \subseteq V(J)$ such that for any $\mathcal{R} \in \mathcal{R}_{\mathcal{S}}(v)$, $|X(v) \cap K_{\mathcal{R}}| = 1$, and such that $X(v) \cap X(u) = \emptyset$ for any $u \neq v$. Notice that the condition $X(v) \cap X(u) = \emptyset$ is possible as $|K_{\mathcal{R}}| = \mathrm{ply}(\mathcal{R})$, and thus any vertex v can take its "private" vertex in $X(v) \cap \mathcal{R}$ for any $\mathcal{R} \in \mathcal{R}_{\mathcal{S}}(v)$.

Let us prove that G is a $\mathrm{lr}(G)$-diameter contraction of J by defining a surjection $\sigma : V(J) \to V(G)$ as follows. For any $v \in V(G)$, we define $\sigma^{-1}(v) = X(v)$ (informally we contract all vertices in $X(v)$). As for any $v \in V(G)$, $J[X(v)]$ is isomorphic to $A_{\mathcal{S}}(v)$, we immediately have $\mathrm{diam}(J[\sigma^{-1}(v)]) = \mathrm{lr}(G)$. Moreover, it is straightforward to check that for every $x, y \in V(G)$, $xy \in E(G) \iff J[\sigma^{-1}(x) \cup \sigma^{-1}(y)]$ is connected. Now, observe that $A_{\mathcal{S}}$ (which is planar) is a $\text{ply}(\mathcal{S})$-size contraction of J using $\sigma' : V(J) \to V(A_{\mathcal{S}})$ such that for any $v \in V(A_{\mathcal{S}})$, v corresponding to a region $\mathcal{R}$ of the plane delimited by the boundaries of the objects of $\mathcal{S}$, ${\sigma'}^{-1}(v) = K_R$. As $\text{ply}(\mathcal{S}) \leq \omega(G)$, we get the desired result. □

The following corollary is immediate from Theorem 6.6 and Theorem 6.13.

Corollary 6.14. *Given an hereditary graph class $\mathcal{C}$ which is a subclass of string graphs such that*

- *maximum clique can be $\mathcal{O}(1)$-approximated in polynomial time,*
- *for any $G \in \mathcal{C}$, there exists a subneighborhood function $\mathrm{N}^\star$ that has $\mathcal{O}(\omega(G)^{c_1})$-bounded occurrences for some $c_1 \in \mathbb{N}$, and*
- *there exists a multivariate polynomial such that for any $G \in \mathcal{C}$, $\mathrm{lr}(G) = P(\omega(G), \mu^{\mathrm{N}^\star}(G))$.*

Then, any problem $\Pi \in \mathcal{P}$ admits a parameterized subexponential algorithm on $\mathcal{C}$. More precisely, let $P'(\omega(G), \mu^{\mathrm{N}^\star}(G)) = \omega(G)P(\omega(G), \mu^{\mathrm{N}^\star}(G))$. For any $\varepsilon > 0$ such that $P'(k^\varepsilon, k^{(c_1+2)\varepsilon}) = \mathcal{O}(k^{\frac{1}{2}-\varepsilon})$, FVS can be solved in time $k^{\mathcal{O}(k^{1-\varepsilon})} n^{\mathcal{O}(1)}$. This algorithm does not need a representation except if one is required for finding the $\mathcal{O}(1)$-approximation of a maximum clique.

6.4.3 Upper bounding the local radius for square graphs

To demonstrate the applicability of Corollary 6.14, we show here that square graphs matchs the conditions of Corollary 6.14. This requires to define an appropriate $\mathrm{N}^\star$ and prove that $\mathrm{lr}(G) = \omega(G)^{O(1)} \cdot \mu^{\mathrm{N}^\star}(G)^{\mathcal{O}(1)}$. A second application, for contact segment graphs, is given in the next section.

Given a square graph, by slightly altering the sizes and positions of the squares in a collection we can obtain a collection where exactly the same pairs of squares intersect and, in addition, all the side lengths of the squares are different from each other and no two sides squares are aligned. Furthermore this can easily be performed in polynomial time. From now on we will assume that all the representations we consider satisfy this property.

The first requirement of Corollary 6.14 is provided by following lemma from [BGM20], which describes an EPTAS for the clique problem in the more general case of the intersection graph of a fixed convex geometric shape with a central symmetry, while allowing rescaling.

Theorem 6.15 ([BGM20])**.** *There is a polynomial-time 2-approximation of maximum clique in intersection graphs of squares, even when no representation is provided.*

Definition 6.16. *Given a square representation $\mathcal{S} = (\mathcal{D}_v)_{v \in V(G)}$ of a graph G, we denote $\ell_{\mathcal{S}}(\mathcal{D}_v)$ the length of a side of the square $\mathcal{D}_v$, $N^-_{\mathcal{S}}(v)$ (resp. $N^+_{\mathcal{S}}(v)$) the set of vertices u such that $u \in N_G(v)$ and $\ell_{\mathcal{S}}(\mathcal{D}_u) < \ell_{\mathcal{S}}(\mathcal{D}_v)$ (resp. $>$). When $\mathcal{S}$ is clear from the context, we will instead write ℓ, N^- and N^+.*

As the lengths of all sides differ, $\{N^+(v), N^-(v)\}$ is a partition of $N(v)$ for every vertex v.

Lemma 6.17. *Given a square representation $\mathcal{S}$ of a graph G, N^- is a $\mathcal{O}(\omega(G))$-occurrences bounded subneighborhood function.*

Proof. N^- is clearly a subneighborhood function. For $v \in V(G)$, observe that a square larger than $\mathcal{D}_v$ has to contain one of the four corners of $\mathcal{D}_v$ if the two squares intersect. However, a corner of $\mathcal{D}_v$ cannot be contained in more than $\omega(G)$ squares. Hence there are at most $4\omega(G)$ vertices $u \in V(G)$ such that $v \in N^-(u)$, and so N^- is $4\omega(G)$-occurrences bounded. □

We will prove that choosing $N^* = N^-$ allows us to bound the local radius.

Definition 6.18. *Given a square graph G with representation $\mathcal{S}$, for any $v \in G$, we define $H(v)$ as a minimum vertex cover of $G[N^-(v)]$, $I(v) = N^-(v) \setminus H(v)$, and $X(v) = H(v) \cup N^+(v)$.*

Claim 6.19. *For every vertex v of a square graph G with representation $\mathcal{S}$, the following properties hold:*

1. *$I(v)$ is an independent set of G;*
2. *$|H(v)| \leq 2\,\mu^{\mathrm{N}^\star}(G)$;*
3. *$|N^+(v)| = \mathcal{O}(\omega(G))$ (as in the proof of Lemma 6.17);*
4. *$|X(v)| = \mathcal{O}(\mu^{\mathrm{N}^\star}(G) + \omega(G))$; and*
5. *$\{X(v), I(v)\}$ is a partition of $N(v)$.*

Definition 6.20. *For a curve $\mathcal{C} : [0,1] \to \mathbb{R}^2$ such that for $t \in [0,1]$, $\mathcal{C}(t) = (x(t), y(t))$, we say that $\mathcal{C}$ is* **monotonic** *if the functions x and y are monotonic. For $k \geq 2$ we say that $\mathcal{C}$ is k-monotonic if it is the composition[1] of k monotonic curves.*

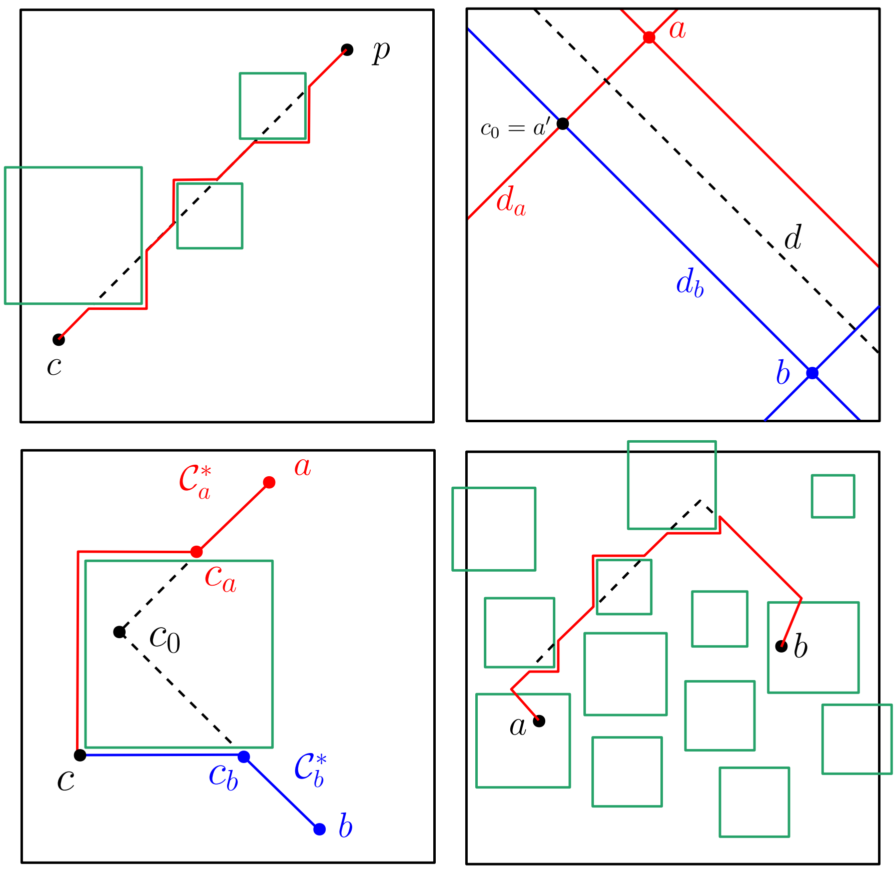

Figure 6.3: Illustrations of the construction used in the proof of the Lemma 6.21. Squares of $I(v)$ are represented in green. Top left: construction used for the Claim 6.22 . Top right: construction used for the Claim 6.23. Bottom left: construction used for Claim 6.24. Observe that in this situation c_a and c_b are next to opposite sides of the square containing c_0, that $\mathcal{C}_a^*$ can be extended in an counter-clockwise direction, and $\mathcal{C}_b^*$ in a clockwise direction, which ensure the existence of a common point c of their monotonic extensions. Bottom right: an example of a 4-monotonic curve between a and b obtained by the construction of Lemma 6.21. Observe that only two squares of $I(v)$ are crossed.

Recall in the next Lemma that $\mathcal{D}_{I(v)}$ denotes the union of all squares in $I(v)$.

Lemma 6.21. *Let G be a square graph and $\mathcal{S}$ a representation. Let $v \in V(G)$ and a, b two points contained in $\mathcal{D}_v$. There exists a 4-monotonic curve $\mathcal{C}$ contained in*

[1]A curve $\mathcal{C}(t) = (x(t), y(t))$ is the ***composition*** of k curves $\mathcal{C}_i(t) = (x_i(t), y_i(t))$ for $i \in \{1, \dots, k\}$ if $(x(0), y(0)) = (x_1(0), y_1(0))$, $(x(1), y(1)) = (x_k(1), y_k(1))$, $(x_i(1), y_i(1)) = (x_{i+1}(0), y_{i+1}(0))$ for every $i \in \{1, \dots, k-1\}$ and the set of points $\{(x(t), y(t)), t \in [0, 1]\}$ is the union of the $\{(x_i(t), y_i(t)) : t \in [0, 1]\}$ for $i \in \{1, \dots, k\}$.

$\mathcal{D}_v$ joining the point a to the point b, and crossing at most twice a boundary of the squares of $I(v)$.

Proof. In what follows, what we call a ***diagonal line (resp. half line)*** any line (resp. half line) having an angle +45° or −45° with the horizontal axis, and a ***diagonal of a point p in the plan*** a diagonal half line whose endpoint is p.

The first step for the creation of the curve is to reduce to the case where the point a and b are outside $\mathcal{D}_{I(v)}$. If this is not the case, for example if a in contained in a square $s = \mathcal{D}_u$ with $u \in I(v)$, we create a rectilinear curve from a toward the outside of s, in a direction such that the intersection of the curve with the boundary of s is still in $\mathcal{D}_v$ (see the construction in Figure 6.3 for an example of such reduction). As such curve is monotonic and crosses the boundary of a square of $I(v)$ exactly once, after the reduction we are in the case where we want to construct a 2-monotonic curve between two points of $\mathcal{D}_v \setminus \mathcal{D}_{I(v)}$ such that no square of $I(v)$ is crossed. In what follow we suppose we have reduced to this case and we still denote a and b the two points of $\mathcal{D}_v \setminus \mathcal{D}_{I(v)}$ we want to join by a curve.

Claim 6.22. *Given two points $c, p \in \mathcal{D}_v \setminus \mathcal{D}_{I(v)}$ on the same diagonal line, there is a monotonic curve included in $\mathcal{D}_v \setminus \mathcal{D}_{I(v)}$ between c and p.*

Proof. The construction is represented in Figure 6.3. The curve is created by starting from the point c, then by following the diagonal line toward p. When encountering a square $s = \mathcal{D}_u$ of a vertex $u \in I(v)$, it is always possible of getting around s in order to join back the diagonal on the other side, and doing so in a direction such that the curve is still monotonic and contained in $\mathcal{D}_v$. □

Claim 6.23. *There are diagonals d_a of a and d_b of b intersecting on a point $c_0 \in \mathcal{D}_v$.*

Proof. Consider the line d parallel to the top left to bottom right diagonal of $\mathcal{D}_v$ (see Figure 6.3), at equal distances of the points a and b. By symmetry of the square and of the variables a and b, we can suppose that d goes from top left to bottom right, is above the diagonal of $\mathcal{D}_v$, and that a is above d. The symmetric a' of the point a with respect to d is inside $\mathcal{D}_v$ and is contained in a diagonal of both a and b. □

Now, if $c_0 \in \mathcal{D}_v \setminus \mathcal{D}_{I(v)}$, composing the two curves toward c_0 given by the previous claim gives the wanted result.

It remains to deal with the case where c_0 lies in some square $s = \mathcal{D}_u$ for $u \in I(v)$. Let c_a be a point of d_a between a and the square s, at an infinitely small distance outside of s. Claim 6.22 gives a monotonic curve $\mathcal{C}_a^*$ from a to c_a. In the same way we define c_b and $\mathcal{C}_b^*$.

Claim 6.24. *There exists a point $c \in \mathcal{D}_v \setminus \mathcal{D}_{I(v)}$ such that $\mathcal{C}_a^*$ and $\mathcal{C}_b^*$ can be extended to c while still being monotonic and contained in $\mathcal{D}_v \setminus \mathcal{D}_{I(v)}$.*

Proof. We can assume that d_a and d_b are perpendicular as otherwise the points a and b are on the same diagonal and so Claim 6.22 gives the wanted result by taking $c = b$. Observe that if c_a and c_b are arbitrarily close to the same side of s, then prolonging $\mathcal{C}_a^*$ toward c_b would keep the curve monotonic, as $\mathcal{C}_a^*$ was already going toward d_b as d_a and d_b intersect in s. Therefore, taking $c = c_b$ would give the wanted result.

Otherwise, if c_a and c_b are at arbitrarily small distance from two different sides, observe that the curve $\mathcal{C}_a^*$ can be extended running alongside the boundary of s until crossing 2 corners. The same is true for $\mathcal{C}_b^*$ so the only situation where those extensions do not cross each other would be if c_a and c_b are next to opposite side of s, and that the orientations of d_a and d_b force the extensions of $\mathcal{C}_a*$ and $\mathcal{C}_b^*$ to go in the same direction around s. However, this is impossible: as d_a and d_b cross each other inside of s, one extension will go clockwise around s and the other counterclockwise (see Figure 6.3). This ensures that $\mathcal{C}_a^*$ and $\mathcal{C}_b^*$ can be extended around s while still being monotonic in order for them to join on a point c while staying outside of $\mathcal{D}_{I(v)}$. □

Composing the two curves obtained by the above claim gives a path as wanted. □

We are now ready to prove the main combinatorial statement of this section.

Lemma 6.25. *Let G be a square graph. There exists a subneighborhood function* $\mathrm{N}^\star$ *which is $\omega(G)$-occurrences bounded and such that* $\mathrm{lr}(G) = \mathcal{O}(\mu^{\mathrm{N}^\star}(G) + \omega(G))$.

Proof. Let $\mathcal{S}$ be a square representation of G, and let $\mathrm{N}^\star$ as defined in Definition 6.16, which is $\omega(G)$-occurrences bounded according to Lemma 6.17. Let us now prove that $\mathrm{lr}_{\mathcal{S}}(G) = \mathcal{O}(|X(v)|)$. This will imply the required result as $\mathrm{lr}(G) \leq \mathrm{lr}_{\mathcal{S}}(G)$ and $|X(v)| = \mathcal{O}(\mu^{\mathrm{N}^\star}(G) + \omega(G))$ by Claim 6.19. To that end, let us bound the diameter of $A_{\mathcal{S}}[V_{\mathcal{S}}(v)]$. Let u, v be two vertices of $A_{\mathcal{S}}[V_{\mathcal{S}}(v)]$, and let us bound the distance between these two vertices. Remember that any vertex in $A_{\mathcal{S}}[V_{\mathcal{S}}(v)]$ corresponds to an inclusion-wise maximal rectangular region of the plane included in $\mathcal{D}_v$, and delimited by edges of squares of $\mathcal{S}$. Let a and b be points in the regions of u and v respectively. Notice that to any curve inside $\mathcal{D}_v$ we can associate a path in $A_{\mathcal{S}}[V_{\mathcal{S}}(v)]$ by considering the sequence of regions visited by $\mathcal{C}$, and associate to each of the region its corresponding vertex in $A_{\mathcal{S}}[V_{\mathcal{S}}(v)]$ (see Figure 6.4). Thus, we will upper bound the distance from u to v in $A_{\mathcal{S}}[V_{\mathcal{S}}(v)]$ by constructing a curve $\mathcal{C}$ from a to b, and by counting the length of the sequence of regions visited by $\mathcal{C}$.

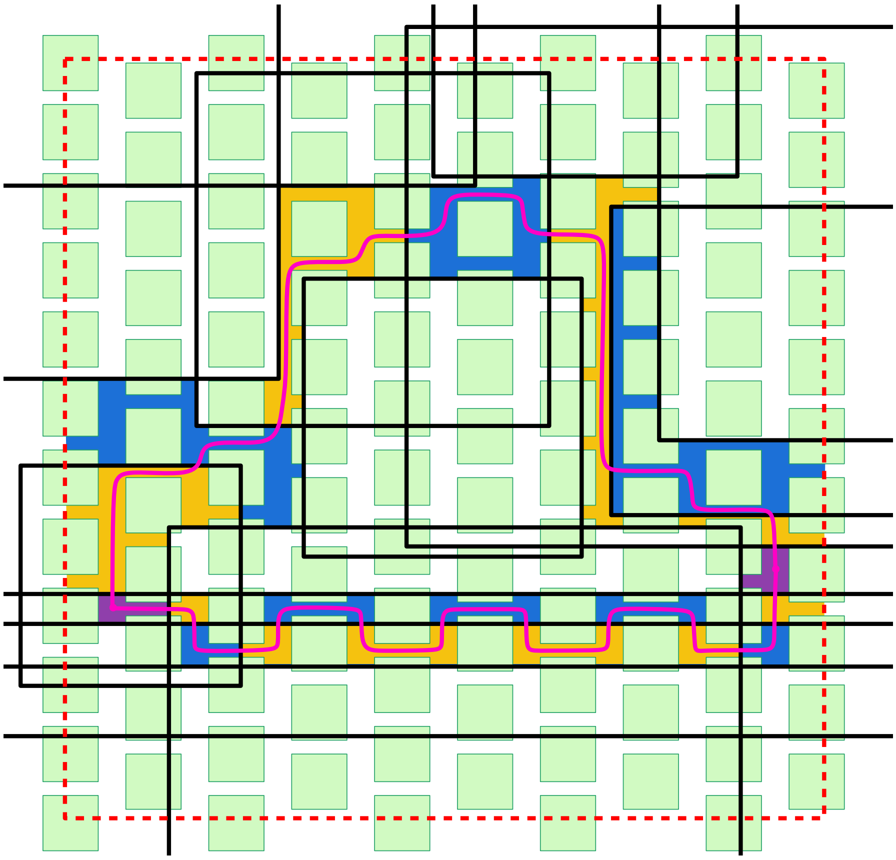

Figure 6.4: Examples of paths in the arrangement graph, with $\mathcal{D}_v$ represented with a dashed red square, $I(v)$ by green squares and the sides of the squares of $X(v)$ in black. Here we can see two curves between the two purple regions, $\mathcal{C}_1$ (that goes up and then down) and $\mathcal{C}_2$, and the path in $A_S(v)$ associated to each curve as in the proof of Lemma 6.25, where the regions traversed by the paths are alternatively colored blue and yellow. Notice that $\mathcal{C}_1$ is 2-monotone, whereas $\mathcal{C}_2$ is c-monotone, where c could be made arbitrary large by creating more and smaller squares in $I(v)$. As c is large, there is a side of a square in $X(v)$ crossed many times (eight) by $\mathcal{C}_2$, and thus we do not use curve like $\mathcal{C}_2$ in the proof.

We use for $\mathcal{C}$ the 4-monotonic curve between a and b defined in Lemma 6.21. Observe the following property π_0: any monotonic curve inside $\mathcal{D}_v$ crosses at most $4|X(v)|$ sides of squares in $X(v)$. Indeed, as each square in $X(v)$ has at most 4 sides intersecting $\mathcal{D}_v$, and any side, as a vertical or horizontal segment intersecting in $\mathcal{D}_v$, can be crossed at most one time by a monotonic curve. Observe also that, each time $\mathcal{C}$ leaves its current region, $\mathcal{C}$ must cross a side of a square in $N(v)$. However, the total number of crossings between $\mathcal{C}$ and a side of a square in $N(v)$

is at most $16|X(v)| + 4$, as each of the four monotonic part of $\mathcal{C}$ crosses at most $4|X(v)|$ sides of squares in $X(v)$ (by π_0), and $\mathcal{C}$ crosses at most 4 sides of squares in $I(v)$ (the worst case being when $a \neq a'$, and $\mathcal{C}_{a \to a'}$ crosses the corner of the square in $I(v)$ containing a, and same for b, b'). Thus, the curve $\mathcal{C}$ goes from a region to the next one at most $16|X(v)| + 4$ times, implying that the diameter of $A_{\mathcal{S}}[V_{\mathcal{S}}(v)]$, and so the local radius $\mathrm{lr}_{\mathcal{S}}(G)$, are in $\mathcal{O}(|X(v)|)$.

□

As announced in the introduction of the section, we are now able to apply Corollary 6.14.

Theorem 6.26. *Any problem $\Pi \in \mathcal{P}$ can be solved in time $2^{\mathcal{O}(k^{9/10} \log(k))} n^{\mathcal{O}(1)}$ in square graphs, even when no representation is given.*

Proof. Let $\Pi \in \mathcal{P}$. According to Theorem 6.15, Lemma 6.25, we can apply Corollary 6.14 with $c_1 = 1$, and $P(x, y) = x + y$. This implies that for any ε such that $k^{\varepsilon}(k^{\varepsilon} + k^{3\varepsilon}) = \mathcal{O}(k^{\frac{1}{2}-\varepsilon})$, Π can be solved in $k^{\mathcal{O}(k^{1-\varepsilon})} n^{\mathcal{O}(1)}$ in square graphs. Taking $\varepsilon = \frac{1}{10}$ leads to the claimed complexity. □

6.4.4 Upper bounding the local radius for contact segment graphs

As announced above, we will show here that contact segment graphs satisfy the requirements of Corollary 6.14, which will have the following algorithmic consequence.

Theorem 6.27. *Any problem $\Pi \in \mathcal{P}$ can be solved in time $2^{\mathcal{O}(k^{7/8} \log(k))} n^{\mathcal{O}(1)}$ in contact segment graphs, even when no representation is given.*

Again, the first requirement of Corollary 6.14 is the approximability of maximum clique. This is Lemma 6.30, for the proof of which we first prove two simple statements.

Lemma 6.28. *Let $n \geq 1$. For every representation of K_n as a contact segment graph there is a point of the plane that belongs to at least $n - 1$ segments.*

Proof. The result is trivial for $n \leq 3$. For $n \geq 4$, let $\mathcal{S}$ a representation of K_n as contact segment. If for all segments, the contact with the other segments is made on a single point, then this contact point is common for all the segments and we get the desired result. Otherwise, as represented in Figure 6.5 let $\mathcal{D}_v$ a segment in contact with the other segments on at least two distinct points $p_a, p_b \in \mathcal{D}_v$ and a, b vertices such that $\mathcal{D}_a$ (respectively $\mathcal{D}_b$) intersects $\mathcal{D}_v$ on p_a (respectively p_b). As they correspond to adjacent vertices, $\mathcal{D}_a$ and $\mathcal{D}_b$ have to intersect: we denote

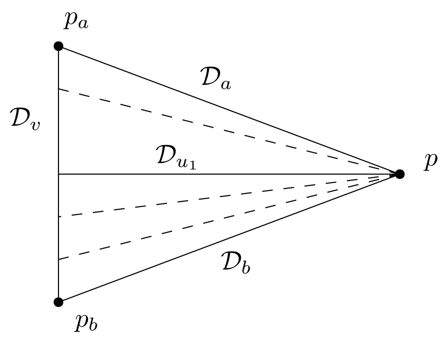

Figure 6.5: The construction used in the proof of Lemma 6.28. The dashed segments represent examples of potential additional segments in the representation of the clique.

p their contact point. By definition of contact segment graphs, $p \notin \mathcal{D}_v$. Let u_1 a vertex distinct from v, a and b. $\mathcal{D}_{u_1}$ is in contact with $\mathcal{D}_v$, $\mathcal{D}_a$ and $\mathcal{D}_b$, this can be done only by containing either p_a, p_b or p. Without loss of generality suppose that $\mathcal{D}_{u_1}$ contains p, and so an endpoint of the segment $\mathcal{D}_{u_1}$ is contained in $\mathcal{D}_v$. For the remaining vertices $u_2, \ldots, u_{n-3}$, we still have that either p, p_a or p_b are contained in their segment, but as it is not possible to cross the segment $\mathcal{D}_{u_1}$, it is always p which is contained in the segment. Therefore, all the segments except $\mathcal{D}_v$ intersect on p, which yields the claimed result. □

Lemma 6.29. *A contact segment graph G with n vertices has $\mathcal{O}(n^2)$ maximal cliques.*

Proof. There are $\mathcal{O}(n^2)$ maximal cliques with at most 2 vertices in G, and for a maximal clique C of size at least $|C| \geq 3$, Lemma 6.28 gives the existence of a point p_C contained in all the segments of the representation of the maximal clique except at most one. This point is necessarily the endpoint of a segment, so p_C can take at most $2n$ distinct values.

Let us now distinguish two cases. If all the segments representing C contain p_C: because C is maximal, it is exactly the segments containing p_C and so is entirely determined by p_C. This gives that there are at most $\mathcal{O}(n)$ maximal clique for this case. The second case is when there is some vertex $v \in C$ such that p_C is not contained in $\mathcal{D}_v$. By denoting $S = \{u \in V(G) : p_C \in \mathcal{D}_u\}$ we have $C = (S \cap N(v)) \cup \{v\}$ as C is maximal. Therefore, this time C is determined by the point p_c and the vertex v not containing p_C. Thus, there are at most $\mathcal{O}(n^2)$ maximal clique for this case. In total we obtain at most $\mathcal{O}(n^2)$ maximal cliques in G. □

Lemma 6.30. *There is a polynomial time algorithm that, given a contact segment graph, returns a maximum clique, even if no representation is provided.*

Proof. By Lemma 6.29 contact segment graphs have a polynomial number of maximal cliques. Those can be enumerated with polynomial delay (for instance using [MU04]) so overall in polynomial time one can enumerate all maximal cliques of a contact segment graph, and return one with the maximum number of vertices. □

Definition 6.31 (interior and non-trivial contact points). *Given a contact segment representation of a graph and a point p in the plane, we denote by $\mathcal{E}^p$ the set of segments whose one endpoint is p. Given a segment s, we denote by*

- *$\mathcal{E}(s)$ the two endpoints of s,*
- *$\mathcal{ICP}(s)$ (for interior contact point) the set of interior points of s that are endpoints of other segments,*
- *$\mathcal{NT}(s)$ (for non-trivial) the subset of those points $p \in \mathcal{ICP}(s)$ for which there exist s_1 and s_2 on the two sides of s such that $p \in \mathcal{E}(s_1) \cap \mathcal{E}(s_2)$, and*
- *$\mathcal{T}(s) = \mathcal{ICP}(s) \setminus \mathcal{NT}(s)$ ($\mathcal{T}$ for trivial).*

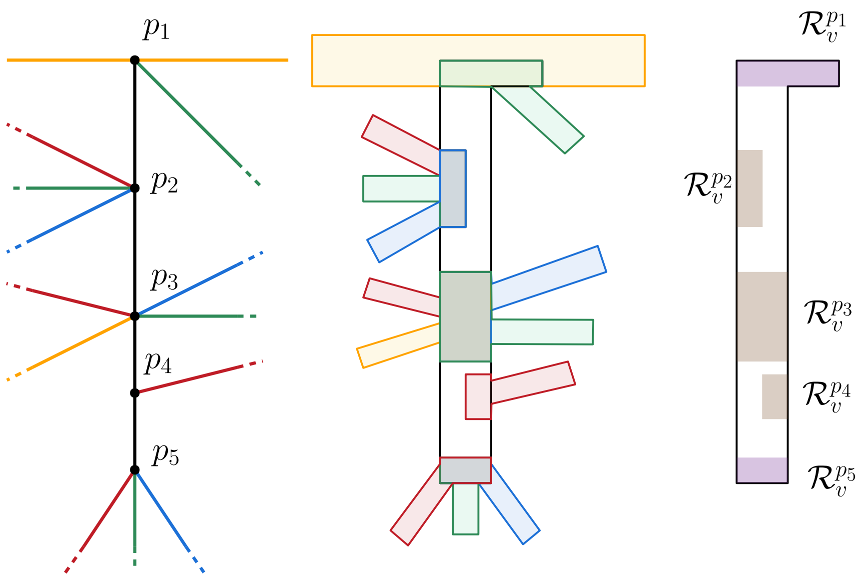

Figure 6.6: Left: a representation $\mathcal{S}_1$ of a contact segment graph. Middle: the thick representation $\mathcal{S}_2$ defined in Lemma 6.34. Right: the slots of the region associated to a vertex v. Here by denoting s the segment representing v we have $\mathcal{E}(s) = \{p_1, p_5\}$, $\mathcal{ICP}(s) = \{p_2, p_3, p_4\}$, $\mathcal{NT}(s) = \{p_3\}$ and $\mathcal{T}(s) = \{p_2, p_4\}$.

Definition 6.32. *Given a contact segment graph G and a representation $\mathcal{S} = \{\mathcal{D}_v : v \in V(G)\}$, for any v we define $\mathrm{N}^\star_{\mathcal{S}}(v) = \{u \in N(v) : \mathcal{E}(\mathcal{D}_u) \cap \mathcal{NT}(v) \neq \emptyset\}$. When $\mathcal{S}$ is clear from context, we use $\mathrm{N}^\star$ instead of $\mathrm{N}^\star_{\mathcal{S}}$.*

Informally, $\mathrm{N}^\star(v)$ is the set of all vertices intersecting v in a non-trivial contact point.

Remark 6.33. *With the same definitions as in Definition 6.32, the following holds:*

- *$\mathrm{N}^\star$ is a 2-occurrences bounded subneighborhood function (direct from the definition).*
- *For any $v \in V(G)$, $|\mathcal{NT}(v)| \leq \mu^{\mathrm{N}^\star}(v)$ (as all segments ending in a point $p \in \mathcal{NT}(s)$ form a clique of size at least 2, and thus contribute to a least 1 on the matching in $G[\mathrm{N}^\star(v)]$).*

Lemma 6.34. *Let G be a contact segment graph. There exists a subneighborhood function $\mathrm{N}^\star$ which is 2-occurrences bounded and such that $\mathrm{lr}(G) = \mathcal{O}(\mu^{\mathrm{N}^\star}(G))$.*

Proof. Let $\mathcal{S}_1$ be a contact segment representation of G, and $\mathrm{N}^\star$ as defined in Definition 6.32. Let us now define a thick representation $\mathcal{S}_2 = \{\mathcal{D}_v : v \in V(G)\}$ of G such that $\mathrm{lr}_{\mathcal{S}_2}(G) = \mathcal{O}\left(\mu^{\mathrm{N}^\star}\right)$. This will imply $\mathrm{lr}(G) \leq \mathrm{lr}_{\mathcal{S}_2}(G) = \mathcal{O}\left(\mu^{\mathrm{N}^\star}(G)\right)$. Informally, we will give to each segment some thickness, and associate special "slots" to draw intersection with other segments (see Figure 6.6).

To any vertex $v \in V(G)$ corresponding to a segment s in $\mathcal{S}_1$, we associate a region $\mathcal{D}_v$, and $|\mathcal{ICP}(s)| + 2$ disjoint and connected subregions (called ***slots***) $\mathcal{R}_v^p$ for $p \in \mathcal{E}(s) \cup \mathcal{ICP}(s)$ such that in particular:

- $\{v, v'\} \in E(G)$ iff $\mathcal{D}_v \cap \mathcal{D}_{v'} \neq \emptyset$ (i.e. $\mathcal{S}_2$ is indeed a representation of G)
- intersections occur within slots: for any vertex $v' \in N(v)$ corresponding to a segment s' in $\mathcal{S}_1$, with $p = s' \cap s$, we have $\mathcal{D}_{v'} \cap \mathcal{D}_v = \mathcal{R}_v^p = \mathcal{R}_{v'}^p$.
- for any $p \in \mathcal{E}(s) \cup \mathcal{T}(s)$, $\mathcal{D}_v \setminus \mathcal{R}_v^p$ is connected,
- for any $p \in \mathcal{NT}(s)$, $\mathcal{D}_v \setminus \mathcal{R}_v^p$ is a set of two connected regions.

This completes the description of $\mathcal{S}_2$. For any $v \in V(G)$ corresponding to a segment s in $\mathcal{S}_1$, we have $\mathrm{lr}_{\mathcal{S}_2}(v) = \mathcal{O}(|\mathcal{NT}(s)|)$ (as for any $p \in \mathcal{T}(s)$, $\mathcal{D}_v \setminus \mathcal{R}_v^p$ is connected), and thus $\mathrm{lr}_{\mathcal{S}_2}(v) = \mathcal{O}(\mu^{\mathrm{N}^\star}(v)) = \mathcal{O}(\mu^{\mathrm{N}^\star}(G))$ by Remark 6.33.

□

We are now ready to prove the main algorithmic result of the section.

Proof of Theorem 6.27. Let $\Pi \in \mathcal{P}$. According to Lemma 6.30 and Lemma 6.34, we can apply Corollary 6.14 with $c_1 = 0$, and $P(x, y) = y$. This implies that for any ε such that $k^{3\varepsilon} = \mathcal{O}(k^{\frac{1}{2}-\varepsilon})$, Π can be solved in time $k^{\mathcal{O}(k^{1-\varepsilon})} n^{\mathcal{O}(1)}$ in contact segment. Taking $\varepsilon = \frac{1}{8}$ leads to the claimed complexity. $\square$

Chapter 7
Hitting cycles

Contents

There are two natural directions of research on subexponential FPT algorithms for FVS, either improving the running times in the classes where an algorithm is already known or extending the tractability horizon of FVS by providing more general settings where subexponential FPT algorithms exist. In this chapter, we are interested by the second direction, that can be summarized by the following question.

Question 7.1. *What are the most general graph classes where FVS admits a subexponential FPT algorithm?*

More precisely we will try to determine what are the geometric graph classes admitting such algorithms. Coming back to the lower bounds presented in Chapter 5, when restricted to unit disk graphs or 2-DIR graphs FVS is NP-complete and does not admit a $2^{o\left(\sqrt{n}\right)}$-time algorithm under ETH. For the positive results presented in Chapter 3, we have for example algorithms solving FEEDBACK VERTEX SET in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$ for string graph [BR19], $n^{\widetilde{\mathcal{O}}(\sqrt{k})}$ for outerstring graphs and subexponential FPT time algorithms for:

- unit disk graphs, with an ETH-tight running time [AO21],
- string graphs of bounded edge degree [BT22],
- and disk graphs Theorem 3.15.

We can add to this non-exhaustive list the subexponential FPT algorithms from the previous chapter, obtained using the ASQGM framework, for square graphs and contact segment graphs. Some of these algorithms are robust. Because the recognition problem is difficult for most of the classes discussed above, robustness is a substantial advantage.

Organization of the chapter. The first section will give sufficient conditions for a graph class to admit a robust subexponential FPT algorithm solving the FVS problem. We will then show that pseudo-disk graphs and s-string graphs, satisfy those conditions and so admit a subexponential FPT algorithm. The second section will explain how, in the case of pseudo-disk graphs, it is possible to obtain robust algorithm with a better running time (for both FEEDBACK VERTEX SET and TRIANGLE HITTING). Finally the last section of this chapter will focus on the pseudo-disk graphs once again, and improve even more the running time for FEEDBACK VERTEX SET, at the cost of needing a geometric representation.

7.1 Pushing the barriers for solving FVS in subexponential FPT time

Toward answering Question 7.1, we identify sufficient conditions for a graph class (then said to be ***nice***) to admit a subexponential parameterized algorithm for FVS. As we will see later, these conditions are satisfied by several natural graph classes, some of which were not known to admit a subexponential parameterized algorithm prior to this work.

Let us provide some intuition behind the conditions we require for a ***nice*** graph class. We discuss here the similarities between these conditions and classical studied properties, while the reasons why these conditions help to get a subexponential parameterized algorithm for FVS are examined in Subsection 7.1.1. A starting point is to review known results about string graphs, which constitute a good candidate to answer the previous question. In particular, the bound for $K_{t,t}$-free string graphs on the number of edges in Theorem 2.11 and the treewidth in Theorem 2.12 are interesting in our case, as a simple folklore branching allows us to reduce the problem to the case where the instance (G, k) of FVS is $K_{r,r}$-free for $r = \lceil k^{\varepsilon} \rceil$ and any $\varepsilon < 1$ (see Subsection 7.2.1). Thus, among the conditions required for a graph class to be nice, two of them correspond to a relaxed version of the previous theorems. Our last main condition is related to neighborhood complexity, already presented in Chapter 4. Let us now proceed to the formal definitions. First, we restate Definition 4.13, which was for subgraph-neighborhood complexity, but this time we consider trees instead of subgraphs.

Definition 7.2. *We say that a graph class $\mathcal{G}$ has* **bounded tree-neighborhood complexity** *(with parameters α, f_1, f_2) if there exists a value α and two polynomial functions f_1, f_2 such that the following conditions hold. For every r, every $K_{r,r}$-free graph $G \in \mathcal{G}$, every set $A \subseteq V(G)$ and every family $\mathcal{T}$ of disjoint non-adjacent[1] trees of $G - A$:*

1. *$|\{N_A(T) : T \in \mathcal{T}\}| \leq f_1(r)|A|^{\alpha}$, where $N_A(T)$ denotes the neighbors of the vertices of T in A, and*

2. *$|\{N_A(T) : T \in \mathcal{T}\}| \leq f_2(r, p, m)|A|$, where p and m denote the maximum over all $T \in \mathcal{T}$ of $|N_A(T)|$ and $|T|$ respectively.*

Remark 7.3. *It is easy to see that having bounded subgraph-neighborhood complexity implies having bounded tree-neighborhood complexity with the same parameters. So Lemma 4.14 and Lemma 4.15, which bound the subgraph-neighborhood complexity in pseudo-disk graphs and s-string graphs respectively, apply for the tree-neighborhood complexity.*

We are now ready to define the considered graph classes.

Definition 7.4. *We say that an hereditary graph class $\mathcal{G}$ is* **nice** *(for parameters $\alpha, f_1, f_2, \delta, f, d$) if all the following conditions hold:*

(a) $\mathcal{G}$ is stable by contraction of adjacent degree-two vertices that do not belong to a triangle.

(b) $\mathcal{G}$ has bounded tree neighborhood complexity (for some parameters α, f_1, f_2).

[1]Two vertex subsets are ***non-adjacent*** in a graph if there is no edge from one to the other.

(c) *There exist $\delta < 1$ and a constant f_r that depends polynomially in r such that for any $K_{r,r}$-free graph $G \in \mathcal{G}$, $\mathbf{tw}(G) = \mathcal{O}(f_r \cdot n^{\delta})$.*

(d) *There is a constant d_r that depends polynomially in r such that for any $K_{r,r}$-free graph $G \in \mathcal{G}$, $|E(G)| \leq d_r \cdot |V(G)|$. Without loss of generality we will assume $d_r \geq r$.*

Our main result is then the following.

Theorem 7.5. *For every nice hereditary graph class $\mathcal{G}$ there is a constant $\eta < 1$ such that FVS can be solved for $\mathcal{G}$ in time $2^{\mathcal{O}(k^{\eta})} \cdot n^{\mathcal{O}(1)}$ by a robust algorithm.*

Actually, we provide a single generic algorithm for all nice classes and the parameters of the class (in the definition of nice) appear in the complexity analysis and are used to define η. The techniques used to prove the above result are discussed in Subsection 7.1.1. For the time being, let us focus on consequences. As hinted above, being nice is a natural property shared by several well-studied classes of graphs. In particular, we show that it is the case for s-string graphs and pseudo-disk graphs, hence we have the following applications.

Corollary 7.6. *There exists $\eta < 1$, such that for all s there is a robust parameterized subexponential algorithm solving FVS in time $2^{\widetilde{\mathcal{O}}(s^{\mathcal{O}(1)} k^{\eta})} n^{\mathcal{O}(1)}$ for n-vertex s-string graphs.*

Corollary 7.7. *There exists $\eta < 1$, such that there is a robust parameterized subexponential algorithm solving FVS in time $2^{\mathcal{O}(k^{\eta})} n^{\mathcal{O}(1)}$ for n-vertex pseudo-disk graphs.*

Observe that the two corollaries above encompass a wide range of classes of geometric intersection graphs for which subexponential parameterized algorithms have been given in previous works such as planar graphs, map graphs, unit disk graphs, disk graphs, or more generally pseudo-disk graphs, and string graphs of bounded edge-degree. In this sense, our main result unifies the previous algorithms.

Also, it captures new natural classes such as segment graphs, or more generally s-string graphs, where previous tools were not sufficient (as discussed in Subsection 7.1.1). We point out that before this work, the existence of subexponential parameterized algorithms for FVS was open even for the very restricted class of 2-DIR graphs.

Notations

For $v \in V(G)$ and $B \subseteq V(G)$, we denote $N_B(v) = N(v) \cap B$ and for $A \subseteq V(G)$ we denote $N_B(A) = N(A) \cap B$, with the additional notation $d_B(A) = |N_B(A)|$. Two disjoint vertex subsets or subgraphs Z, Z' of a graph G are said to be ***non-adjacent*** (in G) if there is no edge in G with one endpoint in Z and the other in Z'.

7.1.1 Our techniques

Why bidimensionality fails and differences with classes of “fat” objects

Even if our goal is to abstract from a specific graph class, let us consider in this section the class of 2-DIR graphs, corresponding to the intersection graphs of vertical or horizontal segments in the plane. As these objects are non “fat” and can cross (unlike pseudo-disks), this class constitutes a good candidate to exemplify the difficulties.

A common approach is as follows. Given an instance (G, k) of FVS, we compute first in polynomial time a 2-approximation, implying that we either detect a No-instance or define a set M with $|M| \leq 2k$ and such that $G - M$ is a forest. The goal is then to reduce, using kernelization or subexponential branching rules, to equivalent instances (G', k') with small treewidth, that is $\mathbf{tw}(G') = \mathcal{O}(k^{1-\varepsilon})$. As FVS can be solved in $2^{\widetilde{\mathcal{O}}(\mathbf{tw}(G'))} n^{\mathcal{O}(1)}$ using a classical dynamic programming approach, we get a subexponential parameterized algorithm. Thus, one has to find a way to destroy in G the obstructions preventing a small treewidth. A first type of obstructions is K_r and $K_{r,r}$, which are easy to handle as there are folklore subexponential branchings when $r = k^{\varepsilon}$. Now, one can see the hard part is destroying $K_{r,r}$ hidden (as a minor for example) (see Figure 7.1).

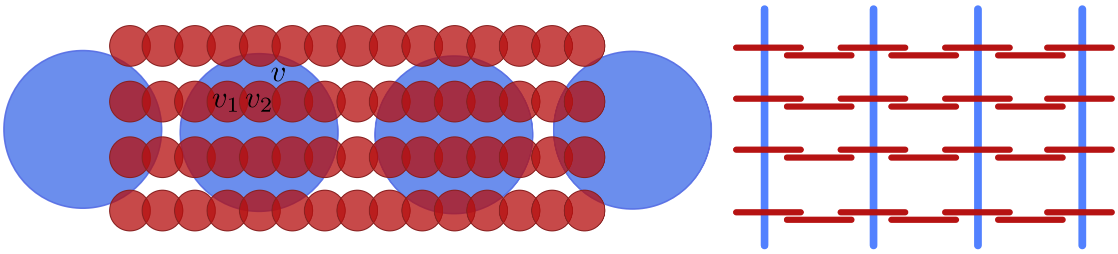

Figure 7.1: Example of a $K_{r,r}$ contained as a minor for $r = 4$ in a disk graph (left) and a 2-DIR graph (right). In the case of disk graphs, v has a matching of size $r - 2$ in its neighborhood, forming a bundle, which can be exploited to branch. The set M is depicted in blue. For the 2-DIR graph, the vertices of the long paths are represented by segments with small variations in their height and not intersecting for better clarity, but are in fact on the same level and intersecting.

A point that seems crucial to us is the following. In intersection graphs of “fat objects” (like disks, squares, or pseudo-disks more generally), the “locally nonplanar structure” when an object (vertex v in Figure 7.1) is “traversed” (by v_1, v_2 in Figure 7.1) comes to the price of an edge ($\{v_1, v_2\}$) in the neighborhood of v. Thus, the presence of a large $K_{r,r}$ as a minor implies that a large matching E_v (of size $\Omega(r)$) will appear in the neighborhood of a vertex v. However, as the bundle $G[\{v\} \cup E_v]$ contains r triangles pairwise intersecting on exactly one vertex v, the set $\{v\} \cup E_v$ is a good structure to perform a subexponential branching for FVS. Indeed, [LPS+22] proposed a "virtual branching" to handle this structure by either taking v in the solution or absorbing E_v in M, implying then that the parameter virtually decreases by $|E_v|$ as a solution which does not contain v has to hit all these edges, even if we cannot branch to determine which are exactly the vertices in the solution.

Once no more virtual branching is possible on large bundles, they obtain by some additional specialized techniques that any vertex in M is such that $N_{V(G)\setminus M}(v)$ is an independent set. Then, it is proved ([LPS+22], Corollary 1.1) that in a disk graph where for any $v \in M$, $N_{V(G)\setminus M}(v)$ is an independent set, and where there does not exist a vertex in $V(G) \setminus M$ whose neighborhood is contained in M, then $\mathbf{tw}(G) = \mathcal{O}\left(\sqrt{|M|}\omega(G)^{2.5}\right)$, where $\omega(G)$ denotes the maximum size of a clique in G. This no longer holds for 2-DIR graphs: the family of pairs (G, M) depicted on the right of Figure 7.1 is indeed a counter-example as they respect the conditions, have $\omega(G) = 2$, but $\mathbf{tw}(G) = \Omega(|M|)$.

More generally, the role of the size of a matching in the neighborhood was studied in Chapter 6 which shows how subexponential parameterized algorithms can be obtained for graph classes having the "almost square grid minor property" (ASQGM), corresponding informally[2] to $\mathbf{tw}(G) = \mathcal{O}(\omega(G)^{\mathcal{O}(1)}\mu_N(G)\boxplus(G))$ where $\mu_N(G)$ is the maximum size of a matching in a neighborhood of a vertex, and $\boxplus(G)$ is the largest size of a grid contained as a minor in G. The previous counter example shows that 2-DIR does not have the ASQGM property, implying that we need another approach to handle them.

7.2 A simpler case study: when trees are only paths

To simplify the arguments, but still understand why properties in Definition 7.4 of a nice graph class are needed, let us assume that the forest $G - M$ only contains paths $(P_i)_i$. Let us describe the algorithm outlined in Algorithm 1. This case remains challenging as a large $K_{r,r}$ can still be hidden as a minor (as in Figure 7.1), and we need to destroy it in order to reduce the treewidth of the graph. To keep notations simple, we use the notation poly(.) to denote a polynomial dependency on the parameter, and thus we do not try to compute tight formulas depending on the polynomial f_1, f_2, f, d given in the definition of a nice class. We assume that we performed folklore branching and that we are left with a $K_{r,r}$-free graph G for $r = k^{\varepsilon}$.

[2]In the correct definition $\mu_N(G)$ is replaced by a slightly more technical parameter.

Algorithm 1 $A(G, k, M, \mathcal{H})$

The algorithm for a nice graph class, if $G - M$ is a forest of paths.

Input: $(G, k, M, \mathcal{H})$ an instance of $(r, \mathcal{G})$-Ann-FVS.

if (one of Rule (KR$_1$), Rule (KR$_2$), Rule (KR$_3$), Rule (KR$_4$), or Rule (KR$_5$) applies on $(G, k, M, \mathcal{H})$) **then**
    Apply the first possible rule to obtain $(G', k', M', \mathcal{H}')$ and **return** $A(G', k', M', \mathcal{H}')$
end if
Partition each path P_i into several 'large degree subpaths' (collected in $\mathcal{T}^+$, those are such that $t \le d_M(P) \le 2t$), and at most one 'small degree subpath' (collected in $\mathcal{T}^-$, it is such that $d_M(P) < t$, with $t = 2d_r$)
if $\mathcal{T}^+$ is too large, there is a $\widetilde{K_{t,t}}$ with a set $X \subseteq M$ and $(T_i)_{1 \le i \le t} \subseteq \mathcal{T}^+$ **then**
    Apply the branching Rule (BR$_2$) on this $\widetilde{K_{t,t}}$, generating a set $\mathcal{C}$ of instances
    return $\bigvee_{(G', k', M', \mathcal{H}') \in \mathcal{C}} A(G', k', M', \mathcal{H}')$
end if
// Here $|V(G)|$ is small implying $\mathbf{tw}(G) = \mathcal{O}(k^{1/\delta - \varepsilon'})$
Solve the instance using dynamic programming (Theorem 7.42)

By item (c) of the definition of a nice graph class (Definition 7.4), we have that $\mathbf{tw}(G) = \mathcal{O}(f_r \cdot n^\delta)$. Thus, our goal is to reduce $|V(G)|$ to $\mathcal{O}(k^{1/\delta - \varepsilon'})$ for some ε'. A first obvious rule is to iteratively contract edges of the P_i's whose endpoints have no neighbors in M. (This corresponds to rules (KR$_1$) and (KR$_2$) in Algorithm 2 and in the following). This explains the property of item (a) of the definition of nice.

Let us now explain how the property of item (b) (bounded tree neighborhood complexity) allows to obtain the following "degree-related size property": for any subpath P of a P_i, $|P| \le \mathsf{poly}(r)(d_M(P)^{\alpha+1})$. Define an independent set $\mathcal{T}$ of size $|P|/2$ by picking every second vertex in P. According to the definition of bounded tree neighborhood complexity (item 1) with $A = N_M(P)$, we get $|\{N_M(v),\ v \in \mathcal{T}\}| \le f_1(r)(d_M(P))^\alpha$. Thus, if $|P| \ge x \cdot f_1(r)(d_M(P))^\alpha$, we found x vertices in P having the same neighborhood M' in M, and thus a vertex $u \in M'$ adjacent to x vertices in P. If x is large enough (about $d_M(P)$), it is always optimal to force u in the solution set. This leads to a kernelization rule (corresponding to (KR$_5$)): If there exists a vertex $u \in M$ adjacent to x vertices in P, for $x \approx d_M(P)$, then it is optimal to force u in the solution. Thus, the kernelization rule checks for all $u \in M$ and all (the $\mathcal{O}(n^2)$ many) paths P, if $d_P(u)$ is large. To sum it up, after applying this rule, for any subpath P of a P_i, if we have $d_M(P) = \mathsf{poly}(r)$, then $|P| = \mathsf{poly}(r)$.

Before the next step we need to apply the following "large degree rule" (corresponding to (KR$_3$)): if there is a vertex v in a P_i such that $d_M(v) > t$, then add v to M. One can prove that by taking $t = 2d_r$, with $d_r = \mathsf{poly}(r)$ the constant defined in item (d) of the definition of a nice class, M does not grow too much after applying this rule exhaustively: by denoting $A \subseteq M$ the set of vertices already in M before applying this rule, we always have $|A| = \mathsf{poly}(r)|M \setminus A|$. This claim will be discussed in Figure 7.2.

Observe that at this stage we may still have a large $K_{r,r}$ as minor, with for example graphs as in Figure 7.1 where no rule applies. It remains to define a crucial rule to destroy these $K_{r,r}$-minors. Let us now present the analog of the `Partitionning-Algorithm` of Lemma 7.2.1 that partitions the $P_i's$ as follows. For any connected component P_i in $G - M$, we start (see Figure 7.2) from an endpoint of P_i and collect greedily vertices until we find a subpath P_i^1 such that $d_M(P_i^1) \geq t$, or that there is no more vertices in P_i. If $d_M(P_i^1) \geq t$, then restart a new path starting from the next vertex to create P_i^2, and so on. This defines a partition $P_i = \bigcup_{\ell \in [x(P_i)]}(P_i^\ell)$, where $d_M(P_i^\ell) \geq t$ for any $\ell \in [1, x(P_i) - 1]$ and no lower bound for $d_M\left(P_i^{x(P_i)}\right)$. As we applied the large degree rule (KR$_3$), we also know that $d_M(P_i^\ell) \leq 2t$ for any $\ell \in [1, x(P_i)]$, because collecting at each step a new vertex in the path P_i^ℓ can increase $d_M(P_i^\ell)$ by at most t. This implies, using the degree-related size property introduced above, that $|P_i^\ell| \leq \mathsf{poly}(r)$ for any $\ell \in [1, x(P_i)]$. Let us denote $\mathcal{T}^+$ the set of P_i^ℓ such that $d_M(P_i^\ell) \geq t$ the "large-degree subpaths". Observe that the last considered subpath $P_i^{x(P_i)}$ of each connected component P_i (on the right of each path of $G - M$ in Figure 7.2) may have $d_M\left(P_i^{x(P_i)}\right) < t$ as there were no more vertices to complete it, hence it is not contained in $\mathcal{T}^+$. We denote $\mathcal{T}^-$ those remaining "small-degree subpaths".

Let us now explain how the bounded tree neighborhood complexity property (item 2) allows to obtain the following "small number of large-degree subpaths" property. By removing half of the $\mathcal{T}^+$'s, we can get a set $\mathcal{T}^{+'}$ of non-adjacent trees (meaning with no edge between the $T_i's$) such that $|\mathcal{T}^{+'}| \geq \frac{1}{2}|\mathcal{T}^+|$. We can then apply the bounded tree neighborhood complexity property with $A = M$, $\mathcal{T} = \mathcal{T}^{+'}$ and $p = m = \mathsf{poly}(r)$ to obtain $|\{N_M(T),\ T \in \mathcal{T}^{+'}\}| \leq \mathsf{poly}(r)|M|$. Thus, if $|\mathcal{T}^{+'}| \geq x \cdot \mathsf{poly}(r)|M|$, we found x large-degree subpaths (denoted T_i') having the same neighborhood X' in M with $|X'| \geq t$. By choosing $x = t$ and considering $X \subseteq X'$ with $|X| = t$, we found a structure that we call a $\widetilde{K_{t,t}}$ (see Figure 7.3), formally defined as a pair $(X, (T_i)_{1 \leq i \leq t})$ where

- $X \subseteq M$ has size t,
- $(T_i)_i$ a family of t vertex-disjoint non-adjacent subtrees (paths here) of $G - M$ such that for all $1 \leq i \leq t$, $X \subseteq N_M(T_i)$, and

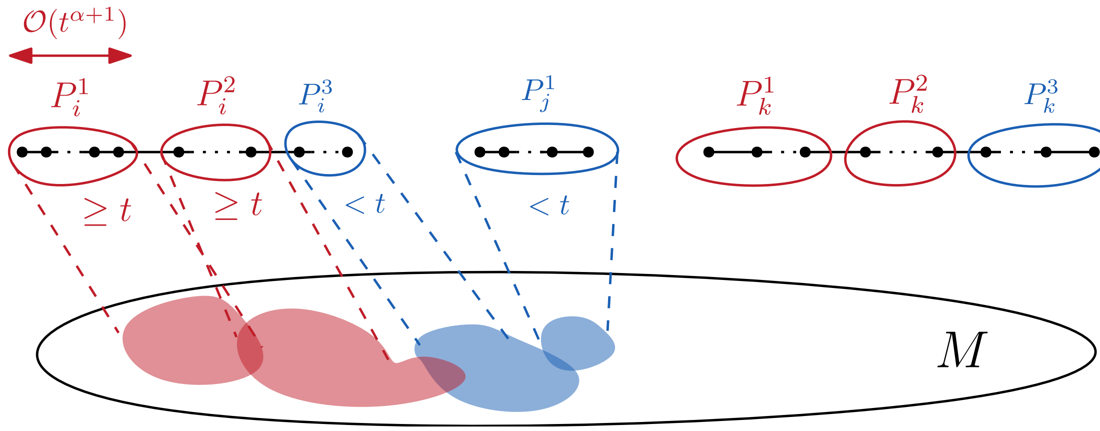

Figure 7.2: Example of partition where P_i is partitioned into $x(P_i) = 3$ subpaths, with P_i^1 and P_i^2 in $\mathcal{T}^+$ and $P_i^3 \in \mathcal{T}^-$.

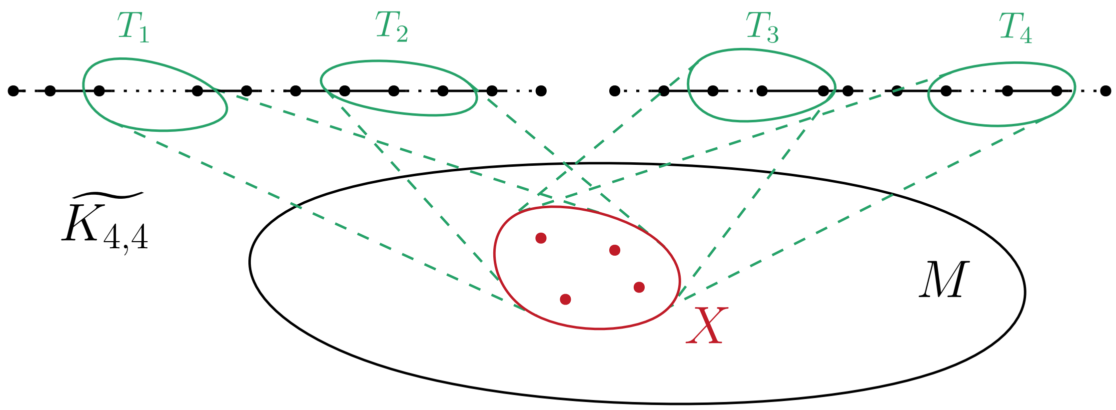

Figure 7.3: Example of $\widetilde{K_{t,t}}$ for $t = 4$. Here we have $X \subseteq N_M(T_i)$ for $1 \leq i \leq 4$.

- for any T_i, $|T_i| \leq \mathsf{poly}(r)$.

Now, inspired by the "virtual branching rule" for a triangle bundle, we introduce a branching rule (corresponding to (BR$_2$)) that either deletes almost all vertices in X, or adds to M a subset of $t-1$ of the $T_i's$. The complexity behind this rule is fine, as in the second branch, the parameter virtually decreases by $t-1$, and M grows by $(t-1)\max_i\{|T_i|\} = \mathsf{poly}(r)$. This explains how we deal with the $K_{t,t}$-minors.

Finally, if this $\widetilde{K_{t,t}}$ rule cannot be applied, it remains to bound $|V(G) \setminus M|$. Recall that any $P \in \mathcal{T}^+ \cup \mathcal{T}^-$ is such that $|P| \leq \mathsf{poly}(r)$, and thus we only need to bound $|\mathcal{T}^+ \cup \mathcal{T}^-|$. As we cannot apply the previous rule, we know that the number of big paths is small: $|\mathcal{T}^+| \leq \mathsf{poly}(r)|M|$. Now, to bound $|\mathcal{T}^-|$, observe that we can partition $\mathcal{T}^- = \mathcal{T}_1^- \cup \mathcal{T}_2^-$, where

- $\mathcal{T}_1^-$ is the set of small-degree paths P_i^ℓ for some $\ell > 1$ (belonging to the same path P_i than a large-degree path $P_i^{\ell-1}$).
- $\mathcal{T}_2^-$ is the set of small-degree paths which are entire connected components of $G - M$.

As $|\mathcal{T}_1^-| \leq |\mathcal{T}^+|$, it only remains to bound $|\mathcal{T}_2^-|$. Now, we can exploit once again the bounded tree neighborhood complexity property (item 2) to obtain that, if $|\mathcal{T}_2^-| \geq x \cdot \mathsf{poly}(r)|M|$, then we can find x disjoint non-adjacent paths in $\mathcal{T}_2^-$ having the same neighborhood X in M. This case is different from the $\widetilde{K_{t,t}}$ case as X may be arbitrarily small (we only know that $|X| < t$), and thus unlike $\widetilde{K_{t,t}}$ this does not decrease the parameter k by a large amount. However, in this case, paths of $\mathcal{T}_2^-$ are just connected components of $G - M$, and we use a last rule ((KR$_4$)) that identifies paths that can be safely removed. It can be shown that if $x+2$ paths have the same neighborhood in M, of size x, one of the paths is redundant. Hence after applying the rule exhaustively we can assume $|\mathcal{T}_2^-| < x \cdot poly(r)|M| = poly(r)|M|$.

This concludes the sketch of proof for this restricted setting where the connected components of $G - M$ are paths, as we obtain by taking ε small enough $|V(G) \setminus M| \leq \mathsf{poly}(r)|M| \leq \mathsf{poly}(k^\varepsilon)k = \mathcal{O}(k^{2-\varepsilon'})$ $\mathcal{O}(k^{1/\delta-\varepsilon'})$ as required.

Challenges to lift the result from paths to trees

We now consider the real setting where given (G, k) where G is $K_{r,r}$-free for $r = k^\varepsilon$, and given M a feedback vertex set of size at most $2k$, we want to reduce the graph to obtain $|V(G) \setminus M| = \mathcal{O}(2^{1/\delta-\varepsilon'})$. The approach still consists to appropriately partition $G - M$ (called a t-uniform partition).

A first problem when trying to adapt the approach of Section 7.2 is the degree-related size property. Indeed, after the first two sections Lemma 7.2.1 and Corollary 7.2.1, we are now only able to obtain that for any subtree T of $G - M$, $|T| \leq \mathsf{poly}(r)\mu(T)^{\mathcal{O}(\alpha)}$ where $\mu(T) = \max(d_M(T), b_{\overline{M}}(T))$ and $b_{\overline{M}}(T) = |\{v \in T,\ N(v) \nsubseteq M \cup T\}|$ is the size of the "border of T". Observe that $b_{\overline{M}}(P)$ is at most 2 for any subpath P of path P_i, whereas $b_{\overline{M}}(T)$ can only be bounded by $|T|$ for a subtree T. Informally, in the path case $|P|$ was only polynomially dependent on $d_M(P)$, and now $|T|$ is also polynomially depends on $b_{\overline{M}}(T)$.

A second problem is the large degree rule. Suppose that this rule no longer applies (meaning that for every $u \in V(G - M)$, we have $d_M(u) \leq t$), and suppose now that because of another rule a vertex $v \in V(G) \setminus M$ is added to M, denoting $M' = M \cup \{v\}$. Then this can create a new large degree vertex v' with $d_{M'}(v') > t$ (and so $d_M(v') = t$). Then v' would need to be added and the problem may arise again for another vertex v''. This "cascading" can easily be prevented if $G - M$

is a forest of paths: it suffices to apply the rule a first time at the start of the algorithm but with $t' = t - 2$. We then have for each $v \in V(G - M)$ the bound $d_M(v) \leq t - 2$, and we do not need to apply the rule again after as adding vertices to M may increase $d_M(v)$ by at most 2 ensuring the wanted bound $d_M(v) \leq t$ for $v \in V(G - M)$. However, in the case of a tree, we can have in $G - M$ a vertex of arbitrarily large degree, so we cannot apply the same solution. The problem is treated with the help of a technical lemma (that we prove at the end of the proof, see Lemma 7.39) which ensures that throughout the execution of the algorithm, we keep $|M| = \mathsf{poly}(r)k$.

Finally, a third problem is the definition of the partition. As in the case of paths we want to partition $G - M$ into a "t-uniform partition" $\mathcal{T}$, where in particular we have $\mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$, and for any $T \in \mathcal{T}$, $d_M(T) \leq 2t$ and $|T| \leq \mathsf{poly}(r)\mu(T)^{\mathcal{O}(\alpha)}$ (see Definition 7.29 for the complete definition). The greedy approach presented for the case of paths is now more involved, as we have to cut each tree of $G - M$ into subtrees that have small border $b_{\overline{M}}(T)$, as otherwise the previous bound $|T| \leq \mathsf{poly}(r)\mu(T)^{\mathcal{O}(\alpha)}$ becomes useless when $\mu(T)$ is too large.

This partitioning procedure is defined in Lemma 7.2.1. It can either:

- Fail and find a subtree T with $|T| > \mathsf{poly}(r)\mu(T)^{C\alpha}$ for some constant C, implying that our degree-related size rule can be applied.
- Fail and find too many subtrees $T_i \in \mathcal{T}^+$ with large degree, implying that we found a $\widetilde{K_{t,t}}$, and that our $\widetilde{K_{t,t}}$ rule can be applied.
- Produce a t-uniform partition with $|\mathcal{T}^+| \leq \mathsf{poly}(r)|M|$.

The third case is treated in Remark 7.2.1, where we either find another way to apply one more time a reduction rule, or prove that $|V(G) \setminus M| = \mathcal{O}(k^{2-\varepsilon'})$.

7.2.1 FVS in subexponential FPT time in nice graph classes

Preliminary branching to remove $K_{r,r}$

To avoid confusion, we refer to the initial instance with (G_0, k_0). In this section, we use a folklore branching for FVS to remove the large bicliques $K_{r,r}$, where $r = k_0^{\varepsilon}$ with ε to be set later depending on the considered graph class $\mathcal{G}$. Before performing any branching, we compute a 2-approximation of a minimum feedback vertex set of G_0 using Theorem 2.5, and denote it by M_0.

If $|M_0| \leq k_0$ or $|M_0| > 2k_0$ we can conclude that the instance is positive or negative. We thus are left with the case where $k_0 < |M_0| \leq 2k_0$.

Let us now describe a branching algorithm starting with (G_0, k_0, M_0) (with M_0 being a feedback vertex set of G_0 with $|M_0| \leq 2k_0$) and leading to a set of instances $\mathcal{I}$, whose properties are discussed below.

Proofs of the following results are not included as they follow the same arguments as those presented in Subsection 3.1.1.

The algorithm initializes $\mathcal{I}$ to $\{(G_0, k_0, M_0)\}$ and applies the following branching rule to the elements of $\mathcal{I}$ as long as possible.

(BR$_1$) Given an instance $(G, k, M) \in \mathcal{I}$, if G contains a $K_{r,r}$-subgraph with parts A and B, we replace this instance by $2r$ instances $(G - X, k - (r-1), M \setminus X)$, for any set X of size $r - 1$ that is either contained in A, or contained in B.

Remark 7.8. *If the parameter $k - (r-1)$ is negative, then this instance is negative and we remove it from $\mathcal{I}$.*

Lemma 7.9. *There is a $2^{\mathcal{O}(r \log |M|)}|V(G)|^{\mathcal{O}(1)}$-time algorithm that, given an instance $(G, k, M) \in \mathcal{I}$, applies rule (BR$_1$) or correctly concludes that G is $K_{r,r}$-free.*

We summarize the properties obtained after this series of branchings with the following lemma:

Lemma 7.10. *At the end of the preliminary branching, the set $\mathcal{I}$ satisfies the properties:*

1. *The instance (G_0, k_0) is a Yes-instance if and only if $\mathcal{I}$ contains a Yes-instance.*
2. *For any $(G, k, M) \in \mathcal{I}$ the graph G is $K_{r,r}$-free induced subgraph of G_0.*
3. *For any $(G, k, M) \in \mathcal{I}$, M is a feedback vertex set of G with $|M| \leq 2k_0$.*
4. *The total time to generate $\mathcal{I}$ is in $2^{\mathcal{O}(r \log k_0)}|V(G_0)|^{\mathcal{O}(1)}(2r)^{\frac{k_0}{r-1}}$ and $|\mathcal{I}| = \mathcal{O}\left((2r)^{\frac{k_0}{r-1}}\right)$.*

The main recursive algorithm

We now consider each element (G_i, k_i, M_i) of $\mathcal{I}$ as an instance $(G_i, k_i, M_i, \emptyset)$ of the following problem $(r, \mathcal{G})$-Ann-FVS, and our goal now is to solve these instances of $(r, \mathcal{G})$-Ann-FVS using our main recursive Algorithm 2.

Definition 7.11. *Given a nice graph class $\mathcal{G}$ and $r > 3$, the $(r, \mathcal{G})$-Annotated Feedback Vertex Set problem ($(r, \mathcal{G})$-Ann-FVS for short) is the decision problem where given $(G, k, M, \mathcal{H})$ where G is a $K_{r,r}$-free graph of $\mathcal{G}$, k an integer, $M \subseteq V(G)$ a feedback vertex set of G, and $\mathcal{H}$ a family of connected disjoint subsets of M (meaning that for any $H \in \mathcal{H}$, $G[H]$ is connected) such that $|\mathcal{H}| \leq k$, and where the question is whether there exists a feedback vertex set S of G of size at most k that additionally intersects every set of $\mathcal{H}$.*

For the sake of completeness, we provide here the complete pseudo-code of Algorithm 2, even if it uses rules and subroutine which will be defined later. At this stage, we recommend the reader to only read the following sketch, as the following sections will cover in detail the properties we obtain after each step. The sketch of Algorithm 2 is as follows. We first try to apply (line 1) rules (KR_1), (KR_2), (KR_3), and (KR_4), which are like kernelization rules: given the instance $(G, k, M, \mathcal{H})$ we perform a single recursive call on a slightly simpler instance $(G', k', M', \mathcal{H}')$. If none of these first rules apply, the algorithm tries to build a special partition of $G - M$ using the `Partitionning-Algorithm`. If `Partitionning-Algorithm` fails (line 5 or 8) and falls into what we call Case 1 or Case 2, then we apply a kernelization or branching rule. Otherwise, we either apply (KR_4) or (KR_5) (line 15 or line 18), or reach our final point (line 22) where we can prove that $|V(G)|$ is small, implying that $\mathbf{tw}(G) = \mathcal{O}(k^{1-\varepsilon'})$, and solve the instance using a classical DP algorithm.

Algorithm 2 $A(G, k, M, \mathcal{H})$

Input: $(G, k, M, \mathcal{H})$ an instance of $(r, \mathcal{G})$-Ann-FVS.

if (one of Rule (KR$_1$), Rule (KR$_2$), Rule (KR$_3$), or Rule (KR$_4$) applies on $(G, k, M, \mathcal{H})$) **then**
    Apply the first possible Rule to obtain $(G', k', M', \mathcal{H}')$ and **return** $A(G', k', M', \mathcal{H}')$
end if
Apply `Partitionning-Algorithm` (with $t = 2d_r$) of Lemma 7.32 that tries to build $\mathcal{T}$: a t-uniform partition of $G - M$ with $|\mathcal{T}^+| \leq p_3(r, t)|M|$ (where p_3 is defined in Lemma 7.31)
if (procedure fails and falls into Case 1 (output a large subtree T)) **then**
    Apply Rule (KR$_5$) on T to obtain $(G', k', M', \mathcal{H}')$ and **return** $A(G', k', M', \mathcal{H}')$
end if
if (procedure fails and falls into Case 2 (output a $\widetilde{K_{t,t}}$)) **then**
    Apply the branching Rule (BR$_2$) on this $\widetilde{K_{t,t}}$, generating a set $\mathcal{C}$ of instances
    return $\bigvee_{(G',k',M',\mathcal{H}') \in \mathcal{C}} A(G', k', M', \mathcal{H}')$
end if
// $\mathcal{T}$ is as required
Let $Z_1(\mathcal{T})$ and $Z_2(\mathcal{T})$ as defined in Definition 7.35, and $\tilde{M} = M \cup Z_1(\mathcal{T}) \cup Z_2(\mathcal{T})$
// By Lemma 7.36, Rule (KR$_1$) and Rule (KR$_2$) do not apply on $(G, k, \tilde{M}, \mathcal{H})$
if (Rule (KR$_4$) applies on $(G, k, \tilde{M}, \mathcal{H})$, and finds a subtree T that can be removed) **then**
    return $A(G - T, k, M, \mathcal{H})$
end if
if (Rule (KR$_5$) applies on $(G, k, \tilde{M}, \mathcal{H})$ and a connected component T of $G - \tilde{M}$, and finds a vertex $u \in \tilde{M}$ that can be taken) **then**
    return $A(G - T, k - 1, M \setminus \{u\}, \mathcal{H} - \{u\})$
end if
// $|V(G)| = p_4(r, t)|M|$ by Lemma 7.37 implying $\mathbf{tw}(G) = \mathcal{O}(k^{1-\varepsilon'})$ by Theorem 2.4
return $DP(G, k, M, \mathcal{H})$ // Solves the instance using Theorem 7.42

Kernelization rules

Here we provide the four kernelization rules (KR$_1$), (KR$_2$), (KR$_3$), and (KR$_4$) that Algorithm 2 tries to apply at Line 1. Each of these rules takes as input an instance $(G, k, M, \mathcal{H})$ of $(r, \mathcal{G})$-ANN-FVS and outputs a single instance $(G', k', M', \mathcal{H}')$. Such a rule is said to be *safe* if:

- For an instance of $(r, \mathcal{G})$-ANN-FVS the rule returns an instance of $(r, \mathcal{G})$-ANN-FVS (in particular the graph of the output instance is a $K_{r,r}$-free graph in $\mathcal{G}$), and
- the input instance is a Yes-instance if and only if the output instance is a Yes-instance.

Notations. Given any instance $(G, k, M, \mathcal{H})$, recall that every connected component of $G - M$ is a tree. We root each of them at an arbitrary vertex. We define a *subforest* of $G-M$ as a subset $T \subseteq V(G) \setminus M$ and say the set is a *subtree* of $G-M$ if $G[T]$ is a tree. Given a vertex v of a connected component T in $G - M$, we define the subtree T_v of $G - M$ as the connected component of v in $G - M - u$, where u is the parent of v, if any. If v is the root of T, then $T_v = T$. In any case, T_v is rooted at v. Given $X \subseteq V(G)$, we denote $\mathcal{H} - X = \{H \in \mathcal{H} \mid H \cap X = \emptyset\}$. Given a subtree T of $G - M$, $\partial_{\overline{M}}(T)$ denotes the set $\{v \in T,\ N(v) \not\subseteq M \cup T\}$, and $b_{\overline{M}}(T)$ denotes the size of this set. We also denote $\mu(T) = \max(d_M(T), b_{\overline{M}}(T))$.

We start with two basic reduction rules often used to deal with FVS that allow us to get rid of vertices of degree 1 and arbitrarily long paths of vertices of degree 2. Notice that we only apply here the reduction to vertices in $G - M$.

(KR$_1$) Given an instance $(G, k, M, \mathcal{H})$, if there exists a vertex $v \in V(G) \setminus M$ of degree $d(v) \leq 1$, output $(G - \{v\}, k, M, \mathcal{H})$.

(KR$_2$) Given an instance $(G, k, M, \mathcal{H})$, if there exists a path $quvw$ in $V(G) \setminus M$ such that the four vertices have degree 2 in G and $d_M(u) = d_M(v) = 0$, output $(G', k, M, \mathcal{H})$, where G' is the graph obtained from G by contracting the edge uv.

Lemma 7.12. *The rules (KR$_1$) and (KR$_2$) are safe and can be applied in polynomial time.*

Proof. Notice that the vertices considered in the two rules belong to $G - M$, in particular they do not belong to any member of $\mathcal{H}$ (which are subsets of M). Contracting an edge with endpoints of degree two preserves being $K_{r,r}$-free and does not modify the size of the minimum feedback vertex sets. Moreover, the obtained graph is still in $\mathcal{G}$ by the definition of a nice class. The output is then an instance of $(r, \mathcal{G})$-ANN-FVS equivalent to the input. The running time claim is immediate.

□

Remark 7.13. *Observe that we did not use the condition in (KR$_2$) that the endpoints of the considered path have degree at most* 2 *in* G*. However, it will later be used in Lemma 7.39 to bound the size of* M *during the execution of the algorithm.*

The next rule ensures that the vertices outside M have a small neighborhood in M. It increases the size of M, but in a controlled manner as we will see later in Lemma 7.39.

(KR$_3$) Given an instance $(G, k, M, \mathcal{H})$, if there is a vertex $v \in V(G) \setminus M$ such that $d_M(v) \geq t = 2d_r$, with d_r the value defined in Definition 7.4, output $(G, k, M \cup \{v\}, \mathcal{H})$.

It is immediate that rule (KR$_3$) is safe and can be applied in polynomial time.

The fourth kernelization rule deletes unnecessary trees of $G - M$. Consider an instance $(G, k, M, \mathcal{H})$ where none of the previous rules applies. Given a family $\mathcal{T}$ of disjoint subtrees of $G - M$, a tree $T \in \mathcal{T}$ is *redundant* (for $\mathcal{T}$) if for all $v \in M$ such that $d_T(v) \geq 2$, there exists $T' \in \mathcal{T}$ with $T' \neq T$ such that $d_{T'}(v) \geq 2$.

Lemma 7.14. *Consider a set* $X \subseteq M$ *and a set* $\mathcal{T}$ *of subtrees of* $G - M$ *with* $|\mathcal{T}| \geq |X| + 1$*, and such that* $N_M(T) = X$ *for every* $T \in \mathcal{T}$*. Then, there exists a redundant tree for* $\mathcal{T}$*, that can be found in polynomial time.*

Proof. For any $x \in X$, if there is at least one subtree $T \in \mathcal{T}$ such that $d_T(x) \geq 2$, we arbitrarily pick one of them and call it T_x. Now, let T' be one of the remaining trees in $\mathcal{T}$. Such a tree exists as $|\mathcal{T}| \geq |X| + 1$, and there are at most $|X|$ trees of the form T_x, for some $x \in X$. It is immediate from the definition that T' is redundant. □

Recall that we consider a fixed child-parent orientation in the forest $G - M$. In what follows, we say that a subset $F \subseteq V(G - M)$ is a ***downward-closed subtree*** (or ***subforest*** when the set is not necessarily connected) of $G - M$ when for any $v \in F$ and any children u of v, $u \in F$. The fourth kernelization rule is as follows:

(KR$_4$) Given an instance $(G, k, M, \mathcal{H})$, a set $X \subseteq M$ with $|X| \geq 1$ and a set $\mathcal{T}$ of at least $|X| + 2$ disjoint downward-closed subtrees of $G - M$ such that:

- for all $T \in \mathcal{T}$, we have $N_M(T) = X$, and
- either all the roots of the trees in $\mathcal{T}$ have a common parent r, or $\mathcal{T}$ consists only in connected components of $G - M$,

arbitrarily pick one redundant $T \in \mathcal{T}$ (which exists as shown in Lemma 7.14) and output $(G - V(T), k, M, \mathcal{H})$.

Lemma 7.15. *Rule (KR$_4$) is safe and can be applied in polynomial time.*

Proof. The fact that the rule can be applied in polynomial time directly follows from Lemma 7.14. Also, as the graph G' obtained after applying the rule is an induced subgraph of $G \in \mathcal{G}$, we have $G' \in \mathcal{G}$.

Let us now show that the input and output instances are equivalent. As noted above G' is an induced subgraph of G so any feedback vertex set of G is also one for G'. Hence we only have to show that if G' has a feedback vertex set S', then G has a feedback vertex set of size $|S'|$ too. Consider such a set S' and let us transform it (if needed) into a feedback vertex set S of G according to the following cases.

First case $|X \setminus S'| \geq 2$. In this case, we use the following claim.

Claim 7.16. *If $|X \setminus S'| \geq 2$, then S' intersects all the trees of $\mathcal{T}$ except T and at most one other.*

Proof. Let $x_1, x_2 \in X \setminus S'$ be distinct vertices, and suppose by contradiction that there are distinct $T_1, T_2 \in \mathcal{T} \setminus \{T\}$ such that S does not intersect T_1 nor T_2. There exist $a_1 \in N_{T_1}(x_1)$ and $b_1 \in N_{T_1}(x_2)$ as $N_M(T_1) = X$, and because T_1 is connected there exists a path from a_1 to b_1 in T_1 (observe that we may have $a_1 = b_1$). Similarly, we define a_2, b_2 and a path joining them in T_2. Then we have a cycle not hit by S', a contradiction. □

Therefore, if $|X \setminus S'| \geq 2$, there are at least $|X|$ vertices of S' in the trees of $\mathcal{T}$. Let us denote Z this set. The set $S = S' \setminus Z \cup X$ is then a feedback vertex set of G with $|S| \leq |S'|$ as wanted.

Second case there is a unique vertex $x \in X \setminus S'$. Then either S' is a feedback vertex set of G (and so we can take $S = S'$) or there is a cycle in $G - S'$, in which case we consider the following two subcases.

- If $d_T(x) \geq 2$, let $T' \in \mathcal{T}$ with $T' \neq T$ and $d_{T'}(x) \geq 2$ (it exists as T is redundant). Then there is a cycle in $G[T' \cup \{x\}]$, which is hit by the feedback vertex set S' as it is a cycle in G' too. As a result, there is a vertex $v \in T' \cap S'$, and taking $S = (S' \setminus \{v\}) \cup \{x\}$ is a feedback vertex set of G with $|S| \leq |S'|$.

- Otherwise, the graph $G[T \cup \{x\}]$ has no cycle. The cycle C of $G - S'$ necessarily uses some vertices of T and we have $N_M(T) \setminus S' = \{x\}$ and $G[T \cup \{x\}]$ without cycle. This configuration is not possible if T is a connected component of $G - M$, so we are in the case where there exists r a common parent for the roots of the trees in $\mathcal{T}$. The cycle C necessarily contains a path between x and r with inner vertices in T. Note that all the $|X|+1 \geq 2$ trees $T' \in \mathcal{T} \setminus \{T\}$ allow such path between x and r. Therefore, S' contains at least one vertex in some $T' \in \mathcal{T} \setminus \{T\}$ and by taking one such vertex v we can set $S = (S' \setminus \{v\}) \cup \{x\}$ and reach the same conclusion as above.

Third case in the remaining case, where $X \subseteq S'$, we can take $S = S'$.

Hence, the input and output instances are indeed equivalent. □

Observe that Algorithm 2 requires a routine for checking if rule 1 applies, before applying it (with Lemma 7.15).

Lemma 7.17. *Given an instance $(G, k, M, \mathcal{H})$, deciding if Rule (KR$_4$) can be applied, and finding X and $\mathcal{T}$ if it is the case, can be done in polynomial time.*

Proof. We first compute for each connected component T of $G - M$ their neighborhood in M (i.e., $N_M(T)$). If there is a set X such that the set $\mathcal{T}_X$ of connected components T in G_M such that $N_M(T) = X$ is large enough, that is $|\mathcal{T}_X| \geq |X| + 2$, we are done. Finding a family $\mathcal{T}$ for the second version of the rule, where we do not consider connected components of $G - M$ anymore but trees under a common parent r, can be done in a similar way by first testing any $r \in V(G - M)$ as the potential common parent. These operations can be performed in polynomial time. □

Properties of the kernelized instances

The goal of this section is to prove that for an instance $(G, k, M, \mathcal{H})$ for which the kernelization rules do not apply anymore (meaning that we reach Line 4 in Algorithm 2), the size of a subtree T of $G - M$ is strongly related to $\mu(T)$. (Recall that $\mu(T) = \max(d_M(T), b_{\overline{M}}(T))$.)

Remember that because the considered graph class $\mathcal{G}$ is nice, it has bounded tree neighborhood complexity for some parameters α, f_1, f_2. This implies the easy following lemma.

Lemma 7.18. *For every $K_{r,r}$-free $G \in \mathcal{G}$, every set $A \subseteq V(G)$, and every family $\mathcal{T}$ of disjoint non-adjacent subtrees of $G - A$, and every x, if $|\mathcal{T}| \geq x f_1(r)|A|^\alpha$ then there exists $X \subseteq A$ such that at least x subtrees $T \in \mathcal{T}$ satisfy $N_A(T) = X$.*

Moreover, suppose that for every $T \in \mathcal{T}$ we have $d_A(T) \leq p$ and $|T| \leq m$, then if $|\mathcal{T}| \geq x f_2(r, p, m)|A|$, there exists $X \subseteq A$ such that at least x subtrees $T \in \mathcal{T}$ satisfy $N_A(T) = X$.

Proof. The results are obtained from Definition 7.2 by using the pigeonhole principle. □

Recall that given a subtree T of $G - M$, $\partial_{\overline{M}}(T)$ denotes the set $\{v \in V(T),\ N(v) \not\subseteq M \cup V(T)\}$, and $b_{\overline{M}}(T)$ denotes the size of this set.

We are now ready to bound the degree of the subtrees of $G - M$:

Lemma 7.19. *Consider an instance $(G, k, M, \mathcal{H})$ of $(r, \mathcal{G})$-Ann-FVS such that neither the rule (KR$_1$) nor (KR$_4$) applies. For any subtree T of $G - M$ and any vertex v of T, we have*

$$d_T(v) \leq \mathcal{O}(\max(b_{\overline{M}}(T), f_1(r) d_M(T)^{\alpha+1})).$$

Proof. Remember that we chose an arbitrary root for T, and T_v denotes the subtree of T with root v. For bounding the number of neighbors of v in T, it suffices to bound the number of children of v. For this, we first use the fact that the number of children u of v such that $V(T_u) \cap \partial_{\overline{M}}(T) \neq \emptyset$ is bounded by $b_{\overline{M}}(T)$. Let N be the set of remaining children, meaning the set of children u such that $T_u \cap \partial_{\overline{M}}(T) = \emptyset$. Applying Lemma 7.18, with the family $(T_u)_{u \in N}$, the vertex set $N_M(T)$ and $x = d_M(T) + 2$, if $|N| > (d_M(T) + 2) f_1(r) d_M(T)^{\alpha}$ then there would be a set $X \subseteq N_M(T)$ such that $d_M(T) + 2$ vertices $u \in N$ satisfy $d_M(T_u) = X$. Moreover, $X \neq \emptyset$ as otherwise the rule (KR$_1$) would have applied. Then Rule (KR$_4$) would apply. Hence $|N| = \mathcal{O}(f_1(r) d_M(T)^{\alpha+1})$. □

We now show that the previous rules allow us to bound the size of certain types of trees that we define now.

Definition 7.20. *A subtree T of $G - M$ is* **weakly connected** *to M if $G[T \cup \{u\}]$ is acyclic for every u in M (i.e. $d_T(u) \leq 1$ for all $u \in M$). A subtree T of $G - M$, rooted at a vertex v, is* **sharp w.r.t.** *M if T is not weakly connected to M but for every children u of v, T_u is weakly connected to M.*

Lemma 7.21. *Consider an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$) and (KR$_2$) applies. For any subtree T in $G - M$ that is weakly connected to M, we have that $|T| \leq \mu(T)$.*

Proof. Indeed, as Rules (KR$_1$) and (KR$_2$) do not apply, each vertex v of T belongs to (at least) one of the following types:

- $v \in \partial_{\overline{M}}(T)$,
- $d_M(v) \geq 1$,

- $d_T(v) \geq 3$, and
- $d_T(v) = 2$.

Observe that the leaves of T are either of the first type or the second. Let l be the number of leaves of T, there are $b_{\overline{M}}(T)$ vertices of the first type, and at most $d_M(T)$ vertices of the second type, so $l \leq b_{\overline{M}}(T) + d_M(T)$. Moreover, the number of vertices of the third type is at most l. Let us denote T_4 the vertices of the fourth type. Observe that the connected components of $G[T_4]$ are paths of size at most 3 (as otherwise (KR$_2$) would apply). Replacing such a connected component with an edge between the neighbors of the endpoints of the path would result in a tree with vertices $T \setminus T_4$ and whose number of edges (which is at most $|T \setminus T_4| - 1$) is an upper bound on the number of connected components of $G[T_4]$. Therefore, $|T_4| \leq 3|T \setminus T_4| \leq 6l$, and finally T has less than $8l = 8d_M(T) + 8b_{\overline{M}}(T)$ vertices in total. □

We now consider sharp subtrees. Notice that in such a tree, v has bounded degree (by Lemma 7.19), and the subtrees below v have bounded size (by Lemma 7.21). This leads to the following corollary.

Corollary 7.22. *Consider an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$), (KR$_2$) and (KR$_4$) applies. Given a sharp subtree T of $G - M$, we have $|T| = O\left(f_1(r) d_M(T)^{\alpha} \mu(T)^2\right)$.*

Proof. The previous lemmas give

$$\begin{aligned} |T| &= O\left(\mu(T) \max(b_{\overline{M}}(T), f_1(r) d_M(T)^{\alpha+1})\right) \\ &= O\left(f_1(r) d_M(T)^{\alpha} \mu(T)^2\right). \end{aligned}$$

□

Kernelizing when a big tree is found.

When reaching line 4 of Algorithm 2, we call the method `Partitionning-Algorithm` which tries to build a special partition of $G - M$. As we will see later, one output of this procedure is a failure (called case 1) where a "big" (whose size is too large with respect to $d_M(T)$ and $b_{\overline{M}}(T)$) tree T is found in $G - M$. In this section, we explain how we can get rid of such a big tree.

(KR$_5$) Consider an instance $(G, k, M, \mathcal{H})$, with a subtree T of $G - M$ which contains $d_M(T) + b_{\overline{M}}(T)$ vertex disjoint paths of length at least 1 and whose endpoints are all adjacent to some vertex $u \in N_M(T)$. Then output $(G - u, k - 1, M \setminus \{u\}, \mathcal{H} - \{u\})$ if $k \geq 1$ (or a trivial no instance otherwise).

Lemma 7.23. *The rule (KR$_5$) is safe.*

Proof. Again, as the graph G' obtained after applying the rule is an induced subgraph of G, the rule preserves the property of being a $K_{r,r}$-free graph in the considered graph class. For the safeness, it is sufficient to prove that if G has a minimum feedback vertex set of size at most k, then it has a solution S' of size at most k containing u. Let S be a minimum feedback vertex set, and suppose it does not contain u. Then S has at least a vertex in each of the $d_M(T) + b_{\overline{M}}(T)$ considered disjoints paths in T. Therefore, the set $S' = (S \setminus T) \cup N_M(T) \cup \partial_{\overline{M}}(T)$ satisfies $|S'| \leq |S| \leq k$ and contains u. Moreover, this set is a feedback vertex set of G : a cycle in $G - S'$ would contain vertices of T as S is a feedback vertex set of G, but then, as T is a tree, such a cycle would need to intersect the set $N_G(T) = N_M(T) \cup \partial_{\overline{M}}(T) \subseteq S'$, which is a contradiction. □

Lemma 7.24. *There is a multivariate polynomial p_1 with $p_1(x, y) = \mathcal{O}(f_1(x)y^{6+\alpha})$ such that for every instance $(G, k, M, \mathcal{H})$ of $(r, \mathcal{G})$-*Ann-FVS* where none of the rules (KR$_1$), (KR$_2$), and (KR$_4$) applies, and for every subtree T of $G - M$ such that the rule (KR$_5$) does not apply, we have $|T| \leq p_1(r, \mu(T))$.*

Proof. Note that a sharp subtree of T contains a path whose endpoints are distinct and adjacent to a same vertex $u \in M$. Since rule (KR$_5$) does not apply, we have that T contains at most $d_M(T)(d_M(T) + b_{\overline{M}}(T))$ disjoint sharp subtrees.

Let $T^1 = T$ (and remember that the trees of $G - M$ are rooted). Then, iteratively let v_i be a lowest vertex of T^i such that $T^i_{v_i}$ is sharp, and let $T^{i+1} = T^i \setminus T^i_{v_i}$. This process stops when the remaining tree is weakly connected to M or empty. This leads to a partition of T into $c + 1$ disjoint subtrees, where the first c are sharp subtrees, and the last one is weakly connected to M, with $c \leq d_M(T)(d_M(T) + b_{\overline{M}}(T)) = \mathcal{O}(\mu(T)^2)$. Using the bound on the size of the weakly connected and sharp trees from Lemma 7.21 and Corollary 7.22, we get:

$$\begin{aligned}
|T| &= |T^{c+1}| + \sum_{1 \leq i \leq c} |T^i_{v_i}| \\
&= \mathcal{O}\left(\mu(T^{c+1}) + \sum_{1 \leq i \leq c} f_1(r) d_M(T^i_{v_i})^{\alpha}\, \mu(T^i_{v_i})^2\right) \quad \text{by Lemma 7.21, Corollary 7.22} \\
&= \mathcal{O}\left(\mu(T) + f_1(r) d_M(T)^{\alpha} \sum_{1 \leq i \leq c} \mu(T^i_{v_i})^2\right) \\
&= \mathcal{O}\left(\mu(T) + f_1(r) d_M(T)^{\alpha} \left(\sum_{1 \leq i \leq c} \mu(T^i_{v_i})\right)^2\right).
\end{aligned}$$

Note that each vertex of $\partial_{\overline{M}}(T)$ belongs to exactly one of these subtrees, and each subtree $T^i_{v_i}$ creates exactly one vertex in $\partial_{\overline{M}}(T^{i+1})$, and so will contribute to at most one of the $b_{\overline{M}}(T^j_{v_j})$ for $j > i$. Hence, $\sum_i b_{\overline{M}}(T^i_{v_i}) \leq b_{\overline{M}}(T)+c = \mathcal{O}(\mu(T)^2)$. Moreover, observe that $\sum_i d_M(T^i_{v_i}) \leq cd_M(T) = \mathcal{O}(\mu(T)^3)$ so $\sum_i \mu(T^i_{v_i}) = \mathcal{O}(\mu(T)^3)$.

We then obtain $|T| = \mathcal{O}(f_1(r)\,\mu(T)^{6+\alpha})$. □

Corollary 7.25. *Given an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$), (KR$_2$), and (KR$_4$) applies, and given a subtree T of $G - M$ with $|T| > p_1(r, \mu(T))$, Rule (KR$_5$) can be applied in polynomial time on this tree.*

Proof. The fact that Rule (KR$_5$) applies is the contrapositive of Lemma 7.24, and the $d_M(T) + b_{\overline{M}}(T)$ paths needed to apply the rule can be found using the partition used in its proof, which can be computed in polynomial time. □

Branching when there is a $\widetilde{K_{t,t}}$

When reaching line 4 of Algorithm 2, we call the `Partitionning-Algorithm` which tries to build a special partition of $G - M$. As we will see later, one output of this procedure is a failure (called case 2) where a certain dense structure (denoted $\widetilde{K_{t,t}}$, with $t = 2d_r$) is found. In this section, we explain how we branch on such a $\widetilde{K_{t,t}}$.

This new branching rule is a variation of Rule (BR$_1$) that appeared in Subsection 7.2.1: instead of dealing with $K_{r,r}$ subgraphs, we will now consider a set A of t vertices and a set B of small trees such that contracting those trees would result in a $K_{t,t}$-subgraph. More formally:

Definition 7.26. *Given an instance $(G, k, M, \mathcal{H})$ of $(r, \mathcal{G})$-Ann-FVS and an integer t, a $\widetilde{K_{t,t}}$ (of $(G, k, M, \mathcal{H})$) is a pair $(X, (T_i)_{1\leq i\leq t})$ where*

- *$X \subseteq M$ has size t,*
- *$(T_i)_i$ a family of t disjoint non-adjacent subtrees of $G - M$ such that for all $1 \leq i \leq t$, $X \subseteq N_M(T_i)$, and*
- *for any T_i, $|T_i| \leq p_2(r, t)$ with $p_2(r, t) = p_1(r, 2t)$.*

Remark 7.27. *Similarly to a $K_{t,t}$ subgraph, given a $\widetilde{K_{t,t}}$ with parts $(X, (T_i)_{1\leq i\leq t})$, the subgraph of G induced by two vertices of X and two trees of $(T_i)_i$ always contains a cycle. Therefore, a feedback vertex set of G must either contain at least $t - 1$ vertices of X, or intersect each tree of $(T_i)_i$ except at most one.*

The difference between the treatment of $K_{t,t}$ and $\widetilde{K_{t,t}}$ is that, when we are in the branch where a solution must intersect each tree of $(T_i)_i$, we cannot branch in subexponential time to guess which vertex of each T_i is picked, and rather perform the following "virtual branching" trick (introduced in [LPS+22] for the special case where T_i are edges). In this branch, we add all these T_i in M, and maintain a packing $\mathcal{H}$ of such sets T_i. The size of the packing (and thus the growth of M) will be bounded by observing that $|\mathcal{H}|$ is a lower bound on the solution size.

We are now ready to define the last branching rule:

(BR$_2$) Given an instance $(G, k, M, \mathcal{H})$ and a $\widetilde{K_{t,t}}$ with parts $(X, (T_i)_i)$, we generate the following instances:

- if $k \geq t-1$, then for each $v \in X$, denoting $X_{\overline{v}} = X \setminus \{v\}$, output the instance $((G - X_{\overline{v}}), k - (t-1), M \setminus X_{\overline{v}}, \mathcal{H} - X_{\overline{v}})$.
- if $|\mathcal{H}| + (t-1) \leq k$, then for each $1 \leq i \leq t$, denoting $R_i = \bigcup_{j \neq i} T_j$ and $\mathcal{T}_{R_i} = \{T_j : j \neq i\}$, output the instance $\left(G, k, M \cup R_i, \mathcal{H} \cup \mathcal{T}_{R_i}\right)$.

Lemma 7.28. *The Rule (BR$_2$) is safe, and can be applied in polynomial time (assuming a $\widetilde{K_{t,t}}$ is provided).*

Proof. The obtained graph is an induced subgraph of the original graph, so the rule preserves the property of being $K_{r,r}$-free s-string graph of $\mathcal{G}$. The running time bound is straightforward.

The fact that the input and output instances are equivalent comes from Remark 7.27. More formally, suppose that S is a solution to $(G, k, M, \mathcal{H})$. If $|S \cap X| \geq t-1$, then there exists $v \in X$ such that $S \supseteq X_{\overline{v}}$, implying that $((G - X_{\overline{v}}), k - (t-1), M \setminus X_{\overline{v}}, \mathcal{H} - X_{\overline{v}})$ is a Yes-instance. Otherwise, ($|S \cap X| \leq t-2$), there exist two vertices u, v in $X \setminus S$. As for any i, $G[T_i]$ is a tree and $X \subseteq N_M(T_i)$, there cannot be $i \neq j$ such that $S \cap T_i = S \cap T_j = \emptyset$, as otherwise $G[\{u, v\} \cup T_i \cup T_j]$ would contain a cycle not hit by S. This implies that there exists i such that all trees of $\mathcal{T}_{R_i}$ are hit by S. As S must also hit any tree in $\mathcal{H}$ (which are disjoint), it implies that $|\mathcal{H}| + t - 1 \leq k$, and thus the rule generates in particular $\left(G, k, M \cup R_i, \mathcal{H} \cup \mathcal{T}_{R_i}\right)$, which is a Yes-instance. The reverse direction of safeness is immediate. □

Attempting to build a t-uniform partition of $G - M$

In this section, we define the `Partitionning-Algorithm` that is called line 4 when (KR$_1$), (KR$_2$), (KR$_3$), and (KR$_4$) do not apply. The (KR$_4$) will either fail (case 1) and find a "big" subtree T that allows us to apply (KR$_3$) of Corollary 7.2.1, fail (case 2) and find a $\widetilde{K_{t,t}}$ that allows us to apply (BR$_2$) of Corollary 7.2.1, or find a special partition of $G - M$ that we define now.

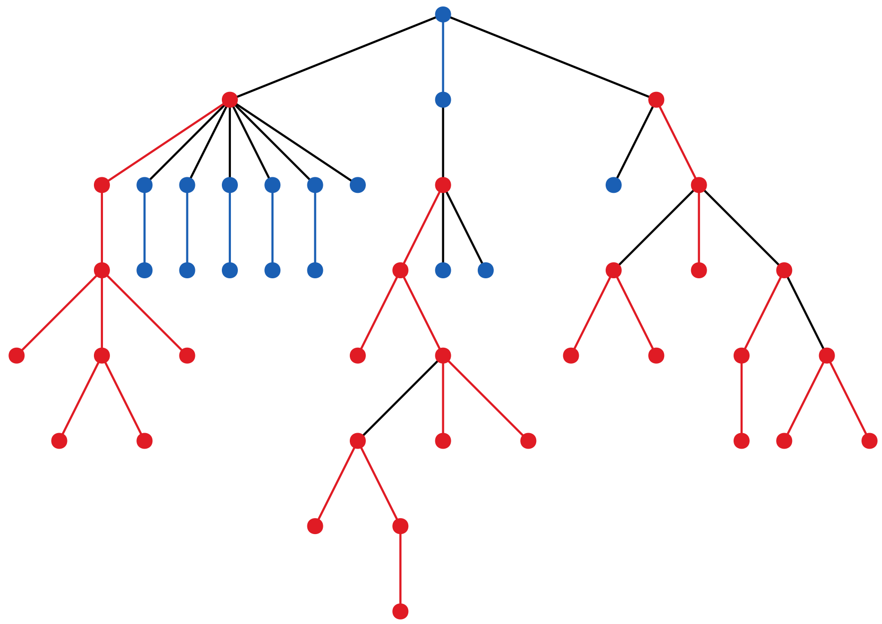

Figure 7.4: Representation of a t-uniform partition of a subtree T of $G - M$. Only the vertices of T are represented here (no vertices of M are included). Vertices of the trees in $\mathcal{T}^+$ (respectively $\mathcal{T}^-$) are represented in red (respectively blue), so as the edges between two vertices in the same tree of $\mathcal{T}^+$ (respectively $\mathcal{T}^-$). Edges between distinct trees of $\mathcal{T}$ are represented in black.

Definition 7.29. *Let $(G, k, M, \mathcal{T})$ be an instance of $(r, \mathcal{G})$-ANN-FVS and t be a positive integer. Let $F \subseteq V(G) \setminus M$ be a downward-closed subforest of $G - M$. A family of trees $\mathcal{T}$ of $G - M$ with vertices in F is a* t**-uniform partition of** *F if:*

1. *Each vertex of F is in exactly one tree of $\mathcal{T}$.*

2. *Every $T \in \mathcal{T}$ has $d_M(T) \leq 2t$, $|T| \leq p_1(r, \mu(T))$ (where p_1 is the polynomial function defined in Lemma 7.24) and $\mathcal{T}$ is partitioned into two subsets, $\mathcal{T}^-$ and $\mathcal{T}^+$ such that a tree $T \in \mathcal{T}$ belongs to $\mathcal{T}^-$ if $d_M(T) < t$, and to $\mathcal{T}^+$ otherwise.*

3. *If a root r of a tree in $\mathcal{T}^-$ has a parent v in $G - M$, then v is the root of a tree in $\mathcal{T}^+$.*

Remark 7.30. *The definition implies that there are no edges between trees of $\mathcal{T}^-$.*

Our goal is to apply the branching rule each time the set $\mathcal{T}^+$ build by `Partitionning-Algorithm` is too large. We show in the next lemma that in this case we can find a $\widetilde{K_{t,t}}$ in polynomial time.

Lemma 7.31. *Consider an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$), (KR$_2$), and (KR$_4$) applies, a downward-closed subforest $F \subseteq V(G-M)$ and a t-uniform partition $\mathcal{T}$ of F with $|\mathcal{T}^+| = p_3(r,t)|M|$ for $p_3(r,t) = 4tf_2(r, 2t, p_2(r,t))$. Then, we can find a $\widetilde{K_{t,t}}$ in polynomial time.*

Proof. Consider $F_{\mathcal{T}^+}$ the forest obtained from $G-M$ by contracting each subtree in $\mathcal{T}$ to a single vertex, and removing the vertices of the trees in $\mathcal{T}^-$. Denoting V_s the set of vertices of $F_{\mathcal{T}^+}$ with degree at most 2, we have $|V_s| \geq |\mathcal{T}^+|/2$ as the number of leaves in a tree is an upper bound on the number of vertices of degree at least than 3. Then using a 2-coloration of the forest we partition the vertices of $F_{\mathcal{T}^+}$ in two independent sets, V_1 and V_2. Without loss of generality, we suppose that $|V_s \cap V_1| \geq |V_s \cap V_2|$. Therefore, denoting $\mathcal{T}'$ the set of trees in $\mathcal{T}^+$ associated to the vertices in $V_s \cap V_1$, we have $|\mathcal{T}'| \geq |V_s|/2 \geq |\mathcal{T}^+|/4$. Now for $T \in \mathcal{T}'$ we want to bound the size of its border. Firstly, for the trees in $\mathcal{T}^+$ adjacent to T, there are at most two of them by construction of $\mathcal{T}'$. Secondly, by item 3 of Definition 7.29 the only vertex in T which can be adjacent to a tree $T' \in \mathcal{T}^-$ is the root of T. Thus, the border of T contains at most 3 vertices overall, and so $b_{\overline{M}}(T) \leq 3$. Moreover, as V_1 is an independent set, there is no edge between two trees of $\mathcal{T}'$. By Definition 7.29, any tree of $T \in \mathcal{T}'$ has size at most $p_1(r, \mu(T))$, but we have $\mu(T) = \max(d_M(T), b_{\overline{M}}(T)) = d_M(T) \leq 2t$ so $|T| \leq p_1(r, 2t) = p_2(r,t)$. Let us now apply the second result of Lemma 7.18 with $x = t$, $A = M$, $p = 2t$ and $m = p_2(r,t)$. As $|\mathcal{T}'| \geq |\mathcal{T}^+|/4 = tf_2(r, 2t, p_2(r,t))|M|$, there exists a subset $X \subseteq M$ and a subset $\mathcal{T}'_X \subseteq \mathcal{T}'$ such that $|\mathcal{T}'_X| = t$, and for all $T \in \mathcal{T}'_X$, we have $N_M(T) = X$. Observe that as $\mathcal{T}' \subseteq \mathcal{T}^+$, we have $|X| \geq t$, so we can take any subset X_0 of X such that $|X_0| = t$ and then $(X_0, \mathcal{T}'_X)$ is a $\widetilde{K_{t,t}}$, as wanted.

(Running time) The set $\mathcal{T}'$ can be computed in polynomial time, and finding the subset X and $\mathcal{T}'_X$ whose existence is proved by Lemma 7.18 can again be done in polynomial time by listing the neighborhoods in M of the trees in $\mathcal{T}'$. ◻

The following lemma ensures the existence of `Partitionning-Algorithm`.

Lemma 7.32. *Let t be an integer. Consider an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$), (KR$_2$), (KR$_4$) applies, and such that $d_M(v) < t$ for every $v \in V(G-M)$. There exists a polynomial time algorithm that given $(G, k, M, \mathcal{H})$ either returns:*

1. *a subtree T of the forest $G-M$ such that Rule (KR$_5$) applies,*
2. *a $\widetilde{K_{t,t}}$ such that Rule (BR$_2$) applies, or*
3. *a t-uniform partition $\mathcal{T}$ of the whole forest $G-M$ with $|\mathcal{T}^+| < p_3(r,t)|M|$ (p_3 being defined in Lemma 7.31).*

Proof. We try to construct a partition of $G - M$ fulfilling item 3, and if we fail we prove that we fall into the case of item 1 or item 2. We denote $p = p_3(r, t)$ in this proof.

The construction proceeds from the deepest leaves to the roots of the trees of $G - M$. For this, we define L_0 as the set of roots of the connected components of $G - M$, and more generally L_i the set of vertices of depth i in the trees of the forest $G - M$. Let h be the depth of the deepest leaves of the highest trees of $G - M$, and $F_i = \bigcup_{j=i}^{h} L_j$ the set of vertices with depth at least i. Observe that we have $F_0 = V(G - M)$.

As a first step, we construct $\mathcal{T}_h$ a t-uniform partition of F_h. Observe that two vertices of the same depth cannot share an edge, so we have $G[F_h]$ an edgeless graph. For constructing the partition we then have no choice but to take each connected component (containing only one vertex) as a tree in $\mathcal{T}$, and more precisely in $\mathcal{T}_h^-$ as for $v \in F_h$ we have $d_M(v) < t$ by hypothesis, and so $\mathcal{T}_h^+ = \emptyset$. $\mathcal{T}_h$ is then a t-uniform partition as there is no edge between two trees of $\mathcal{T}_h$, and each tree has degree in M at most $2t$ as wanted (even t as said above).

Now suppose we are given $\mathcal{T}_i$ a t-uniform partition of F_i with $i \geq 1$, such that $|\mathcal{T}_i^+| < p$. We will now see how to extend this partition to a new partition $\mathcal{T}_{i-1}$ of F_{i-1}, or stop the induction and fall into the case of item 1 or item 2.

Let $\mathcal{T}_{i-1} = \mathcal{T}_i$, $\mathcal{T}_{i-1}^+ = \mathcal{T}_i^+$ and $\mathcal{T}_{i-1}^- = \mathcal{T}_i^-$. Let us now modify $\mathcal{T}_{i-1}$ by considering each $v \in L_{i-1}$ as follows.

For any $v \in L_{i-1}$, let $v_1, \dots, v_l$ be its children, which are in the level L_i. Observe that by the definition of a t-uniform partition of F_i, each of these vertices v_j are the root of some trees $T(v_j)$ in $\mathcal{T}_i$. Informally, we will try to group some of the $T(v_j) \in \mathcal{T}_i^-$ to create a tree in $\mathcal{T}_{i-1}^+$. Let $Y^-(v) = \{j : T(v_j) \in \mathcal{T}_i^-\}$. If $d_M(v \cup \bigcup_{j \in Y^-(v)} T(v_j)) < t$, we set $X(v) = Y^-(v)$. Otherwise, we define $X(v)$ as an inclusion-wise minimal set $Y'(v) \subseteq Y^-(v)$ such that $d_M\left(\{v\} \cup \bigcup_{j \in Y'(v)} T(v_j)\right) \geq t$. Let

$$T(v) = \{v\} \cup \bigcup_{j \in X(v)} T(v_j).$$

If $|T(v)| > p_1(r, \mu(T))$ we stop and output $T(v)$ (we call this "Case 1"). From Corollary 7.25 we get that (KR$_5$) applies as stated in item 1. Let us now assume that $|T(v)| \leq p_1(r, \mu(T))$.

If $d_M(T(v)) < t$, we remove from $\mathcal{T}_{i-1}^-$ all the $T(v_j)$ for $j \in X(v)$, and add $T(v)$ to $\mathcal{T}_{i-1}^-$. We consider v as ***treated***, and proceed with the next vertex $v \in L_{i-1}$.

Otherwise, ($d_M(T(v)) \geq t$), we also remove from $\mathcal{T}_{i-1}^-$ all the $T(v_j)$ for $j \in X(v)$, and we add $T(v)$ to $\mathcal{T}_{i-1}^+$. If $|\mathcal{T}_{i-1}^+| < p$, then we consider v as ***treated***, and proceed with the next vertex $v \in L_{i-1}$. Otherwise, $|\mathcal{T}_{i-1}^+| = p$, then we stop and output $F = \bigcup_{T \in \mathcal{T}_{i-1}} T$ and $\mathcal{T}_{=p} = \mathcal{T}_{i-1}$ (we call this "Case 2", and we will prove that it matches requirements of item 2).

This concludes the description of the algorithm that, given $\mathcal{T}_i$ a t-uniform partition of F_i with $i \geq 1$ such that $|\mathcal{T}_i^+| < p$, either stops in Case 1, stops in Case 2, or compute a set $\mathcal{T}_{i-1}$.

Claim 7.33. *If the algorithm stops in Case 2, then the output matches the requirements of item 2. Otherwise, when the algorithm has processed all the vertices $v \in L_{i-1}$, then $\mathcal{T}_{i-1}$ is a t-uniform partition of F_{i-1} with $|\mathcal{T}_{i-1}^+| < p$.*

Proof. Let us consider as input $\mathcal{T}_i$ a t-uniform partition of F_i with $i \geq 1$ such that $|\mathcal{T}_i^+| < p$, and consider an execution of the algorithm that considers vertices $v \in L_{i-1}$ in an arbitrary order.

Observe first that for any $v \in L_{i-1}$, we have $d_M(T(v)) < 2t$. Indeed, in particular when $d_M(T(v)) \geq t$, we cannot have $X(v) = \emptyset$ as then $d_M(T(v)) = d_M(v) \leq t$. We can thus take any $j \in X(v)$, and we have $d_M(T(v)) \leq d_M\left(\{v\} \cup \bigcup_{k \in X(v) \setminus \{j\}} T(v_k)\right) + d_M\left(T(v_j)\right) < 2t$ by minimality of $X(v)$, and as $T(v_j) \in \mathcal{T}_{i-1}^-$. This directly implies that at any time $\mathcal{T}_{i-1}$ is a t-uniform partition of the treated vertices.

Now, let us prove the following invariant called Π_1. At any point when considering all the $v \in L_{i-1}$ and constructing $\mathcal{T}_{i-1}$, $\mathcal{T}_{i-1}^+$, and $\mathcal{T}_{i-1}^-$:

- If a root r of a tree in $\mathcal{T}_{i-1}^-$ has a parent u in $G - M$, then either $r \in L_{i-1}$ and $u \in L_{i-2}$ (u is not treated), or $r \in L_i$ and $u \in L_{i-1}$ with u being either not treated yet, or the root of a tree in $\mathcal{T}_{i-1}^+$.
- $\mathcal{T}_{i-1}^-$ is a set of disjoint non-adjacent subtrees.

For the first property, let r be the root of a tree $T \in \mathcal{T}_{i-1}^-$ which has a parent u in $G - M$. If $r \in L_j$ for $j > i$, by hypothesis $\mathcal{T}_i$ is a t-uniform partition of F_i, then u is the root of a tree $T \in \mathcal{T}_i^+$, and $T \in \mathcal{T}_{i-1}^+$ as we never remove trees from $\mathcal{T}_i^+$. If $r \in L_{i-1}$, then $u \in L_{i-2}$ and we are done. Finally, assume that $r \in L_i$. If u is not treated then we are done. Otherwise, as u is treated and r is still the root of a tree $T \in \mathcal{T}_{i-1}^-$, it implies that $X(u) \subsetneq Y^-(u)$, and thus that $T(u)$ was added to $\mathcal{T}_{i-1}^+$, satisfying the required condition.

For the second property, first note that the vertices that are in a tree of $\mathcal{T}_{i-1}^-$ are either vertices of L_{i-1}, or these vertices were in a tree of $\mathcal{T}_i^-$. By induction, there is no edge between two distinct trees in $\mathcal{T}_i^-$, we are thus left with the case of an edge uv with $u \in L_i$ and $v \in L_{i-1}$, u being the child of v. If u and v are in different trees of $\mathcal{T}_{i-1}$, it is either because $u \notin Y^-(v)$ or because $X(v) \subset Y^-(v) \setminus \{u\}$. In the former case, u belongs to a tree of $\mathcal{T}_i^+ \subseteq \mathcal{T}_{i-1}^+$, and in the latter case that is $T(v)$ which belongs to $\mathcal{T}_{i-1}^+$.

Let us now finish the proof of Claim 7.33. If the algorithm stops in case 2, then observe that by definition F is downward-closed. Moreover, the output $\mathcal{T}_{i-1}$

is indeed a t-uniform partition of F. Indeed, Property of item 2 of Definition 7.29 holds as, by item 1 of property Π_1, if a root r of a tree in $\mathcal{T}^-_{i-1}$ has a parent u in $G-M$, then $u \in L_{i-2}$ (and thus does not belong to F), or $u \in L_{i-1}$ and u is not treated (so u does not belong to F), or u is the root of a tree in $\mathcal{T}^+_{i-1}$. Thus, in Case 2 we obtain $\mathcal{T}_{i-1}$, a t-uniform partition of F with $\mathcal{T}^+_{i-1} = p$. By Lemma 7.31, we can find in polynomial time a $\widetilde{K_{t,t}}$ and output it.

Finally, if the algorithm does not stop, it remains to prove that $\mathcal{T}_{i-1}$ is a t-uniform partition of F_{i-1} with $|\mathcal{T}^+_{i-1}| < p$. The inequality $|\mathcal{T}^+_{i-1}| < p$ holds as we did not fall into Case 2. Property of item 2 of Definition 7.29 holds because we proved that for any $v \in L_{i-1}$, $d_M(T(v)) < 2t$, and we did not fall into Case 1. Property of item 3 of Definition 7.29 holds as by item 1 of property Π_1, if a root r of a tree in $\mathcal{T}^-_{i-1}$ has a parent u in $G-M$, then $u \in L_{i-2}$ (and thus does not belong to F_{i-1}), or $u \in L_{i-1}$ and is not treated yet (which is not possible as all vertices of L_{i-1} are treated in this case), or u is the root of a tree in $\mathcal{T}^+_{i-1}$. □

Now that Claim 7.33 is proved, Lemma 7.32 directly holds by induction on i. □

Remark 7.34. *Observe that in Line 4, all required condition of Lemma 7.32 to use* `Partitionning-Algorithm` *are fulfilled. In particular as (KR$_3$) does not apply, we get $d_M(v) \leq 2d_r \leq t$ for any $v \in V(G) - M$.*

Final step when $G - M$ admits a t-uniform partition with a small number of large-degree parts

In this section, we consider Line 12 of Algorithm 2 where we found a t-uniform partition $\mathcal{T}$ as required. To bound $|V(G)|$ the algorithm moves some sets $Z_1(\mathcal{T})$ and $Z_2(\mathcal{T})$ (as defined below) to M (to get a slightly larger set $\widetilde{M}$ and tries to apply Rule (KR$_4$) or Rule (KR$_5$) on $(G, k, \widetilde{M}, \mathcal{H})$. If it is not possible, we can prove that $|V(G) - M|$ is small.

Definition 7.35. *Given a t-uniform partition $\mathcal{T}$ of $G - M$, we define the set $Z_1(\mathcal{T})$ as the roots of the trees in $\mathcal{T}^+$, and $Z_2(\mathcal{T})$ as the set of vertices v of $G - M$ having three edge-disjoint paths P_1, P_2, P_3 in $G - M$ linking them to vertices of $Z_1(\mathcal{T})$.*

Lemma 7.36. *Given an instance $(G, k, M, \mathcal{H})$, an integer t, and $\mathcal{T}$ a t-uniform partition of $G - M$, we have $|Z_1(\mathcal{T})| + |Z_2(\mathcal{T})| \leq 2|\mathcal{T}^+|$, and by denoting $\widetilde{M} = M \cup Z_1(\mathcal{T}) \cup Z_2(\mathcal{T})$, we have the following properties:*

- *Rule (KR$_1$) and Rule (KR$_2$) do not apply on $(G, k, \widetilde{M}, \mathcal{H})$.*
- *A connected component T of $G - \widetilde{M}$ satisfies $d_{\widetilde{M}}(T) \leq 2t + 2$.*

Proof. We denote $Z_1 = Z_1(\mathcal{T})$ and $Z_2 = Z_2(\mathcal{T})$. The bound $|Z_1| \leq |\mathcal{T}^+|$ is trivial. To bound $|Z_2|$, let us define the following graph G_Z. The graph G_Z is obtained from $G - M$ by contracting every edge zx with $z \in Z_1 \cup Z_2$ and $x \notin Z_1 \cup Z_2$. Clearly, G_Z is a forest with vertex set $Z_1 \cup Z_2$, and where $d_{G_Z}(v) \geq 3$ for every $v \in Z_2$. Indeed, the three edge disjoint paths leaving v do not completely vanish while constructing G_Z. As the number of vertices of degree at least three in a tree is bounded by the number of leaves, we get $|Z_2| \leq |Z_1| \leq |\mathcal{T}^+|$.

The non applicability of Rule (KR$_1$) and Rule (KR$_2$) on $(G, k, \widetilde{M}, \mathcal{H})$ immediately follows from the non applicability of Rule (KR$_1$) and Rule (KR$_2$) on $(G, k, M, \mathcal{H})$.

It then remains to bound $d_{\widetilde{M}}(T)$ for T a connected component of $G - \widetilde{M}$. Let T be a connected component of $G - \widetilde{M}$. We have $T \subseteq T'$ for some $T' \in \mathcal{T}$, as each edge between trees of $\mathcal{T}$ contains a vertex from $Z_1(\mathcal{T}) \subseteq \widetilde{M}$, and as $\mathcal{T}$ is a t-uniform partition, we get $d_M(T) \leq d_M(T') \leq 2t$. Moreover, notice that $d_{\widetilde{M}}(T) \leq d_M(T) + |\{vz \in E(G) : v \in T \text{ and } z \in Z_1 \cup Z_2\}|$. Thus, assume by contradiction that $d_{\widetilde{M}}(T) \geq 2t + 3$, and thus there are three vertices v_1, v_2, v_3 in T such that $N(v_i) \cap (Z_1 \cup Z_2) \neq \emptyset$. In fact, there are three edges v_1z_1, v_2z_2, and v_3z_3, where the vertices z_1, z_2, z_3 are distinct (but not necessarily the vertices v_1, v_2, v_3). The vertex v at the intersection of the three paths among v_1, v_2, v_3 in T should thus be in Z_2, a contradiction. □

Lemma 7.37. *Given an instance $(G, k, \widetilde{M}, \mathcal{H})$ such that Rules (KR$_1$), (KR$_2$), and (KR$_4$) do not apply, and such that for any T connected component of $G - \widetilde{M}$, Rule (KR$_5$) does not apply on T and $d_{\widetilde{M}}(T) \leq 2t + 2$. Then, $|G - \widetilde{M}| \leq p_4(r, t)|\widetilde{M}|$ with $p_4(r, t) = (2t + 4) f_2(r, 2t + 2, p_1(r, 2t + 2))$.*

Proof. Let $\mathcal{C}$ be the set of connected components of $G - \widetilde{M}$. Let $T \in \mathcal{C}$. As Rule (KR$_1$), (KR$_2$), and (KR$_4$) do not apply, and Rule (KR$_5$) does not apply on T, by Corollary 7.25 we get $|T| \leq p_1(r, \mu(T)) \leq p_1(r, 2t + 2)$. Now, as Rule (KR$_4$) does not apply in particular with $\mathcal{C}$, by the second result of Lemma 7.18 with $A = \widetilde{M}$, $x = 2t + 4$, $p = 2t + 2$ and $m = p_1(r, 2t + 2)$, we get that if $|\mathcal{C}| > x f_2(r, p, m)|\widetilde{M}|$, then we would have x connected components of $\mathcal{C}$ having the same neighborhood X in $\widetilde{M}$. Then we would have $X \neq \emptyset$ as otherwise Rule (KR$_1$) would apply, and as $|X| \leq p = 2t + 2$ and $x \geq |X| + 2$, (KR$_4$) could be applied, a contradiction with the hypothesis. Thus, we get $|G - \widetilde{M}| \leq (2t + 4) f_2(r, 2t + 2, p_1(r, 2t + 2))|\widetilde{M}|$. □

Lemma 7.38. *Let $(G, k, M, \mathcal{H})$ be an input of $(r, \mathcal{G})$-Ann-FVS. If $A(G, k, M, \mathcal{H})$ reaches Line 22, then $|G - M| = \mathcal{O}(p_3(r, t) p_4(r, t)|M|)$.*

Proof. As $A(G, k, M, \mathcal{H})$ reaches Line 22, it implies that `Partitionning-Algorithm` output a t-uniform partition of $G - M$ (where $t = 2d_r$ and $|\mathcal{T}^+| \leq p_3(r, t)|M|$). Let $\widetilde{M} = M \cup Z_1(\mathcal{T}) \cup T_2(\mathcal{T})$. By Lemma 7.36, Rule (KR$_1$) and Rule (KR$_2$) do not apply on $(G, k, \widetilde{M}, \mathcal{H})$. As we did not apply Rule (KR$_4$) (Line 15) or Rule (KR$_5$) (Line 18), and as by Lemma 7.36, $d_{\widetilde{M}}(T) \leq 2t + 2$ for any connected component T of $G - \widetilde{M}$, we can apply Lemma 7.37 to obtain $|G - \widetilde{M}| \leq p_4(r,t)|\widetilde{M}|$. As $|G - M| = |G - \widetilde{M}| + |Z_1(\mathcal{T})| + |Z_2(\mathcal{T})|$, we get $|G - M| = \mathcal{O}(p_4(r,t)(|M| + |Z_1(\mathcal{T})| + |Z_2(\mathcal{T})|))$. By Lemma 7.36 we have $|Z_1(\mathcal{T})| + |Z_2(\mathcal{T})| \leq 2|\mathcal{T}^+|$ and $|\mathcal{T}^+| \leq p_3(r,t)|M|$, we get $|G - M| = \mathcal{O}(p_3(r,t)p_4(r,t)|M|)$. ☐

Let us now bound the size of M in any call to $A(G, k, M, \mathcal{H})$ by providing invariants on the input of Algorithm 2. Informally, $|M|$ can be bounded as follows. Recall that we perform our first call with $A(G_i, k_i, M_i, \emptyset)$ where $(G_i, k_i, M_i) \in \mathcal{I}$ and $M_i \subseteq M_0$, with $|M_0| \leq 2k_0$. Then, if we follow any path in the tree of calls whose root is $A(G_i, k_i, M_i, \emptyset)$, we may add some vertices to M because of (KR$_3$), let us denote A_1 those vertices, or because of (BR$_2$), we denote A_2 those vertices. On a given branch the rule (BR$_2$) adds at most k_0 trees of size at most $p_2(r,t)$ so $|A_2| \leq k_0 p_2(r,t)$. It remains to bound the vertices of A_1. Observe that when a vertex v is added to the set M because of (KR$_3$) (and so to A_1) we have $d_M(v) \geq 2d_r$, and so the number of edges in the graph $G[M]$ increases by at least $2d_r$. We may think that as we have at most $d_r|M|$ edges in $G[M]$, we obtain $2d_r|A_1| \leq d_r|M| \leq d_r(|M_0| + |A_1| + |A_2|)$ which gives a bound on the size of A_1. However, this reasoning is flawed as we need to consider all the vertices added to M, even the ones deleted because of (KR$_5$). Then, in the supergraph of G' containing those vertices, there may be some vertices obtained after contraction of edges by (KR$_2$), which were not of degree 2 in their current supergraph considering deleted vertices. Thus, this supergraph may not be in $\mathcal{G}$, and in fact may even contain a $K_{r,r}$ as a subgraph. Therefore, a more technical analysis is needed to bound the size of M.

Lemma 7.39. *For any input $(G, k, M, \mathcal{H})$ of Algorithm 2, we have $|M| = \mathcal{O}(k_0 p_5(r,t))$ with $p_5(r,t) = 4 + 2p_2(r,t)$.*

Proof. To prove the result, we need to prove some invariant to our problem. For stating this invariant we need to consider additional annotations to our problem. We now consider instances $(G, k, M, \mathcal{H}, A_0, A_1, A_2, X, C, G')$. Informally, A_0 will denote the vertices of $M \cap M_0$, A_1 the vertices of M added by (KR$_3$), A_2 for the ones added to M by rule (BR$_2$), C the set of vertices obtained by contracting at least one edge by (KR$_2$), X the vertices removed by application of (KR$_5$) (so X is a set of vertices outside of $V(G)$), and G' is a supergraph of G with $V(G') = (V(G) \cup X)$ that will be useful for "remembering" the links between the vertices of $V(G)$ and X. More formally, initially for $(G, k, M, \emptyset) \in \mathcal{I}$ the instance is such that $A_1 = M$, $A_2 = A_3 = X = C = \emptyset$ and $G' = G$. The generated instances update the additional annotations in the following manner:

- When removing vertices from $V(G-M)$, so for (KR$_1$) and (KR$_4$), remove the vertices from G' as well and from the sets containing them.
- If we apply (KR$_2$), denoting uv the contracted edge in G and w the vertex added to G for replacing u and v, we do the same operation in G', with an edge between a vertex $q \in V(G) \cup X$ and w if and only if there was an edge between q and u, q and v, or both. Moreover, we add w to the set C.
- If this is (KR$_3$) which is applied, the vertex v added to M is added to A_1 as well.
- For (KR$_5$), the vertex $v \in M$ removed from G is now added to X, and removed from the sets A_i in which it is. G' is not modified.
- For (BR$_2$), in the t branches where we add vertices to M, we add them to A_2 as well now. For the t branches where $t-1$ vertices are removed from M, again they are now added to X, and removed from the sets A_i which contained them.

We now state some invariants on those sets:

Claim 7.40. *1. M is partitioned in A_0, A_1, A_2.*

2. $G'[(V(G) \cup X) \setminus C]$ is a $K_{r,r}$-free graph in $\mathcal{G}$.

3. For $v \in V(G)$, if $v \in C$ or $N_{G'}(v) \cap C \neq \emptyset$ then $d_G(v) \leq 2$.

4. $C \cap A_1 = \emptyset$.

5. For $v \in A_1$, $|(N_{G'}(v) \cap (M \cup X)) \setminus C| \geq 2d_r$.

Proof. The first claim is trivial to verify. For the second, observe that the only modification to this graph is the deletion of vertices, so the result follows by hereditary of $\mathcal{G}$. For each rule, the degree of a vertex does not increase in G at any time (but it can in G'), so it suffices to prove the result when the vertex was just added. If a vertex $v \in V(G)$ is added to C or one of its neighbors it implies that $d_G(v) \leq 2$. The fourth result can be proved by observing that an element of the intersection would have first been in C and then A_1. Adding an element in A_1 requires that it has $2d_r > 2$ neighbors in G, but this would contradict the third claim. Finally, the last inequality is obtained by observing that the considered set stays the same once v is added to A_1. And when v is added to A_1 we have $d_M(v) \geq 2d_r > 2$, so $N_{G'}(v) \cap C = \emptyset$ and so $N_{G'}(v) \subseteq (M \cup X) \setminus C$ □

Now following the same reasoning as in the sketch but this time using the graph $G'[(V(G) \cup X) \setminus C]$, we obtain $|A_0| \leq 2k_0$, $|A_2| \leq k_0 p_2(r,t)$ and $2d_r|A_1| \leq d_r(|A_0| + |A_1| + |A_2|)$ so $|M| \leq 4k_0 + 2k_0 p_2(r,t)$. □

Using Lemma 7.38 and Lemma 7.39, the following corollary is now immediate.

Corollary 7.41. *Let $(G, k, M, \mathcal{H})$ be an input of the algorithm $(r, \mathcal{G})$-*Ann*-FVS. If $A(G, k, M, \mathcal{H})$ reaches Line 22, then we have $|V(G)| = \mathcal{O}(k_0 p_6(r,t))$ where $p_6(r,t) = p_3(r,t)p_4(r,t)p_5(r,t)$.*

7.2.2 Complexity analysis

Complexity of the DP in the base case of Algorithm 2

Let us first describe how Algorithm 2 solves an instance $(G, k, M, \mathcal{H})$ using dynamic programming when no rule applies (Line 22). Recall that by the definition of $(r, \mathcal{G})$-Ann-FVS, G is a $K_{r,r}$-free graph from a nice graph class $\mathcal{G}$, and $\mathcal{H}$ a family of disjoint subsets of M, with each $H \in \mathcal{H}$ inducing a connected graph in G.

Theorem 7.42 ([CFK⁺15]). *An $(r, \mathcal{G})$-*Ann*-FVS instance $(G, k, M, \mathcal{H})$ with G an n-vertex graph can be solved in time $\mathbf{tw}(G)^{\mathcal{O}(\mathbf{tw}(G))} n^{\mathcal{O}(1)}$.*

Proof. For an instance $(G, M, k, \mathcal{H})$ with G a graph on n vertices, computing a tree decomposition with bags of size $\mathbf{tw}(G)$ up to a constant factor can be done in time $2^{\mathcal{O}(\mathbf{tw}(G))} n$ with Theorem 2.2. Given such a tree decomposition observe that a bag of size b can intersect at most b distinct $H \in \mathcal{H}$. Moreover, for $H \in \mathcal{H}$, having that H is connected ensures that the set of bags of the tree decomposition of G containing at least one vertex of H is a connected subset. Using those two observations, it is easy to adapt the standard dynamic programming algorithm solving FVS for solving the more general $(r, \mathcal{G})$-Ann-FVS without worsening the running time, and then obtaining the claimed complexity. □

Corollary 7.43. *Let $(G, k, M, \mathcal{H})$ be an input of $(r, \mathcal{G})$-*Ann*-FVS. If $A(G, k, M, \mathcal{H})$ reaches Line 22 and calls $DP(G, k, M, \mathcal{H})$, then the worst-case running time of $DP(G, k, M, \mathcal{H})$ is $T_{DP}(k_0, r, t) = 2^{\mathcal{O}\left(k_0^{\delta} p_7(r,t)\right)}$ with $p_7(r,t) = f_r \log(f_r k_0 p_6(r,t))(p_6(r,t))^{\delta}$.*

Proof. Denoting $n = |V(G)|$, according to Corollary 7.41, we have $n = \mathcal{O}(k_0 p_6(r,t))$, and as $\mathcal{G}$ is a nice class (see Definition 7.2), we have $\mathbf{tw}(G) \leq f_r n^{\delta}$. Thus, Theorem 7.42 implies that $DP(G, k, M, \mathcal{H})$ runs in time $(f_r n)^{\mathcal{O}\left(f_r n^{\delta}\right)} n^{\mathcal{O}(1)}$. □

Complexity of Algorithm 2

Lemma 7.44. *Given an input $(G, k, M, \mathcal{H}) \in \mathcal{I}$ with G a n-vertex graph, $A(G, k, M, \mathcal{H})$ runs in time*

$$T_{DP}(k_0, r, t)\,(2t)^{\frac{2k}{t-1}} n^{\mathcal{O}(1)}$$

where T_{DP} is defined in Corollary 7.43.

Proof. Let $f(n,k,m,h)$ be the worst-case complexity of $A(G,k,M,\mathcal{H})$ when $|V(G)| = n$, $|M| = m$, and $\mathcal{H} = h$. We denote $C_{DP} = T_{DP}(k_0,r,t)$. Let $p(n)$ be the worst-case complexity of all operations performed in one call to Algorithm 2 (which is bounded by the sum of the complexity of testing and applying all the kernelization and branching rules, and applying Lemma 7.2.1). Notice that, as all rules and Lemma 7.2.1 are polynomial, p is polynomial. Let us show by induction (on $2(n+k)-m-h$) that

$$f(n,k,m,h) \leq (p(n)+C_{DP})(2n-m)\left((2t)^{\frac{2k-h}{t-1}}-1\right).$$

Without loss of generality, we assume that p is non-decreasing. Let us bound f according to which recursive call we perform. In particular, if we apply (KR$_5$), we delete one vertex v from M, and as we remove from the packing $\mathcal{H}$ at most one set, the set containing v, we decrease h by $x \in \{0,1\}$. If we apply (BR$_2$), then we make t recursive calls on instances $((G-X_{\bar{v}}),k-(t-1),M\setminus X_{\bar{v}},\mathcal{H}-X_{\bar{v}})$ (re-using notations of (BR$_2$)), and thus in these calls we remove $x \in [t-1]$ hyperedges from $\mathcal{H}$, and in the worst-case also t recursive calls on instances $\left(G,k,M\cup R_i,\mathcal{H}\cup\mathcal{T}_{R_i}\right)$, where R_i is the union of $t-1$ parts of a $\widetilde{K_{t,t}}$, each part having size at most $p_2(r,t)$ by the definition of a $\widetilde{K_{t,t}}$. Subsequently, denoting

$$\begin{aligned}
h_1(n,k,m,h) &= f(n-(t-1),k-(t-1),m-(t-1),h-x)\\
h_2(n,k,m,h) &= f(n,k,m+(t-1)p_2(r,t),h+(t-1)),
\end{aligned}$$

this leads to the following bound:

$$f(n,k,m,h) \leq \begin{cases} p(n)+f(n-x,k,m,h), & x \geq 1,\ \text{(KR}_1\text{) (KR}_2\text{) (KR}_4\text{)}\\ p(n)+f(n,k,m+1,h) & \text{when (KR}_3\text{)}\\ p(n)+f(n-1,k-1,m-1,h-x) & x\in\{0,1\}\text{ when (KR}_5\text{)}\\ p(n)+th_1(n,k,m,h)+th_2(n,k,m,h) & 0\leq x<t\text{ when (BR}_2\text{)}\\ p(n)+C_{DP} & \text{No rule applied, DP}\end{cases}$$

It remains to check by induction that our bound hold in any of these cases. The first two and last cases are straightforward. For the third case, we get

$$\begin{aligned}
f(n,k,m,h) &\leq p(n)+f(n-1,k-1,m-1,h-x)\\
&\leq p(n)+(p(n-1)+C_{DP})(2n-m-1)\left((2t)^{\frac{2k-h-1}{t-1}}-1\right)\\
&\leq p(n)+(p(n)+C_{DP})(2n-m-1)\left((2t)^{\frac{2k-h-1}{t-1}}-1\right)\\
&\leq (p(n)+C_{DP})(2n-m)\left((2t)^{\frac{2k-h}{t-1}}-1\right).
\end{aligned}$$

For the fourth case,

$$h_1(n,k,m,h) \leq (p(n-(t-1)) + C_{DP})(2n-m-(t-1))\left((2t)^{\frac{2k-h-2(t-1)+x}{t-1}} - 1\right)$$
$$\leq (p(n) + C_{DP})(2n-m)\left((2t)^{\frac{2k-h}{t-1}-1} - 1\right),$$

$$h_2(n,k,m,h) \leq (p(n) + C_{DP})(2n-m)\left((2t)^{\frac{2k-h}{t-1}-1} - 1\right).$$

This leads to the following upper bound for the fourth case:

$$\begin{aligned} f(n,k,m,h) &\leq p(n) + t(h_1(n,k,m,h) + h_2(n,k,m,h)) \\ &\leq p(n) + (p(n) + C_{DP})(2n-m)(2t)\left((2t)^{\frac{2k-h}{t-1}-1} - 1\right) \\ &\leq (p(n) + C_{DP})\,(2n-m)\left((2t)^{\frac{2k-h}{t-1}} - 1\right). \end{aligned}$$

This concludes the induction. The wanted result is then easily obtained from the proven bound. □

Let us now bound the running time of the main algorithm (that branches to create the family of instance $\mathcal{I}$ and runs Algorithm 2 on each instance).

Theorem 7.5. *For every nice hereditary graph class $\mathcal{G}$ there is a constant $\eta < 1$ such that FVS can be solved for $\mathcal{G}$ in time $2^{\mathcal{O}(k^{\eta})} \cdot n^{\mathcal{O}(1)}$ by a robust algorithm.*

Proof. Let us denote $n_0 = |V(G_0)|$. According to Lemma 7.10, the complexity to generate $\mathcal{I}$ is in $2^{\mathcal{O}(r\log k_0)} n_0^{\mathcal{O}(1)} (2r)^{\frac{k_0}{r-1}}$ and $|\mathcal{I}| = \mathcal{O}\left((2r)^{\frac{k_0}{r-1}}\right)$. Deciding one instance $(G,k,M,\mathcal{H}) \in \mathcal{I}$ with G a n-vertex graph can be done in time $T_{DP}(k_0,r,t)(2t)^{\frac{2k_0}{t-1}} n_0^{\mathcal{O}(1)}$ where $T_{DP}(k_0,r,t) = 2^{\mathcal{O}\left(k_0^{\delta} p_7(r,t)\right)}$ according to Lemma 7.44, Corollary 7.43 and the observation that $k \leq k_0$ and $n \leq n_0$.

Thus, the overall running time is bounded by:

$$\left(2^{\mathcal{O}\left(r\log k_0 + \frac{k_0\log r}{r}\right)} + 2^{\mathcal{O}\left(\frac{k_0\log r}{r} + \frac{k_0\log t}{t} + k_0^{\delta} p_7(r,t)\right)}\right) n^{\mathcal{O}(1)}.$$

Now let us recall that t is defined accordingly to r by $t = 2d_r = r^{\mathcal{O}(1)}$ and so $p_7(r,t) = r^{\mathcal{O}(1)}$. More precisely if we have $f_1(r) = \widetilde{\mathcal{O}}\left(r^{c_{f_1}}\right)$, $f_2(r,p,m) = \widetilde{\mathcal{O}}(r^{c_{f_2}}(p+m)^{c'_{f_2}})$, $f_r = \widetilde{\mathcal{O}}(r^{c_f})$ and $d_r = \widetilde{\mathcal{O}}(r^{c_d})$ (remember that without loss of generality we supposed $c_d \geq 1$), then we have $p_7(r,t) = \widetilde{\mathcal{O}}(r^{c_7})$ with

$$\begin{aligned} c_7 &= c_f + \delta(2(c_d + c_{f_2} + c_{f_2'}(c_{f_1} + (6+\alpha)c_d)) + c_{f_1} + (6+\alpha)c_d) \\ &= c_f + \delta(2(c_d + c_{f_2}) + (c'_{f_2} + 1)(c_{f_1} + (6+\alpha))). \end{aligned}$$

Thus, by taking $r = k_0^{\varepsilon}$ with $\varepsilon = \frac{1-\delta}{c_7+1}$ we obtain a running time $2^{\widetilde{\mathcal{O}}(k_0^{1-\varepsilon})} n_0^{\mathcal{O}(1)}$. For getting rid of the hidden logarithmic factors in the exponent and obtaining the wanted result it then suffices to replace ε with $0 < \varepsilon' < \varepsilon$. □

7.2.3 Applications

In this section, we prove that s-string graphs and pseudo-disk graphs are nice graph classes for some parameters. We recall that a system of pseudo-disks is a collection of regions in the plane homeomorphic to a disk such that any two of them share at most 2 points of their boundary. Similar arguments are used for both considered classes, but in the case of pseudo-disks the arguments are a bit simpler, we then consider this class first.

Application to pseudo-disk graphs

In this section, we prove that the class of pseudo-disk graphs is nice, and so by our main theorem it admits a robust subexponential FPT algorithm for FVS.

Lemma 7.45. *There exist constants c_1, c_2, c_3, c_4 such that the class of pseudo-disk graphs is a nice class with parameters $\alpha = 4$, $f_1(r) = c_1$, $f_2(r, p, m) = c_2 p^3$, $\delta = \frac{1}{2}$, $f_r = c_3\sqrt{r \log r}$ and $d_r = c_4 r \log r$.*

Proof. Most of the conditions were already stated with the bounds of Corollary 2.13 for the treewidth, Theorem 2.11 for the number of edges and Lemma 4.14 for the tree-neighborhood complexity. The only property that remains to check is that in a pseudo-disk graph G, identifying adjacent degree-two vertices u and v obtaining a new vertex w preserves being a pseudo-disk graph. By considering $(\mathcal{D}_q)_{q \in V(G)}$ a pseudo-disk representation of G, we can take $\mathcal{D}_w = \mathcal{D}_u \cup \mathcal{D}_v$ as it is homeomorphic to a disk and its boundary will cross the second neighbor of u (respectively v) as many time than the boundary of $\mathcal{D}_u$ (respectively $\mathcal{D}_v$). □

And so applying our main theorem we obtain:

Corollary 7.7. *There exists $\eta < 1$, such that there is a robust parameterized subexponential algorithm solving FVS in time $2^{\mathcal{O}(k^{\eta})} n^{\mathcal{O}(1)}$ for n-vertex pseudo-disk graphs.*

Note that our main theorem gives that it suffices to take $\eta > \frac{44}{45}$ for the result to hold. Sharper results for pseudo-disk graphs will be given in the rest later in this chapter, first in Section 7.3 by generalizing the robust algorithm of [LPS+22] dealing with disk graphs to pseudo-disk graphs, and secondly in Section 7.4 by using the representation as pseudo-disks for obtaining an even better running time.

Application to s-string graphs

We now show how to adapt the arguments from the previous section to the case of s-string graphs.

Lemma 7.46. *There exist constants c_1, c_2, c_3, c_4 such that for every $s \geq 1$ the class of s-string graphs is a nice class with parameters $\alpha = 4$, $f_1(r) = c_1 s^4 r \log r$, $f_2(r, p, m) = c_2(sr \log r)^4 (p+m)^3$, $\delta = \frac{1}{2}$, $f_r = c_3\sqrt{r \log r}$ and $d_r = c_4 r \log r$.*

Proof. Again, most of the conditions were already stated in Corollary 2.13, Theorem 2.4 and Lemma 4.15. The only remaining property to prove is that given an s-string graph G, identifying adjacent degree-two vertices u and v preserves being an s-string graph. We do this by replacing in an s-string representation $(\mathcal{S}_q)_{q \in V(G)}$ of G the strings $\mathcal{S}_u$ and $\mathcal{S}_v$ with a new string $\mathcal{S}_w$ while preserving the global property of being an s-string family. $\mathcal{S}_u$ contains a Jordan arc a_u linking its two neighbors, without being intersected by any string in its interior (see Figure 7.5). Let us denote p_v the extremity of a_u in $\mathcal{S}_v$. Then $\mathcal{S}_v$ contains a Jordan arc a_v linking p_v to the string representing its second neighbor, without being intersected by any other string than $\mathcal{S}_u$. The string $\mathcal{S}_w$ is then obtained by taking the union of a_u and a_v. This construction ensures that the number of crossings between two strings other than $\mathcal{S}_u$ and $\mathcal{S}_v$ of the initial representation remains the same. We now have to bound the number of crossing of $\mathcal{S}_w$ with other strings. Recalling that u and v do not share a common neighbor, and that by constructions w has 2 crossings (one for each distinct neighbor), w does not have more than 1 crossings with another string. Therefore, after replacing $\mathcal{S}_u$ and $\mathcal{S}_v$ by $\mathcal{S}_w$ we still have an s-string family. $\square$

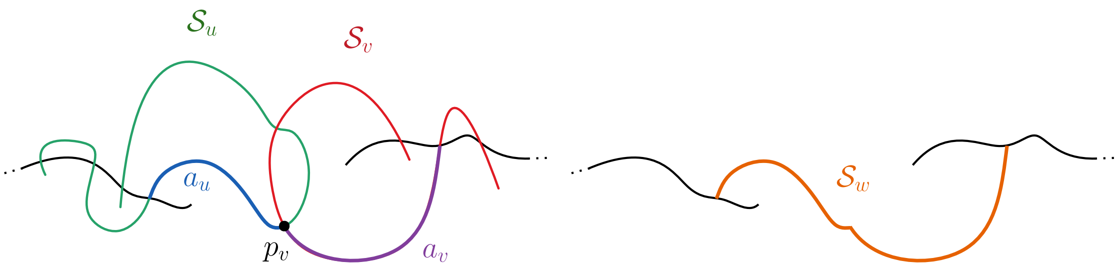

Figure 7.5: Illustration of the construction used in the proof of Lemma 7.46 for contracting an edge between adjacent degree-two vertices without a common neighbor.

And so applying our main theorem we obtain:

Corollary 7.6. *There exists $\eta < 1$, such that for all s there is a robust parameterized subexponential algorithm solving FVS in time $2^{\widetilde{\mathcal{O}}(s^{\mathcal{O}(1)}k^{\eta})}n^{\mathcal{O}(1)}$ for n-vertex s-string graphs.*

More precisely, we can take $\eta = \frac{52}{53}$.

7.3 Robust algorithms for TH and FVS in pseudo-disk graphs.

In this section, we explain how to generalize the robust algorithms for TH and FVS in disk graphs of Theorem 3.15 for pseudo-disk graphs. The first obstacle for generalization is that both algorithms use an EPTAS [BBB+21] for the clique problem in disk graphs, and such an algorithm is not known for pseudo-disk graphs. They use this EPTAS for the branching process removing large cliques presented in Subsection 3.1.1. Actually, to apply this branching (without altering its time complexity), one only needs an algorithm running in time $2^{\mathcal{O}\left(\frac{k}{p}\log p\right)}n^{\mathcal{O}(1)}$ finding a clique of size p, if such clique exist. If p is chosen such that $p = \mathcal{O}\left(\sqrt{k}\right)$, the algorithm of Lemma 3.2 runs in time $2^{\mathcal{O}(\frac{k}{p}\log p)}n^{\mathcal{O}(1)}$, and with this value the branching holds, with the same time-complexity. After this branching process, the algorithms for TH and for FVS differ.

For the case of Triangle Hitting a core (see [ACO23] for the definition) of size $\mathcal{O}(pk)$ can be found in the same manner as in the original article (using Theorem 2.9 to obtain the bound on the size of the core), giving a generalization to pseudo-disks with the same complexity as the case of disk graphs.

For the FVS problem, the running time of the robust algorithm is bounded using the key result Lemma 3.14 from [LPS+22]. It was stated for disk graphs, but the proof only uses the fact that disks are pseudo-disks so it can be directly generalized to pseudo-disk graphs.

However, being able to bound the size of the cliques and this key lemma are not sufficient, a last obstacle to generalizing the robust algorithm of [ACO23] is to bound the number of "deep and regular" vertices (see [ACO23] for the definition). In the Lemma 8 of [ACO23], the number of such vertices in disk graphs is bounded by $\mathcal{O}(p^2k)$ using arguments specifics to disks. However, using Theorem 4.12 would give the bound $\mathcal{O}(p^3k)$ for pseudo-disk graphs,[3] resulting in a worse complexity. The obtained results are summarized in Table 7.1. The symbol $\star$ next to the

[3]More details will be given in the next section for the proof of Lemma 7.50, where a set of vertices equivalent to deep and regular vertices will be defined. They will be described as "inner-M'-vertices whose hosted graphs are paths" and their number is bounded by Claim 7.59.

complexity indicates that a geometric representation of the input graph is needed.

Class	Problem	Time complexity	Section
Disk	FVS	$2^{\widetilde{\mathcal{O}}(k^{7/8})} n^{\mathcal{O}(1)}$ ★	[ACO23]
		$2^{\widetilde{\mathcal{O}}(k^{9/10})} n^{\mathcal{O}(1)}$	
	TH	$2^{\widetilde{\mathcal{O}}(k^{2/3})} n^{\mathcal{O}(1)}$ ★	
		$2^{\widetilde{\mathcal{O}}(k^{3/4})} n^{\mathcal{O}(1)}$	
Pseudo-disk	FVS	$2^{\widetilde{\mathcal{O}}(k^{6/7})} n^{\mathcal{O}(1)}$ ★	Subsection 7.4.3
		$2^{\widetilde{\mathcal{O}}(k^{10/11})} n^{\mathcal{O}(1)}$	Section 7.3
	TH	$2^{\widetilde{\mathcal{O}}(k^{3/4})} n^{\mathcal{O}(1)}$	

Table 7.1: Comparison of previous results with the results presented in this chapter.

Here we have only considered the generalizations of the robust algorithms of [ACO23], as their analysis for the complexities of the algorithms taking a representation is mainly based on the result that disk graphs have balanced separators of clique-size $\mathcal{O}(\sqrt{n})$, but for pseudo-disk graphs only separators of clique-size $\mathcal{O}(n^{2/3} \log n)$ are known (see [dBKBMT23] for the definition of the clique-size of a separator, and the mentioned bounds), giving worse complexities than the robust cases.

7.4 Using a pseudo-disk graph representation

In this section, we keep investigating the existence of FPT subexponential algorithms for the cycle-hitting problems Feedback Vertex Set. We now focus on the class of pseudo-disk graphs, and prove that assuming a geometric representation is given, FVS can be solved in time $2^{\widetilde{\mathcal{O}}(k^{6/7})} n^{\mathcal{O}(1)}$ and TH can be solved in time $2^{\widetilde{\mathcal{O}}(k^{3/4})} n^{\mathcal{O}(1)}$ as well.

7.4.1 Presentation of the algorithms

We show in Section 7.3 that the algorithms of Theorem 3.15 and the analysis of their complexities can be generalized to pseudo-disk graphs with minor modifications and using some results already introduced in this thesis. Here, in the case of solving the FVS problem with a representation given as input, we provide a new approach leading to an improved time complexity of $2^{\mathcal{O}(k^{6/7}\log k)}n^{\mathcal{O}(1)}$ (Theorem 7.49), improving on the state of the art for disk graphs.

For this, we apply the following strategy. Given an input (G,k), the objective is to reduce the treewidth of G to $o(k)$ in order to solve FVS in $2^{\mathcal{O}(\mathbf{tw}(G))}n^{\mathcal{O}(1)}$ using a standard dynamic programming algorithm [CFK⁺15]. To do so, we first branch using techniques of [LPS⁺22] to reduce our input, to a collection of pseudo-disk graphs with bounded clique number (or equivalently with ply at most p). Then we reduce the sizes of these instances to kernels of size $\mathcal{O}(p^4k)$. This implies, using a bound on the number of edges in a pseudo-disk graph of bounded clique number, that these kernels have treewidth $\mathcal{O}(\sqrt{pn}) = \mathcal{O}(\sqrt{p^5k})$.

Organization of the section. In Subsection 7.4.2 we give the necessary definitions and properties and present the aforementioned preprocessing step. Subsection 7.4.3 reduces our main result to a technical kernelization lemma, that is proved in Subsection 7.4.4. Subsection 7.4.5 is devoted to a subclass of pseudo-disk graphs, contact graphs of segments, for which the analysis of the algorithm provides a better running time.

7.4.2 Preliminaries

Representation of pseudo-disk graphs. The considered algorithms in this section takes a representation of the pseudo-disk graph as an input, so we need to choose the encoding of such representation. A system of pseudo-disks $\mathcal{S}$ is represented by the directed plane multi-graph $\overrightarrow{P_\mathcal{S}}$ defined as follows (see Figure 7.6). For any two $\mathcal{D}_1,\mathcal{D}_2 \in \mathcal{S}$, every point in $\partial\mathcal{D}_1 \cap \partial\mathcal{D}_2$ is a vertex of $\overrightarrow{P_\mathcal{S}}$. For any $\mathcal{D} \in \mathcal{S}$, the Jordan arcs in $\partial\mathcal{D}$ joining any two such points form the arcs of $\overrightarrow{P_\mathcal{S}}$. Those arcs are oriented in such a way that $\partial\mathcal{D}$ corresponds to a clockwise cycle around $\mathcal{D}$. It may remain disks $\mathcal{D} \in \mathcal{S}$ with uncrossed boundaries (i.e. $\partial\mathcal{D} \cap \partial\mathcal{D}' = \emptyset$ for any $\partial\mathcal{D}' \in \mathcal{S}$). For such a disk, we pick an arbitrary point in $\partial\mathcal{D}$ as a vertex for $\overrightarrow{P_\mathcal{S}}$, and the rest of $\partial\mathcal{D}$ corresponds to a clockwise loop on this vertex. Note that $\overrightarrow{P_\mathcal{S}}$ has at most $2|E(G_\mathcal{S})| + |V(G_\mathcal{S})|$ vertices, and at most $4|E(G_\mathcal{S})| + |V(G_\mathcal{S})|$ edges. This is a convenient feature of pseudo-disk systems to admit a polynomial (in terms of $G_\mathcal{S}$) space data structure representing them. This graph $\overrightarrow{P_\mathcal{S}}$ is called the ***pseudo-disk representation*** of $\mathcal{S}$ and $G_\mathcal{S}$. We denote $\overrightarrow{P^*_\mathcal{S}}$ the dual graph of $\overrightarrow{P_\mathcal{S}}$. Observe that any arc is oriented from a region (of $\mathcal{S}$) with lower ply towards one with higher ply.

For several useful operations, updating a pseudo-disk representation is simple.

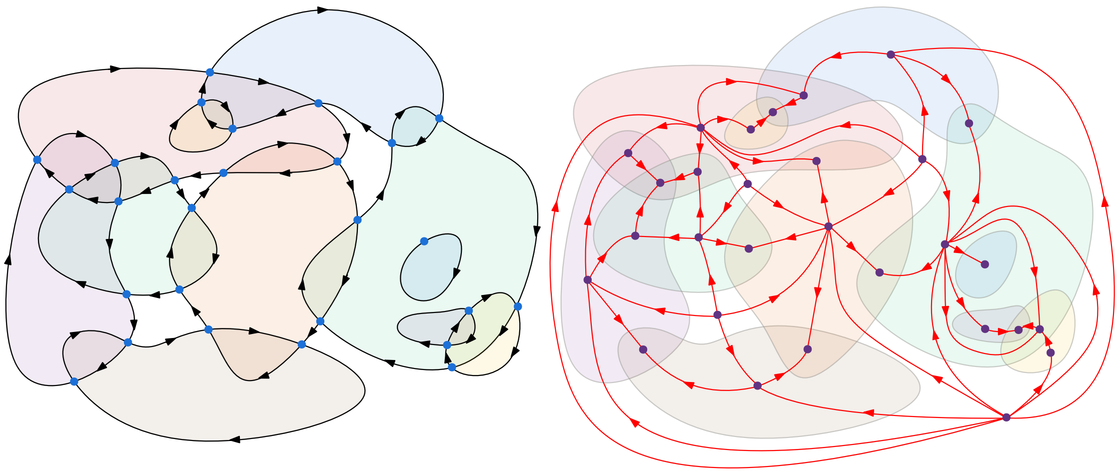

Figure 7.6: A pseudo-disk representation $\overrightarrow{P_S}$ and its dual $\overrightarrow{P_S^*}$.

Lemma 7.47. *Given a pseudo-disk graph G, deleting any vertex, or any edge uv that does not belong to a triangle, or contracting any edge uv that does not belong to a triangle results in a pseudo-disk graph. Furthermore, given a representation of G of ply p, one can update the representation in linear time and the new representation has ply at most p.*

Proof. The deletion of a vertex v is obtained by simply removing $\mathcal{D}_v$ from the representation.

The deletion of an edge uv that does not belong to any triangle is obtained by replacing $\mathcal{D}_u$ and $\mathcal{D}_v$, by $\mathcal{D}_u \setminus \mathcal{D}_v$ and by $\mathcal{D}_u \setminus \mathcal{D}_v$,[4] respectively. As we start from a system of pseudo-disks, these subsets are homeomorphic to disks. Furthermore, as $\mathcal{D}_u \cap \mathcal{D}_v$ does not intersect any other pseudo-disk $\mathcal{D}_w$, for $w \in V(G) \setminus \{u, v\}$, we have that $\partial\mathcal{D}_w$ intersect the new $\partial\mathcal{D}_u$ (resp. $\partial\mathcal{D}_v$) as many times as the former one, that is zero or twice. So, this system is a system of pseudo-disks.

The contraction of uv is obtained by replacing $\mathcal{D}_u$ and $\mathcal{D}_v$ by $\mathcal{D}_u \cup \mathcal{D}_v$, and let us denote w the new vertex. As we start from a system of pseudo-disks $\mathcal{D}_w = \mathcal{D}_u \cup \mathcal{D}_v$ is homeomorphic to a disk. Now, consider any vertex $t \in N(u) \setminus \{v\}$ and let us prove that $\partial\mathcal{D}_t$ and $\partial\mathcal{D}_w$ intersect at 0 or 2 points. The case of a vertex $t' \in N(v) \setminus \{u\}$ is handled symmetrically. We divide the closed curves $\partial\mathcal{D}_u$ and $\partial\mathcal{D}_v$ into two arcs each, as follows: $\mathcal{A}_u^{int} = \partial\mathcal{D}_u \cap \mathcal{D}_v$, $\mathcal{A}_u^{ext} = \partial\mathcal{D}_u \setminus \mathcal{D}_v$, $\mathcal{A}_v^{int} = \partial\mathcal{D}_v \cap \mathcal{D}_u$, and $\mathcal{A}_v^{ext} = \partial\mathcal{D}_v \setminus \mathcal{D}_u$. Note that as $t \notin N(v)$, we have that none of $\mathcal{A}_u^{int}$, $\mathcal{A}_v^{int}$ or $\mathcal{A}_v^{ext}$

[4] To be precise, $\mathcal{D}_u \setminus \mathcal{D}_v$ is not closed, and hence not homeomorphic to a closed disk. Here and later, let us use this notation to denote a closed subset homeomorphic to a disk that is arbitrarily close to $\mathcal{D}_u \setminus \mathcal{D}_v$ while remaining inside this set.

intersects $\partial\mathcal{D}_t$. Thus, $\partial\mathcal{D}_t$ intersects $\mathcal{A}_u^{ext}$ in either 0 or 2 points (the first case occurs if $\mathcal{D}_w \subsetneq \mathcal{D}_u$). This implies that $\partial\mathcal{D}_t$ intersects $\partial\mathcal{D}_w = \mathcal{A}_u^{ext} \cup \mathcal{A}_v^{ext}$ in either 0 or 2 points. As the other intersections of the system, not involving $\mathcal{D}_w$, did not change, the obtained system is a system of pseudo-disks.

Given a pseudo-disk representation $\overrightarrow{P_\mathcal{S}}$ of G with ply p, in any of the above three operations, the update is very simple. Actually, for the edge deletion and the edge contraction, these could even be performed in constant time. It is also clear that, in any case, the ply can only decrease.

□

Lemma 7.48 ensures us that it is easy to provide a constant approximation of the clique number for a pseudo-disk graph given a representation.

Lemma 7.48. *There is a constant c such that for every graph G, if G admits a pseudo-disk representation with ply p, then $p \leq \omega(G) \leq cp$. This implies that given a pseudo-disk representation, there is a polynomial time algorithm that returns a clique whose size is within a constant factor from the optimum.*

Proof. The lower-bound is trivial so we discuss the upper-bound. It is clear that a maximum clique in G admits a ply p representation. Thus, by Theorem 2.9 applied to this subgraph, $\omega(G)(\omega(G)-1)/2 \leq cp\omega(G)$ for some constant c, which leads to $\omega(G) \leq (2c+1)p$. Hence, the clique formed by all the disks that contain some fixed region of maximum ply has at least $\omega(G)/c'$ vertices with $c' = 2c+1$.

In order to compute such a clique given a representation $\overrightarrow{P_\mathcal{S}}$ we proceed as follows. Recall that $\overrightarrow{P_\mathcal{S}^*}$ is the dual of $\overrightarrow{P_\mathcal{S}}$ (see Figure 7.6). First, we construct a spanning tree T of $\overrightarrow{P_\mathcal{S}^*}$ (ignoring the orientations of the edges). To compute the ply of each region, it suffices to notice that for any arc rr' of T, the ply of r' is one more than the one of r, and that the minimum ply among the regions of $\mathcal{S}$ (i.e. the vertices of T) equals zero. Then, for a region r with maximum ply, to output the list of pseudo-disks containing it, one can consider a path P in T from r to a region of ply zero, and select the pseudo-disks $\mathcal{D}_v$ such that $\partial\mathcal{D}_v$ is crossed an odd number of times by P. □

Preliminary branching step

As first step of our algorithms we make use of the routine presented in Lemma 3.4. For applying this result, an approximation of the maximum clique is needed. Given a pseudo-disk representation Lemma 7.48 ensures us that such approximation can be computed by looking at the maximal ply region. As in this procedure, every graph G' in the collection $\mathcal{C}$ is an induced subgraph of G, Lemma 7.47 ensures us that we can also get a pseudo-disk representation $\overrightarrow{P'}$ for each of them. Note that since every graph G' in $\mathcal{C}$ has clique number at most p, by Lemma 7.48, its representation $\overrightarrow{P'}$ has ply at most p.

7.4.3 Hitting cycles in pseudo-disk graphs

The main result of this section is the following.

Theorem 7.49. *There is an algorithm that, given a n-vertex pseudo-disk graph with a representation and a parameter k, solves FVS in time $2^{\mathcal{O}(k^{6/7}\log k)}n^{\mathcal{O}(1)}$.*

To achieve this, we proceed in three steps. We first use the preprocessing Lemma 3.4 for bounding the ply. Then, we kernelize each instance provided by the branching process in Lemma 7.50. We show that these kernelized instances have small treewidth, and thus we can conclude with a dynamic algorithm to solve each of them. The main technical ingredient of this proof is the following lemma, whose full proof is deferred to Subsection 7.4.4.

Lemma 7.50. *Given a quadruple $(G, \overrightarrow{P_S}, M, k)$ as given by Lemma 3.4, there is a polynomial time algorithm that returns an equivalent instance (G', k') where $k' \leq k$ and G' is a pseudo-disk graph with ply at most p and $\mathcal{O}(p^4 k)$ vertices.*

As we will apply Lemma 3.4 for $p = \mathcal{O}(k^\epsilon)$, the above will directly imply that $|V(G')|$ is almost linear in k, and, using Corollary 2.15, that G' has sublinear treewidth. Thus, the whole technical difficulty is to prove Lemma 7.50. Now that the lemma is stated, we can proceed with the proof of the theorem.

Proof of Theorem 7.49. We apply Lemma 3.4 on an instance (G, k) provided with a representation $\overrightarrow{P_S}$ and on a value $p \in [6, k]$ to be set later. As a result we obtain a collection $\mathcal{C}$ of $2^{\mathcal{O}\left(\frac{k}{p}\log k\right)}$ instances that are each provided with a ply p pseudo-disk representation, and with a feedback vertex set M of size $\mathcal{O}(pk)$ that intersects every triangle on at least two vertices. Furthermore, solving these instances of FVS is enough to get a solution to our initial instance (G, k). This first stage is done in time $2^{\mathcal{O}\left(\frac{k}{p}\log k\right)}n^{\mathcal{O}(1)}$.

Then for each of the $2^{\mathcal{O}\left(\frac{k}{p}\log k\right)}$ obtained instances, we apply the kernelization described in Lemma 7.50, and we get a pseudo-disk graph G' with ply at most p and with only $\mathcal{O}(p^4 k)$ vertices. By Corollary 2.15, this kernel has treewidth at most $\mathcal{O}(\sqrt{p^5 k})$, and can thus be solved in time $2^{\mathcal{O}(\mathbf{tw}(G'))}n^{\mathcal{O}(1)} = 2^{\mathcal{O}\left(p^{5/2}k^{1/2}\right)}n^{\mathcal{O}(1)}$ with classical algorithms [CFK+15]. The overall time complexity of this algorithm is

$$2^{\mathcal{O}\left(\frac{k}{p}\log k\right)}n^{\mathcal{O}(1)} + 2^{\mathcal{O}\left(\frac{k}{p}\log k\right)} \cdot n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}\left(p^{5/2}k^{1/2}\right)}n^{\mathcal{O}(1)}.$$

Thus, setting $p = k^{\frac{1}{7}}$ the time complexity is indeed $2^{\mathcal{O}\left(k^{6/7}\log k\right)}n^{\mathcal{O}(1)}$. $\square$

7.4.4 Proof of Lemma 7.50

Recall that in Lemma 7.50 we are given a quadruple $(G, \overrightarrow{P_\mathcal{S}}, M, k)$ as computed by Lemma 3.4. The equivalent instance computed in Lemma 7.50 will be obtained using reduction rules. Before applying these rules, we define the set M' by adding to M every pseudo-disk $\mathcal{D}$ that contains a point of $\partial\mathcal{D}_u \cap \partial\mathcal{D}_v$, for $u, v \in M$ (see Figure 7.7, left). Given a vertex set X, we denote $\mathcal{R}_X$ the subset of the plane defined by $(\cup_{u\in X}\mathcal{D}_u)$. Let us now partition the vertices of $V(G) \setminus M'$ into three types.

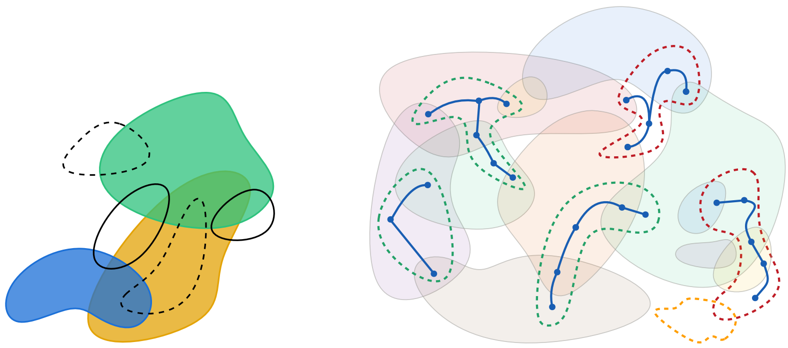

Figure 7.7: Left: From M to M'. The pseudo-disks of M are filled. Among the others, those added to M' have solid border. Right: Here, the filled pseudo-disks belong to M'. In dashed green, the inner pseudo-disks of $I_{M'}$. In dashed red, the border pseudo-disks of $B_{M'}$. In dashed orange, the only pseudo-disk $v \in O_{M'}$. For any pseudo-disk u not in M', we depicted its hosted graph H_u, which is defined in Definition 7.53.

Definition 7.51. *A vertex $v \in V(G) \setminus M'$ is:*

- *an* **inner-M'-vertex** *if $\mathcal{D}_v \subseteq \mathcal{R}_{M'}$,*
- *a* **border-M'-vertex** *if $\mathcal{D}_v$ intersects $\mathcal{R}_{M'}$ without being included, and*
- *an* **outer-M'-vertex** *if $\mathcal{D}_v$ does not intersect $\mathcal{R}_{M'}$.*

We denote $I_{M'}$, $B_{M'}$, and $O_{M'}$ the vertex sets with these three types of vertices (see Figure 7.7).

Let us state a few properties about these vertices.

Claim 7.52.

(a) $|M'| = \mathcal{O}(p|M|) = \mathcal{O}(p^2k)$.

(b) *For any edge uv of $G - M'$, the intersection $\mathcal{D}_u \cap \mathcal{D}_v$ does not intersect $\mathcal{R}_{M'}$.*

(c) *For every $u \in V(G) \setminus M'$, the pseudo-disk $\mathcal{D}_u$ does not contain any point of $\partial\mathcal{D}_v \cap \partial\mathcal{D}_w$, for any two vertices $v, w \in M'$.*

(d) *In $G - M'$ the vertices of $I_{M'}$ are isolated (i.e. $\forall v \in I_{M'}$ we have that $N(v) \subseteq M'$).*

(e) *For every non-empty tree T of $G - M' - I_{M'}$, the set $\mathcal{R}_T \setminus \mathcal{R}_{M'}$ is non-empty and connected, with $\mathcal{R}_T = \left(\cup_{v\in V(T)}\mathcal{D}_v\right)$.*

Proof. Property (a) is a consequence of the following facts. By assumption of the statement we aim to show (Lemma 7.50), G is a pseudo-disk graph with ply at most p, thus Theorem 2.9 implies that $G[M]$ has $\mathcal{O}(p|M|)$ edges. Hence, in the representation of G there are at most $\mathcal{O}(p|M|)$ intersection point of the borders of two pseudo-disks $\mathcal{D}_1, \mathcal{D}_2 \in M$. Then, since each pseudo-disk in $M' \setminus M$ contains such an intersection point (by the definition of M'), and since each of those points belongs to at most one pseudo-disk $\mathcal{D} \notin M$ (otherwise there would be a triangle with at most one vertex in M), we have that $|M' \setminus M| = \mathcal{O}(p|M|)$. Hence, $|M'| = \mathcal{O}(p|M|)$.

We prove Property (b) by contradiction: assume that for some edge uv of $G - M'$, the intersection $\mathcal{D}_u \cap \mathcal{D}_v$ intersects $\mathcal{R}_{M'}$. In that case there is a triangle uvw with at most one vertex, w, in $M \subseteq M'$, a contradiction.

We prove Property (c) by contradiction as well: assume that for some $v \in V(G) \setminus M'$, the pseudo-disk $\mathcal{D}_u$ contains a point of $\partial\mathcal{D}_v \cap \partial\mathcal{D}_w$ with $v, w \in M'$. The vertices u, v and w induce a triangle, so by construction of M at least two of them are in M. As we have $u \notin M'$ and so $u \notin M$, necessarely $v, w \in M$ and so by construction of M' we have $u \in M'$ as D_u contains a intersection point of $\partial\mathcal{D}_v \cap \partial\mathcal{D}_w$ with $v, w \in M$, a contradiction.

Property (d) follows from the definition of inner-M'-vertices and Property (b).

For Property (e), suppose towards a contradiction that some connected component $\mathcal{R}^*$ of $\mathcal{R}_T \cap \mathcal{R}_{M'}$ is bordered by a sequence of Jordan arcs $\partial\mathcal{R}^*_1, \ldots, \partial\mathcal{R}^*_t$ with $t \geq 4$, and such that $\partial\mathcal{R}^*_i$ is either a sub-arc of $\partial\mathcal{R}_T$, if i is odd, or a sub-arc of $\partial\mathcal{R}_{M'}$, if i is even. Note that by Property (b), for any even i the arc $\partial\mathcal{R}^*_i$ is contained inside $\mathcal{D}_u$ for some $u \in V(T)$, but it does not intersect any other pseudo-disk of T. For any two even values i, j, let $\mathcal{A} \subset \mathcal{R}^*$ be a Jordan arc linking $\partial\mathcal{R}^*_i$ and $\partial\mathcal{R}^*_j$. Among the possible Jordan arcs, let $\mathcal{A}$ be one minimizing the number of crossings with borders of M'. Let us denote p_i and p_j the endpoints of $\mathcal{A}$, and let us prove that they belong to the same pseudo-disk of T, say $\mathcal{D}_u$. Indeed, if p_i and p_j would belong to distinct pseudo-disks of T, say $\mathcal{D}_{u_1}$ and $\mathcal{D}_{u_2}$, then when leaving $\mathcal{D}_{u_1}$, the arc $\mathcal{A}$ would still be in $\mathcal{R}_{M'}$ and it would also be in a distinct pseudo-disk $\mathcal{D}_u$ of T, contradicting Property (b). Hence, all the arcs $\partial\mathcal{R}^*_i$, with i even, are contained in the same pseudo-disk $\mathcal{D}_u$.

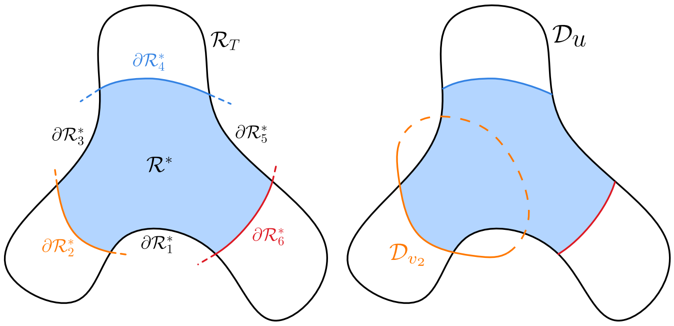

Figure 7.8: Proof of Property (e) (left) The region $\mathcal{R}^*$. (right) The pseudo-disks $\mathcal{D}_u$ and $\mathcal{D}_{v_2}$.

Note that by Property (c), the arc $\partial\mathcal{R}_2^*$ is contained in the border $\partial\mathcal{D}_{v_2}$ of some vertex $v_2 \in M'$. As $\mathcal{D}_u \cap \mathcal{D}_{v_2} \subseteq \mathcal{D}_u \cap \mathcal{R}_T$, we have that $\partial\mathcal{D}_u$ and $\partial\mathcal{D}_{v_2}$ intersect in at least four points (see Figure 7.8), a contradiction.

□

Let $\mathcal{S}'$ be the restriction of $\mathcal{S}$ to vertices of M'. Let us denote $\overrightarrow{P_{\mathcal{S}'}}$ the representation of $G[M']$, and $\overrightarrow{P^*_{\mathcal{S}'}}$ its dual. Since the pseudo-disk system has ply at most p, the graph $G[M']$ has $\mathcal{O}(p|M'|) = \mathcal{O}(p^3k)$ edges by Theorem 2.9. Each of such edge induces at most two vertices in $\overrightarrow{P_{\mathcal{S}'}}$. Hence, $\overrightarrow{P_{\mathcal{S}'}}$ has $\mathcal{O}(p^3k)$ vertices, and thus by planarity, the number of edges, and faces in $\overrightarrow{P_{\mathcal{S}'}}$ is also $\mathcal{O}(p^3k)$.

Definition 7.53. *Given a vertex $u \in V(G) \setminus M'$, we define the* **hosted graph** H_u *as a plane graph drawn within $\mathcal{D}_u$, with a (single) vertex in a face f of $\overrightarrow{P_{\mathcal{S}'}}$ if and only if $\mathcal{D}_u$ and f intersect, and with edges between vertices lying in adjacent faces of $\overrightarrow{P_{\mathcal{S}'}}$, if their common border intersects $\mathcal{D}_u$ (see Figure 7.7, right).*

Claim 7.54. *For any vertex $u \in V(G) \setminus M'$, H_u is a tree.*

Proof. Let us first prove that $\mathcal{D}_u$ does not contain a crossing, meaning that there are no vertices $v_1, v_2 \in V(G)$ such that $\mathcal{D}_u$ contains a point in $\partial\mathcal{D}_{v_1} \cap \partial\mathcal{D}_{v_2}$. Suppose by contradiction that this is the case. Clearly, uv_1v_2 is a triangle so, by the definition of M', $\{v_1, v_2\} \subseteq M'$. However, this then contradicts Property (c) of Claim 7.52.

Now, as $\mathcal{D}_u$ does not contain a crossing, the family $\{\mathcal{D}_u \cap \mathcal{D}_v : v \in N(u)\}$ is a laminar set family. This implies that H_u is a tree (where in particular the leaves of H_u correspond to regions $\mathcal{D}_u \cap \mathcal{D}_v$, where $\mathcal{D}_u \cap \mathcal{D}_v$ is inclusion-wise minimal among the laminar family). □

We can now proceed with the kernelization, that divides into two parts. The first part (see Claim 7.4.4) deals with the inner-M'-vertices, while the second part deals with border- and outer-M'-vertices. The second part is split into three sections. In Claim 7.4.4, we will study the properties of border-M'-vertices. In Claim 7.4.4, we will provide kernelizations rules, and we will prove that the kernel is equivalent to (G, k) with respect to the FVS problem. Then in Claim 7.4.4, we will prove the properties of the kernel (G', k') stated in Lemma 7.50, in particular its small size.

Inner-M'-vertices

The purpose of this section is to reduce the number of inner-M'-vertices. We distinguish two cases according to the maximum degree of H_u. We begin with the vertices $u \in I_{M'}$ such that $\Delta(H_u) \geq 3$. Actually, for these vertices, we do not need to reduce their number. Indeed, we can bound their number by using the following consequence of Euler's formula.

Lemma 7.55 (see [CFK+15, Lemma 9.24])**.** *For any bipartite planar graph $G = (A, B, E)$, if $d(v) \geq 3$ for every $v \in A$, then we have that $|A| \leq 2|B|$.*

Claim 7.56. *The number of vertices $u \in I_{M'}$ such that $\Delta(H_u) \geq 3$ is $\mathcal{O}(p^3 k)$.*

Proof. To see this, let us construct a bipartite pseudo-disk graph as follows. In one part, we consider the arcs of $\overrightarrow{P_{S'}}$. These arcs are slightly shortened (so that they form an independent set) and slightly thickened (so that they intersect the adjacent regions we define below), see Figure 7.9. In the other part, we define a pseudo-disk $\mathcal{D}'_u$ for each vertex $u \in I_{M'}$ such that $\Delta(H_u) \geq 3$ as follows. Consider a vertex x of H_u such that $d_{H_u}(x) \geq 3$, and let $\mathcal{D}'_u$ be the intersection of $\mathcal{D}_u$ and the face of $\overrightarrow{P_{S'}}$ containing x. Since triangle-free pseudo-disk graphs are planar [Kra96], by Lemma 7.55, the second part has size $\mathcal{O}(|E(\overrightarrow{P_{S'}})|) = \mathcal{O}(p|M'|) = \mathcal{O}(p^3 k)$. $\square$

We can now focus on vertices $u \in I_{M'}$ such that $\Delta(H_u) \leq 2$ (see Figure 7.10). In that case, H_u is a path (with possibly only one vertex) of bounded length, as we show now.

Claim 7.57. *For any vertex $u \in I_{M'}$ such that H_u is a path, this path has length at most $d(u)$, and $d(u) \leq 2p$. Furthermore, $N(u)$ can be split into two cliques of size at most p each.*

Proof. For any neighbor v of u, $\mathcal{D}_u$ intersects $\partial\mathcal{D}_v$ at most once as otherwise $\partial\mathcal{D}_v$ and $\partial\mathcal{D}_u$ would intersect on more than two points which is not allowed in systems of pseudo-disks. Hence the number of edges in H_u (i.e. the length of H_u) is at most $d(u)$.

Furthermore, since $\Delta(H_u) \leq 2$, we have that any neighbor v of u is such that $\mathcal{D}_v$ contains one of the faces F_1 and F_2 of $\overrightarrow{P_{S'}}$ (see the left of Figure 7.10)

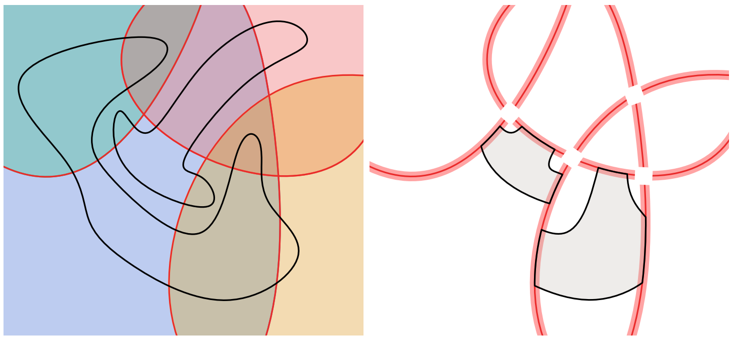

Figure 7.9: Two inner-M' pseudo-disks with $\Delta(H_u) \geq 3$, and the bipartite pseudo-disk graph constructed to prove Claim 7.56.

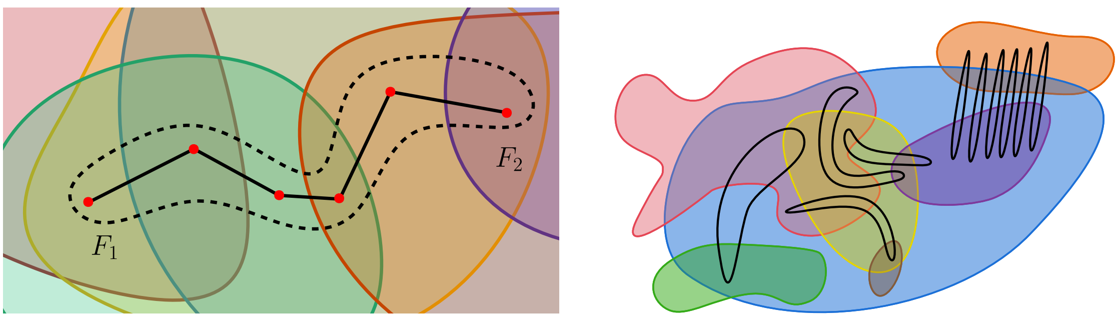

Figure 7.10: Left: a hosted graph H_u that is a path. Right: some vertices $u \in I_{M'}$ such that H_u is a path. Some of these vertices can have many twins.

corresponding to the endpoints of the path H_u in $\overrightarrow{P^*_{S'}}$. Since the representation has ply at most p, there are at most $2p$ pseudo-disks containing one (or two) of these faces. Finally, to split the neighborhood of u into two cliques, it suffices to consider the pseudo-disks containing F_1, which clearly induce a clique, and the remaining ones, since all of them contain F_2. $\square$

To bound the number of such vertices u with $\Delta(H_u) \leq 2$, we need to use the following kernelization rule for the FVS problem.

(R0) Consider two disjoint and possibly empty subsets $A, B \subseteq M'$ inducing complete graphs, and six isolated vertices of $G - M'$, $v_1, \ldots, v_5$ and u. If $N_G(u) \subseteq N_G(v_i) \subseteq A \cup B$, for $1 \leq i \leq 5$, delete vertex u and keep the same parameter k.

Proof. (Safeness of the rule) Let $G' = G - u$, and note that it cannot have a minimum feedback vertex set larger than G. Hence, it is sufficient to show that G' has a minimum feedback vertex set X, that is also a feedback vertex set for G.

Let $A_u = A \cap N_G(u)$ and $B_u = B \cap N_G(u)$. Consider any minimum feedback vertex set X of G', and let us transform it (if needed) into a minimum feedback vertex set including all the vertices of $A_u \cup B_u$, except possibly one. Observe first that $G[A]-X$ and $G[B]-X$ being acyclic, we have that $|A \setminus X| \leq 2$ and $|B \setminus X| \leq 2$.

If $|A_u \setminus X| = 2$, then $\{v_1, \ldots, v_5\} \subseteq X$, as otherwise the two vertices in $A_u \setminus X$ and any v_i not in X would form a triangle in $G' - X$. In such case, defining X' by replacing in X the vertices $v_1, \ldots, v_5$ with the vertices in $A \setminus X$ and in $B \setminus X$ would result in the desired feedback vertex set. Indeed, the vertices $v_1, \ldots, v_5$ are isolated in $G' - X'$, and $G' - \{v_1, \ldots, v_5\} - X'$ is a forest, as $X \subseteq X' \cup \{v_1, \ldots, v_5\}$.

Hence, we consider that $|A_u \setminus X| \leq 1$ and $|B_u \setminus X| \leq 1$. If $|A_u \setminus X| = 1$ and $|B_u \setminus X| = 1$, then $|\{v_1, \ldots, v_5\} \cap X| \geq 4$, as otherwise the two vertices in $\{v_1, \ldots, v_5\} \setminus X$, the vertex in $A_u \setminus X$ and the vertex in $B_u \setminus X$ would form a 4-cycle in $G' - X$. In such case, defining X' by replacing in X four vertices in $\{v_1, \ldots, v_5\} \cap X$ with the vertices in $A \setminus X$ and in $B \setminus X$ would result in the desired feedback vertex set. Indeed, the vertices $v_1, \ldots, v_5$ not in X' are isolated in $G' - X'$, and $G' - \{v_1, \ldots, v_5\} - X'$ is a forest, as $X \subseteq X' \cup \{v_1, \ldots, v_5\}$.

We thus have a minimum feedback vertex set X of G' including all the vertices of $A_u \cup B_u$, except possibly one. Since u has degree zero or one in $G - X$, we have that $G - X$ is the forest $G' - X$ with a new vertex u, that is either a leaf or an isolated vertex, and it is thus a forest. $\square$

Identifying the configuration required by the previous rule and updating the representation of G when we delete vertices can be done in polynomial time (see Lemma 7.47), implying that this kernelization can be performed in time polynomial in $|V(G)|$.

Note that from now on, among the vertices $u \in I_{M'}$ such that H_u is a path (as their neighborhoods induce two complete graphs by Claim 7.57), there are at most five vertices with the same neighborhood. We can thus focus on bounding the number of such vertices u with distinct neighborhoods. Since these vertices have bounded degree (by Claim 7.57 again), Theorem 4.12 bounds the number of inner-M'-vertices by $\mathcal{O}(|M'|p^3)$, but the following claim allows a small improvement. Let us first note that every inner-M'-vertex was already an inner-M-vertex.

Claim 7.58. *For every vertex $u \in I_{M'}$, we have that $\mathcal{D}_u$ is contained in $\mathcal{R}_M = (\cup_{v \in M} \mathcal{D}_v)$.*

Proof. Suppose, towards a contradiction, that for some vertex $u \in I_{M'}$, we have $\mathcal{D}_u \nsubseteq \mathcal{R}_M$. Let x be a point of $\mathcal{D}_u \setminus \mathcal{R}_M$ arbitrarily close to $\mathcal{R}_M$. As $\mathcal{D}_u \subseteq \mathcal{R}_{M'}$, there is a vertex $w \in M' \setminus M$ such that $x \in \mathcal{D}_w$. the point x being arbitrarily close to $\mathcal{R}_M$, there is a vertex $v \in M$ such that $\mathcal{D}_u$, $\mathcal{D}_v$, and $\mathcal{D}_w$ intersect. This contradicts the fact that every triangle of G has at least two vertices in M. □

A vertex $u \in I_{M'}$ such that H_u is a path has degree at most $2p$ (by Claim 7.57), and all of them were in M (by Claim 7.58). Hence, Theorem 4.12 implies that there are at most $\mathcal{O}(|M|p^3)$ such vertices with distinct neighborhoods. Thus, together with Claim 7.56 we have the following bound.

Claim 7.59. *After the kernelization of Claim 7.4.4, the number of inner-M'-vertices is $\mathcal{O}(p^4k)$.*

Properties of Border-M'-vertices

With $\mathcal{S}$ fixed, we define the ***M'-ply*** of a point of the plane as the number of pseudo-disks from M' that contain it. We consider the border-M'-vertices v, that is, those such that $\mathcal{D}_v$ intersects $\mathcal{R}_{M'}$, without being included in it. For such vertices, we can strengthen Claim 7.54, but we have to orient the edges of the hosted graphs H_u. The ***directed hosted graph*** $\overrightarrow{H_u}$ is an orientation of H_u such that the edge ab is oriented from the vertex with lower M'-ply towards the other one. This is always possible as in a pseudo-disk representation, any two incident faces have different ply.

An ***arborescence*** is an oriented tree with exactly one source, called the root.

Claim 7.60. *For any vertex $u \in B_{M'}$, the directed hosted graph $\overrightarrow{H_u}$ is an arborescence whose root is drawn in a face of $\overrightarrow{P_{\mathcal{S}'}}$ of M'-ply zero (see an example in the left of Figure 7.11).*

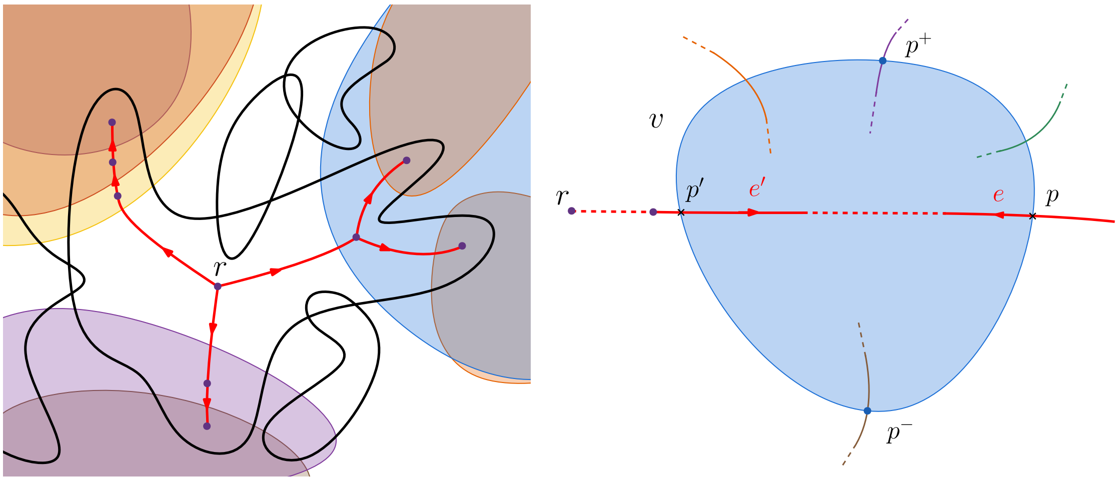

Figure 7.11: (left) The filled pseudo-disks belong to M', and the empty one correspond to a tree of $G - M'$ with vertices in $B_{M'} \cup O_{M'}$. For one of these vertices $u \in B_{M'}$, the graph $\overrightarrow{H_u}$ is drawn inside $\mathcal{D}_u$. (right) Proof of Claim 7.60.

Proof. One consequence of Claim 7.52.(e) is that $\overrightarrow{H_u}$ has exactly one vertex corresponding to a M'-ply zero face in $\overrightarrow{P_{S'}}$, call it the root r. Towards a contradiction, assume there is an arc e pointing in the wrong direction, that is, towards r. Since r has M'-ply zero, the pseudo-disk $\mathcal{D}_v$ whose border is crossed by e, is also crossed in the path linking r and e in $\overrightarrow{H_u}$ (see the right of Figure 7.11). Denote p and p' the crossing point between $\partial\mathcal{D}_v$ and the arcs e and e', respectively. Since e and e' are distinct arcs, the two parts of $\partial\mathcal{D}_v \setminus \{p, p'\}$ are crossed by other borders of M', at p^+ and p^- say. Since $\mathcal{D}_u$ cannot contain these points p^+ and p^- (by the definition of M'), but contains p and p', we have that $\partial\mathcal{D}_u$ and $\partial\mathcal{D}_v$ intersect at least four times (in each of the arcs of $\partial\mathcal{D}_v \setminus \{p, p^-, p', p^+\}$), a contradiction. □

We have to define new plane graphs. For any vertex $u \in B_{M'}$, the digraph $\overrightarrow{H'_u}$ is a plane arborescence drawn in the vicinity of $\overrightarrow{H_u}$, and it is such that

- $\overrightarrow{H'_u}$ has the same root and same leaves as $\overrightarrow{H_u}$, but
- it is a subdivision of an out-star whose paths from r to any leaf ℓ have the same length as in $\overrightarrow{H_u}$.

This graph can have several vertices in the same face of $\overrightarrow{P_S}$, but each arc intersects a single border of M', at a single point in the vicinity of $\overrightarrow{H_u}$ (see the left and the middle of Figure 7.12).

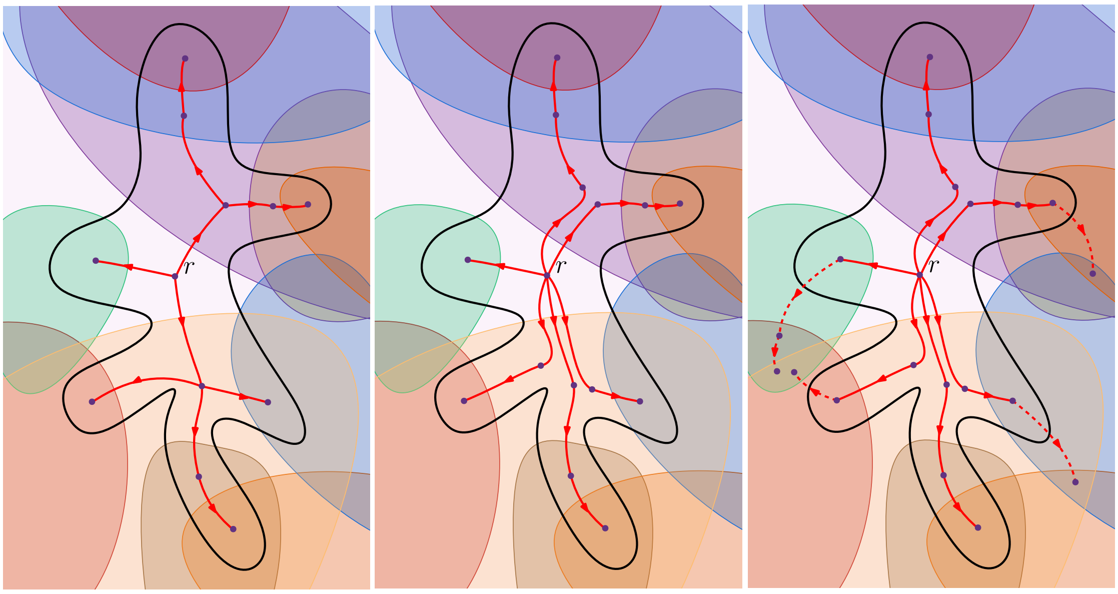

Figure 7.12: Example of graphs $\overrightarrow{H_u}$, $\overrightarrow{H'_u}$, and $\overrightarrow{H''_u}$.

We now have to prolong these graphs $\overrightarrow{H'_u}$.

Claim 7.61. *In polynomial time, the drawn arborescences $\overrightarrow{H'_u}$, for all $u \in B_{M'}$, can be prolonged into arborescences $\overrightarrow{H''_u}$ fulfilling the following properties (see an example in the right of Figure 7.12).*

- *The root of $\overrightarrow{H''_u}$ is the same as $\overrightarrow{H'_u}$.*
- *$\overrightarrow{H''_u}$ is the subdivision of an out star.*
- *Each leaf of $\overrightarrow{H''_u}$ is drawn in a face of $\overrightarrow{P_{S'}}$ whose M'-ply is locally maximal (i.e. such that all the incident faces have lower M'-ply).*
- *Each edge of $\overrightarrow{H''_u}$ crosses exactly one edge of $\overrightarrow{P_{S'}}$, at a single point, and it is oriented from the smaller M'-ply towards the other end.*
- *When considering all the drawings $\overrightarrow{H''_u}$ for all the $u \in B_{M'}$ there are no intersections.*

Proof. All these properties, except the third one, are fulfilled by the original $\overrightarrow{H'_u}$. Thus, we only have to show how to prolong an arborescence if one of its leaves ℓ

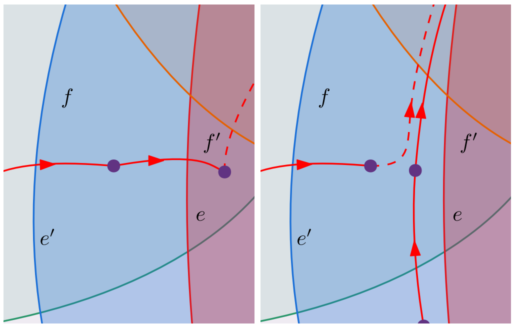

Figure 7.13: In red, the graph $\overrightarrow{H'_u}$. Left: the case where e is reachable. Right: the case where e is not reachable.

is not in a face with locally maximal M'-ply. Around the face f of $\overrightarrow{P_{S'}}$ containing ℓ, there is an edge e of $\overrightarrow{P_{S'}}$, incident to a face f' with larger M'-ply. If it is possible without intersecting an arc of a $\overrightarrow{H'_v}$, for some $v \in B_{M'}$, draw an arc towards f' crossing only e (see the left of Figure 7.13). If it is not possible, then our drawing was blocked by some $\overrightarrow{H'_v}$ (possibly $v = u$), and in that case we just have to follow the arcs of $\overrightarrow{H'_v}$ towards the next face f'' of $\overrightarrow{P_{S'}}$ (see the right of Figure 7.13). $\square$

Definition 7.62. *Given a vertex $u \in B_{M'}$, the corresponding $\overrightarrow{H''_u}$, and a root-to-leaf path P'' of $\overrightarrow{H''_u}$, we denote type(P'') the directed path in $\overrightarrow{P^*_{S'}}$ corresponding to P''. We say that type(P'') is the* **type** *of P''. Then, types(u) $= \{$type(P'') : P'' is a root-to-leaf path of $\overrightarrow{H''_u}\}$. Given a tree T of $G - M'$, with $V(T) \subseteq B_{M'} \cup O_{M'}$, the set types($T$) is defined by types($T$) $= \bigcup_{u \in V(T) \cap B_{M'}}$ types(u). The cardinalities of these sets are denoted #types(u) and #types(T). Finally, let $\mathbb{T}$ be the set $\mathbb{T} = \bigcup_{u \in B_{M'}}$ types(u).*

Note that by construction of $\overrightarrow{H''_u}$, #types(u) corresponds to the number of leaves in $\overrightarrow{H_u}$. Any face f of $\overrightarrow{P_{S'}}$ with locally maximal M'-ply, only contains leaves of the graphs $\overrightarrow{H''_u}$, for $u \in B_{M'}$. When going around the border of such face f, one crosses, with a given order, the ending arcs of several root-to-leaf paths of some graphs $\overrightarrow{H''_u}$. This order is strongly related with the type of these root-to-leaf paths.

Claim 7.63. *Given a face f of $\overrightarrow{P_{S'}}$ with locally maximal M'-ply, and given any path $P \in \mathbb{T}$ ending in f, all the root-to-leaf paths of type P end at f, and these paths are crossed consecutively when going around f. In other words, if going around f one crosses, in this order, four root-to-leaf paths $P''_1, P''_2, P''_3, P''_4$ of some graphs $\overrightarrow{H''_u}$, for*

$u \in B_{M'}$ (these paths can belong to one, two, three or four different graphs $\overrightarrow{H''_u}$), we cannot have that type$(P''_1) = P_a$, type$(P''_2) = P_b$, type$(P''_3) = P_a$, and type$(P''_4) = P_c$, for any triple $P_a, P_b, P_c \in \mathbb{T}$ such that $P_a \neq P_b$ and $P_a \neq P_c$.

Proof. Note that if $\mathsf{type}(P''_1) = \mathsf{type}(P''_3)$, then P''_1 and P''_3 go through the same faces of $\overrightarrow{P_{S'}}$, in the same order, and only visit one M'-ply zero face, the starting one. Hence, one of P''_2 or P''_4 has to visit exactly the same faces, and is thus of the same type (see Figure 7.14). □

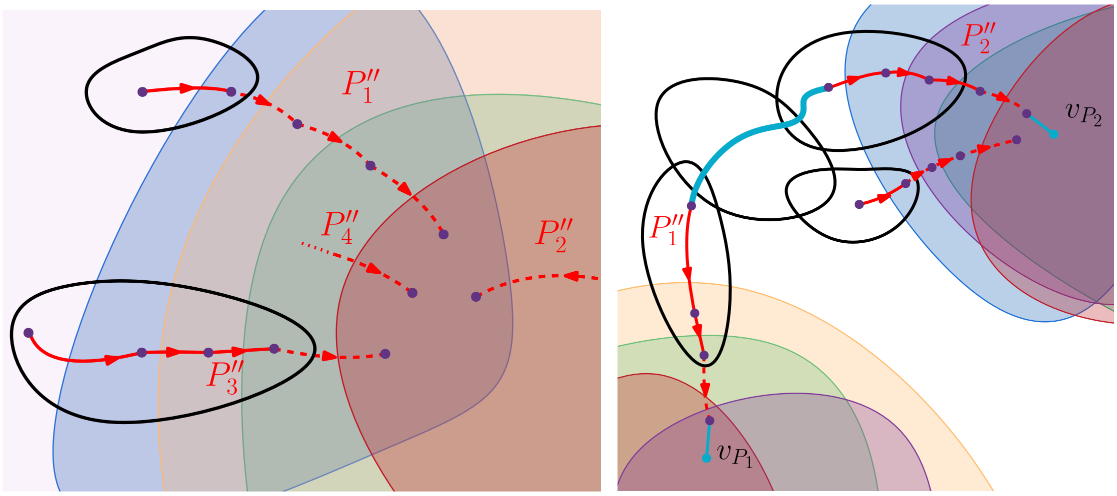

Figure 7.14: Illustration for the proofs of Claim 7.63 and Claim 7.65.

We now show that the number of types is small.

Claim 7.64. *The set $\mathbb{T}$ has size $\mathcal{O}(p|M'|)$.*

Proof. Consider the bipartite plane multigraph $B = (F_0, F_{\max}, \mathbb{T})$ defined as follows. The part F_0 corresponds to the faces of $\overrightarrow{P_{S'}}$ with M'-ply zero, the part $F_{\max}$ to those with locally maximal M'-ply, and there is an edge from $f \in F_0$ to $f' \in F_{\max}$ for each path P of $\mathbb{T}$ linking these two faces. These edges can be drawn by following one of the directed paths P'' such that $\mathsf{type}(P'') = P$. The planarity of B follows from the fact that such paths P'' are not crossing.

This graph has $\mathcal{O}(p|M'|)$ vertices (since those are faces of $\overrightarrow{P_{S'}}$), and $\mathcal{O}(p|M'|)$ faces of degree two (as any such face contains a vertex of $\overrightarrow{P_{S'}}$). By Euler's formula the number of edges is also $\mathcal{O}(p|M'|)$. □

We are now going to consider pairs of paths of $\mathbb{T}$, which appear in the same set $\mathsf{types}(T)$, for some tree T of $G - M' - I_{M'}$.

Claim 7.65. *There are $\mathcal{O}(p|M'|)$ pairs $\{P_1, P_2\} \subseteq \mathbb{T}$ such that types$(T) = \{P_1, P_2\}$, for some tree T of $G - M' - I_{M'}$.*

Proof. We are again going to construct a planar graph (actually a plane graph). Its vertices v_P are in bijection with the paths $P \in \mathbb{T}$, and they are drawn in the face of $\overrightarrow{P_{S'}}$ corresponding to the high M'-ply endpoint of P. The vertices v_P in the same face f are drawn close to its border, in the same order around, as the one given by Claim 7.63.

Then we draw an edge from v_{P_1} to v_{P_2} for each pair $\{P_1, P_2\} \subseteq \mathbb{T}$ such that types$(T) = \{P_1, P_2\}$, for some tree T of $G - M' - I_{M'}$. To draw such an edge, consider a given tree T of $G - M' - I_{M'}$ such that types$(T) = \{P_1, P_2\}$. Then, consider paths P_1'', P_2'' of $\overrightarrow{H''_{u_1}}$, $\overrightarrow{H''_{u_2}}$ for some vertices u_1, u_2 of T (possibly $u_1 = u_2$) such that type$(P_1'') = P_1$ and type$(P_2'') = P_2$. The edge from v_{P_1} to v_{P_2} can then be drawn by following P_1'' and P_2'' and connecting them inside of the union of the pseudo-disks of T (see Figure 7.14). Since the considered trees are not adjacent (they are connected components of $F - M' - I_{M'}$), the unions of their respective pseudo-disks are pairwise disjoint hence the edges drawn as described above do not cross. This shows that the graph we construct is planar. Finally this graph being planar and simple its number of edges is linear in terms of its number of vertices. Hence, there are $\mathcal{O}(|\mathbb{T}|) = \mathcal{O}(p|M'|)$ pairs $\{P_1, P_2\} \subseteq \mathbb{T}$ such that types$(T) = \{P_1, P_2\}$, for some tree T of $G - M' - I_{M'}$. $\square$

In the next section, we will bound the number of trees T of $G - M' - I_{M'}$ such that types$(T) = \{P_1, P_2\}$ for some pair $\{P_1, P_2\}$, and we will bound their size. We will also have to deal with trees T such that #types$(T) \geq 3$. This is why we need the following property.

Claim 7.66.

$$\sum_{\substack{T \in G - M' - I_{M'} \\ \text{with } \#\text{types}(T) \geq 3}} \#\text{types}(T) = \mathcal{O}(p|M'|).$$

Proof. Again, this is shown by constructing a planar graph. Here, it will be simple and bipartite, with one part containing one vertex v_T per tree $T \in G - M' - I_{M'}$ such that #types$(T) \geq 3$, and one part containing one vertex v_P per path $P \in \mathbb{T}$. The vertices v_P are drawn as in the proof above, while the vertices v_T are drawn in the M'-ply zero face of $\overrightarrow{P_{S'}}$ intersecting the pseudo-disks of T (by Claim 7.52 Property (e)).

Then we draw an edge from v_T to v_P if and only if $P \in$ types(T). To draw such an edge, consider a path P'' of $\overrightarrow{H''_u}$ for some vertices u of T such that type$(P'') = P$. Then starting from v_P we can follow P'' and then continue inside of the union of the

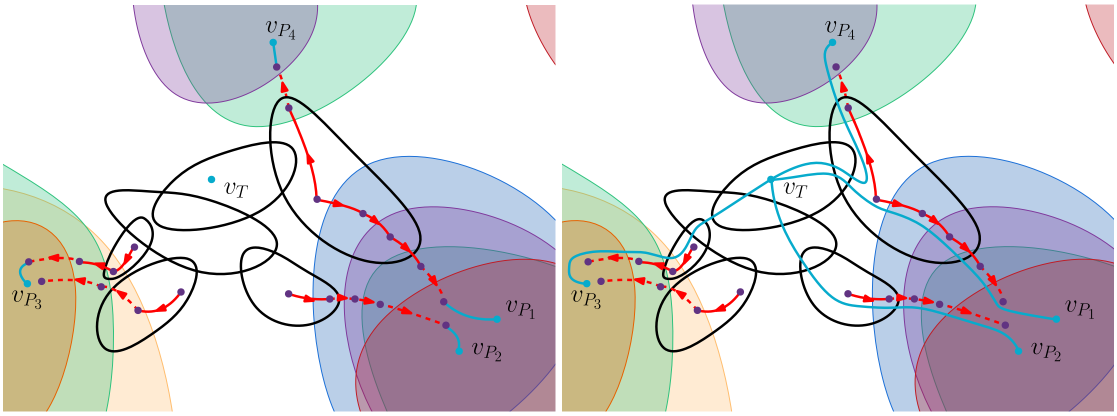

Figure 7.15: Illustration for the proof of Claim 7.66.

pseudo-disks of T up to v_T, similarly to what we did in the proof of Claim 7.65 (see Figure 7.15). Finally, this graph being bipartite planar and simple, Lemma 7.55 allows us to conclude. $\square$

In the next section, we will bound the size of any tree T of $G - M' - I_{M'}$ by $\mathcal{O}(\#\text{types}(T))$, and together with the previous claims, this will allow us to bound $|B_{M'} \cup O_{M'}|$. Towards this goal, we have to introduce the following notions.

Given a path (type) $P \in \mathbb{T}$, we denote by K_P the q-tuple $(v_1, \ldots, v_q)$, where $v_i \in M'$ is the i^{th} pseudo-disk entered by P, and thus q is the M'-ply of the final vertex of P.

Note that since P is a directed path and each arc goes from a region of lower M'-ply to a region of higher M'-ply, $\{v_1, \ldots, v_q\}$ actually forms a clique, hence the name K_P. Let $\mathcal{K} = \{K_P : P \in \mathbb{T}\}$.

Definition 7.67. *Given a q-clique of $G[M']$, for some $q \leq p$, and an ordering of its vertices $K = (v_1, \ldots, v_q)$ (here K is a q-tuple), a vertex u has* **K-monotone neighborhood** *if there is an integer q_u such that:*

$$N(u) \cap M' = \{v_1, \ldots, v_{q_u}\}.$$

In this context, q_u is the **K-index** *of vertex u. Given a q-clique and a q'-clique of $G[M']$ (possibly sharing some vertices), for some $q, q' \leq p$, and two orderings of their vertices $K = (v_1, \ldots, v_q)$ and $K' = (v'_1, \ldots, v'_{q'})$, a vertex u has* **(K, K')-monotone neighborhood** *if there exist two integers q_u, q'_u such that:*

$$N(u) \cap M' = \{v_1, \ldots, v_{q_u}\} \cup \{v'_1, \ldots, v'_{q'_u}\}.$$

In this context, q_u and q'_u are the **K-index** *and the* **K'-index** *of vertex u, respectively.*

Let us see how #types(u) relates to monotone neighborhoods.

Claim 7.68. *Given a vertex $u \in B_{M'}$ such that #types$(u) = 1$, and given $K = (v_1, \ldots, v_q)$ the ordered list of the borders of M' crossed by the unique path $P \in$ types(u). Then the vertex u has a K-monotone neighborhood.*

Proof. This clearly follows from the fact that $\overrightarrow{H''_u}$ is constructed by prolonging the unique path of $\overrightarrow{H'_u}$, and the fact that $N(u) \cap M'$ is exactly the set of borders crossed by this path of $\overrightarrow{H'_u}$. Note that if the last border crossed by $\overrightarrow{H'_u}$ is the border of v_i, then the K-index of u, q_u, is i.

□

Claim 7.69. *Given a vertex $u \in B_{M'}$ such that #types$(u) = 2$, and given the ordered lists of the borders of M' crossed by the two paths in types(u), $K = (v_1, \ldots, v_q)$ and $K' = (v'_1, \ldots, v'_{q'})$. Then the vertex u has a (K, K')-monotone neighborhood.*

Proof. Again, this clearly follows from the fact that $\overrightarrow{H''_u}$ is constructed by prolonging the two paths of $\overrightarrow{H'_u}$, and the fact that $N(u) \cap M'$ is exactly the set of borders crossed by these paths of $\overrightarrow{H'_u}$. Again, the last borders crossed by the branches of $\overrightarrow{H'_u}$ are those of v_{q_u} and $v'_{q'_u}$, where q_u, q'_u are the K- and K'-indices of u. □

Kernelization rules

Here, we provide kernelization rules for FVS that will allow us to bound the size of the trees in $G - M' - I_{M'}$. Most of these rules are related to the paths in $\mathbb{T}$ and to the K- and (K, K')-monotone-neighborhood property satisfied by the vertices $u \in B_{M'}$, such that #types$(u) = 1$ or 2, respectively. As the kernelization has to be performed in polynomial time (in terms of $n = |V(G)|$), we observe that within such a time, one can compute the graphs $\overrightarrow{H_u}$, $\overrightarrow{H'_u}$, and $\overrightarrow{H''_u}$, and the sets type(P''), types(u), types(T), $\mathbb{T}$, $\mathcal{K}$. Furthermore, note that updating these sets when a vertex $v \in M'$ or $u \in V(G) \setminus M'$ is deleted, can also be performed in polynomial time (by Lemma 7.47). Finally, note that such update is also possible when contracting an edge uu' of $G - M'$ (by Lemma 7.47 also).

Let us consider the following kernelization rules for FVS. Those are applied prioritizing the first rules. As those are folklore, we omit the proofs of the two first rules.

(R1) Delete every vertex v with degree at most one, and maintain the parameter k.

(R2) Consider two adjacent vertices $u, u' \notin M'$, such that $d(u) = 2$, and such that u and u' do not have a common neighbor. Then, contract the edge uu', and maintain the parameter k.

It should be noted that after applying (R2), the property that every triangle has at least two vertices in M' might be lost. Note that the rule (R2) implies that from now on, any degree two vertex $u \notin M'$ is in a triangle, or has its two neighbors in M'. Hence in particular, any degree two vertex $u \notin M'$ has at least one neighbor in M'. The following five rules deal with vertices having K-monotone neighborhoods with $K \in \mathcal{K}$.

(R3) Consider a q-tuple $K = (v_1, \ldots, v_q) \in \mathcal{K}$, and two isolated vertices u, w of $G - M'$ having K-monotone neighborhoods. If their K-indices are such that $1 \leq q_u \leq q_w$, delete u and maintain the parameter k.

Proof. (Safeness of (R3)) Let us prove that if $G - u$ has a feedback vertex set of size k, so does G. Consider a feedback vertex set X. Recall that any clique has all its vertices in X, except at most two of them. If $w \in X$, replace it with some vertex of $K \setminus X$, if any. Now we have that there is at most one vertex in $N(w) \setminus X$ (and thus in $N(u) \setminus X$). In other words, u and w have degree at most one in $G - X$. Thus, since $(G - \{u, w\}) - X$ is a forest (as $X \cup \{w\}$ is a feedback vertex set), so is $G - X$. □

(R4) Consider a q-tuple $K = (v_1, \ldots, v_q) \in \mathcal{K}$, and a tree T of $G - M'$ whose vertices have K-monotone neighborhoods. If T has at least two vertices, delete v_1 and decrease the parameter k by one.

Proof. (Safeness of (R4)) As T has at least two vertices, it has at least two leaves, say a and b. Since the rule (R1) does not apply for a or b, these vertices are both adjacent to v_1. Consider a feedback vertex set X not containing v_1. Since there is at least one cycle in $\{v_1\} \cup V(T)$, this feedback vertex set intersects $V(T)$ on at least one vertex, denoted x. Let us prove that replacing x with v_1 in X, we still have a feedback vertex set. Towards a contradiction, we consider a cycle C not hit by this set. This cycle C goes through x and through some vertices of $\{v_1, \ldots, v_q\}$, since T is acyclic. However, since our new hitting set contains at least (and thus exactly) $q - 1$ vertices among $\{v_1, \ldots, v_q\}$, the cycle has two vertices $a', b' \in V(T)$ adjacent to some vertex $v_i \in K \setminus X$, with $i > 1$. Note that X does not hit one of the triangles $a'v_1v_i$ or $b'v_1v_i$, as $a' \neq x$ or $b' \neq x$, a contradiction. We can thus force v_1 in the feedback vertex set. □

Claim 7.70. *After the kernelization described above, there are $\mathcal{O}(p|M'|)$ vertices in trees T of $G - M' - I_{M'}$ such that $\#types(T) \leq 1$.*

Proof. Rule (R4) ensures us that every such tree has at most one vertex. Rule (R3) ensures that there is at most one such tree such that $\mathsf{types}(T) = \{P\}$, for any given $P \in \mathbb{T}$. Thus, in total, there are at most $|\mathbb{T}| = \mathcal{O}(p|M'|)$ such vertices (By Claim 7.64). $\square$

The following rules deal with the connected components of $G - M'$. This subgraph being acyclic, these connected components are trees. Note that the leaves in these trees have degree at least two in G (by rule (R1)), so they are necessarily adjacent to vertices in M'.

(R5) Consider a q-tuple $K = (v_1, \ldots, v_q) \in \mathcal{K}$, and a tree T of $G - M'$ (which is non-necessarily a connected component of $G - M'$), whose vertices have K-monotone neighborhoods, and a particular vertex $r \in V(T)$, such that for every other vertex $u \in V(T) \setminus \{r\}$, $N(u) \subseteq K \cup V(T)$. If there are two vertex disjoint paths in $T - \{r\}$ with endpoints a, b, c, d adjacent to v_1 (i.e. such that $1 \leq q_a, q_b, q_c, q_d$), delete v_1 and decrease the parameter k by one.

Proof. (Safeness of (R5)) Let us show that if G has a feedback vertex set of size k it has one containing v_1. Consider a minimum feedback vertex set X not containing v_1. The two cycles formed by the two paths and v_1 intersect only on v_1, thus X must intersect each of these paths. Let $x, x' \in X$, be two vertices of these paths. We define X' as obtained by replaying in X the vertices x and x' with the at most two vertices of $K \setminus X$. Note that now $K \subseteq X'$, and in particular $v_1 \in X'$. Since $(G - (V(T) \setminus \{r\})) - X'$ is a forest (as $X' \cup \{x, x'\}$ is a feedback vertex set), and since T is a tree where only one vertex, r, has neighbors outside $V(T) \cup K$, the graph $G - X'$ is also a forest. Rule (R5) is thus safe. $\square$

(R6) Consider a q-tuple $K = (v_1, \ldots, v_q) \in \mathcal{K}$, a vertex r of $G - M'$ adjacent to three leaves a, b, c of $G - M'$, that have K-monotone neighborhoods. If their K-indices are such that $1 \leq q_a \leq q_b \leq q_c$, delete a and maintain the parameter k.

Proof. Let us show that if $G - a$ has a feedback vertex set of size k, X, so does G. Inducing a clique, the set $K_a = (v_1, \ldots, v_{q_a})$ has at most two vertices not in X. While possible, replace any vertex of $V(T) \setminus \{r\}$ in X, by a vertex of K_a. After this, we have every vertex of K_a in X, or we have one vertex in $K_a \setminus X$, v_i, and none of the vertices of $V(T) \setminus \{r\}$ in X. In the latter case, we necessarily have $r \in X$. In both cases, the set X is a feedback vertex set for G also. Indeed, if $K_a \subseteq X$, vertex a has at most one neighbor in $G - M'$, r, so it cannot create a cycle. If $K_a \subseteq X = \{v_i\}$ and $r \in X$, we also have that vertex a has at most one neighbor in $G - M'$, v_i here, so it cannot create a cycle. Rule (R6) is thus safe. $\square$

Note that from now on, for every $K \in \mathcal{K}$ and every vertex $v \in G - M'$, v is adjacent to at most two leaves of $G - M'$ that have K-monotone neighborhoods.

(R7) Consider a q-tuple $K = (v_1, \ldots, v_q) \in \mathcal{K}$, and a path P in $G - M'$ of length six. If every vertex $u \in V(P)$ has degree two in $G - M'$, and has a K-monotone neighborhood in M', delete v_1 and decrease the parameter k by one.

Proof. (Safeness of (R7)) Let us prove that if G has a feedback vertex set of size k, it has one containing v_1. Let X be a minimum feedback vertex set of G. By (R2), v_1 is adjacent to all the six vertices in P. The graph induced by $\{v_1\} \cup V(P)$ having three cycles intersecting only at v_1, we have that $v_1 \in X$ or that P has at least three vertices in X. In the first case, we are done. In the second case, we replace any two vertices of $V(P) \cap X$ by the (at most two) vertices in $K \setminus X$ (thus including v_1) and let us note that $V(P) \cap X$ still contains at least one vertex. Since X is a feedback vertex set of $G - V(P)$ (as $G - V(P) - X$ is a subgraph of the forest obtained from the initial feedback vertex set), and since in $G - X$ the vertices of $V(P) \setminus X$ induce paths that each connect with at most one edge to a vertex outside $V(P)$, we have that $G - X$ is indeed a forest. □

The last three rules deal with vertices having (K, K')-monotone neighborhoods with $K, K' \in \mathcal{K}$.

(R8) Consider a q- and a q'-tuple, $K = (v_1, \ldots, v_q)$ and $K' = (v'_1, \ldots, v'_{q'})$ of $\mathcal{K}$, and a length 60 path P in $G - M'$. If every vertex of P has degree two in $G - M'$, and has a (K, K')-monotone neighborhood, delete v_1 and v'_1, and then decrease the parameter k by two.

Proof. (Safeness of (R8)) Let us prove that if G has a feedback vertex set of size k, it has one containing v_1 and v'_1. Let X be a minimum feedback vertex set of G. First, note that if X has at least 5 vertices in P, then we can replace four of them in X with the vertices of $K \setminus X$, and of $K' \setminus X$. This would result in another minimum feedback vertex set.

By (R7) v_1 has at least 10 neighbors in P (otherwise (R7) would apply with v'_1 and a subpath of P) so the graph induced by $V(P) \cup \{v_1\}$ contains five cycles intersecting only on v_1. Thus, a feedback vertex set avoiding v_1 would intersect P on at least five vertices, a contradiction. Thus, we obtain that $v_1 \in X$. We similarly obtain that $v'_1 \in X$. □

We already bounded the number of vertices in trees T such that $\#\mathsf{types}(T) \leq 1$. We now focus on trees T such that $\#\mathsf{types}(T) = 2$. Given two tuples $K, K' \in \mathcal{K}$, let $F_{K,K'}$ be the subforest of $G - M' - I_{M'}$ obtained by keeping the connected components T such that $\mathsf{types}(T) = \{K, K'\}$.

A tree T in $F_{K,K'}$ is said ***trivial*** if it has only one vertex, say u. Clearly, this vertex has a (K, K')-monotone neighborhood. By rule (R0) presented in Claim 7.4.4, the number of trivial trees in $F_{K,K'}$ by $5p$ (as we have at most p incomparable couples (q_u, q'_u) of K- and K'-indices). A tree T in $F_{K,K'}$ is said ***semi-trivial*** if it has only two vertices adjacent to M', one having a K-monotone neighborhood, and the other having a K'-monotone neighborhood. Note that, by rule (R1) and (R2), T has only two vertices.

(R9) Consider a q- and a q'-tuple, $K = (v_1, \dots, v_q)$ and $K' = (v'_1, \dots, v'_{q'})$ of $\mathcal{K}$, and a semi-trivial connected component ab of $F_{K,K'}$, such that vertex a has K-monotone neighborhood, with K-index q_a, and such that vertex b has K'-monotone neighborhood, with K'-index q'_b. If $F_{K,K'}$ has five other semi-trivial connected components, each of them with K-index at least q_a, and with K'-index at least q'_b, delete a and b and maintain the parameter k.

Proof. (Safeness of (R9)) Let $G' = G - \{a, b\}$, and note that it cannot have a minimum feedback vertex set larger than G. Hence, it is sufficient to show that G' has a minimum feedback vertex set X, that is also a feedback vertex set for G.

Let $A = N_G(a) \setminus \{b\} = \{v_1, \dots, v_{q_a}\}$ and $B = N_G(b) \setminus \{a\} = \{v'_1, \dots, v'_{q'_b}\}$, and let a_i, b_i be the vertices of m_i that have K- and K'-monotone neighborhood, respectively. Consider any minimum feedback vertex set X of G', and let us transform it (if needed) into a minimum feedback vertex set including all the vertices of $A \cup B$, except possibly one. Observe first that $G[K] - X$ and $G[K'] - X$ being acyclic, we have that $|K \setminus X| \leq 2$ and $|K' \setminus X| \leq 2$.

If $|A \setminus X| = 2$, then each of $\{a_1, \dots, a_5\} \subseteq X$, as otherwise the two vertices in $A \setminus X$ and a_i would form a triangle in $G' - X$. In such case, defining X' by replacing in X the vertices $a_1, \dots, a_5$ with the vertices in $K \setminus X$ and in $K' \setminus X$ would result in the desired feedback vertex set. Indeed, $m_1, \dots, m_5$ form isolated components in $G - X'$, and $G' - \{a_1, \dots, a_5\} - X'$ is a forest, as $X \subseteq X' \cup \{a_1, \dots, a_5\}$.

Hence, we consider that $|A \setminus X| \leq 1$ and $|B \setminus X| \leq 1$. If $|A \setminus X| = 1$ and $|B \setminus X| = 1$, then X contains at least four vertices among $\{a_1, \dots, a_5\} \cup \{b_1, \dots, b_5\}$, as otherwise there would be a 6-cycle going through a_ib_i and a_jb_j in $G' \setminus X$, for some i, j. In such case, defining X' by replacing in X the four vertices in $X \cap (\{a_1, \dots, a_5\} \cup \{b_1, \dots, b_5\})$ with the vertices in $K \setminus X$ and in $K' \setminus X$ would result in the desired feedback vertex set. Indeed, $m_1, \dots, m_5$ form isolated components in $G' - X'$, and $G' - (\{a_1, \dots, a_5\} \cup \{b_1, \dots, b_5\} \cup X')$ is a forest, as $X \subseteq \{a_1, \dots, a_5\} \cup \{b_1, \dots, b_5\} \cup X'$.

We thus have a minimum feedback vertex set of G' including all the vertices of $A \cup B$, except possibly one. Since ab has zero or one incident edge towards $G' - X$, we have that $G - X$ is a forest. $\square$

As for trivial trees, this rule allows us to bound the number of semi-trivial trees in $F_{K,K'}$ by $5p$ (as we have at most p incomparable couples (q_a, q'_b) of K- and K'-indices).

(R10) If there are two tuples $K, K' \in \mathcal{K}$, such that $F_{K,K'}$ has four disjoint paths with endpoints adjacent to $v_1 \in K$, delete v_1 and decrease the parameter k by one.

Proof. (Safeness of (R10)) Indeed, every feedback vertex set X avoiding v_1 should intersect $F_{K,K'}$ on at least four vertices, but replacing these four vertices by the vertices of $K \setminus X$ and $K' \setminus X$, would result in another feedback vertex set with at most the same size, and containing v_1. We can thus force v_1 in the feedback vertex set. $\square$

Note that from now on, for every $K, K' \in \mathcal{K}$, the forest $F_{K,K'}$ has at most 6 connected components that are neither trivial, nor semi-trivial.

Claim 7.71. *For every pair $K, K' \in \mathcal{K}$, the forest $F_{K,K'}$ can be partitioned in two parts, one with at most $\mathcal{O}(p)$ vertices, and the other with at most six trees.*

In the following subsection, this claim will help us in bounding the size of the graph after the kernelization.

Properties of the kernel

Let us now refer to the obtained graph and parameter as G' and k', and note by construction that $k' \leq k$. As all rules are safe, we know that (G', k') is equivalent to (G, k). Showing that G' is a pseudo-disk graph with ply at most p is simple. The class of pseudo-disk graphs with ply at most p being closed under induced subgraphs, we only have to check that when applying Rule (R2), the graph remains a pseudo-disk graph with ply at most p, and this follows from Lemma 7.47. Now, it only remains to bound the size of G'.

Let us define the graph G^* from a weak pseudo-disk system[5] obtained from the representation of $G' - M'$ by adding a small disk $\mathcal{D}_{v_P}$ for each element $P \in \mathbb{T}$, and for every pseudo-disk $\mathcal{D}_u$ of $B_{M'}$, by prolonging $\mathcal{D}_u$ along $\overrightarrow{H''_u}$ in such a way to fulfill the following conditions (see Figure 7.16).

1. The prolongations do not intersect new pseudo-disks among $V(G') \setminus (M' \cup I_{M'})$.

2. The disks corresponding to $\mathbb{T}$ are disjoint.

[5]That is, a system of regions homeomorphic to disks, and such that for any couple $\mathcal{D}_u, \mathcal{D}_v$, it is (only) required that the region $\mathcal{D}_u \setminus \mathcal{D}_v$ is connected.

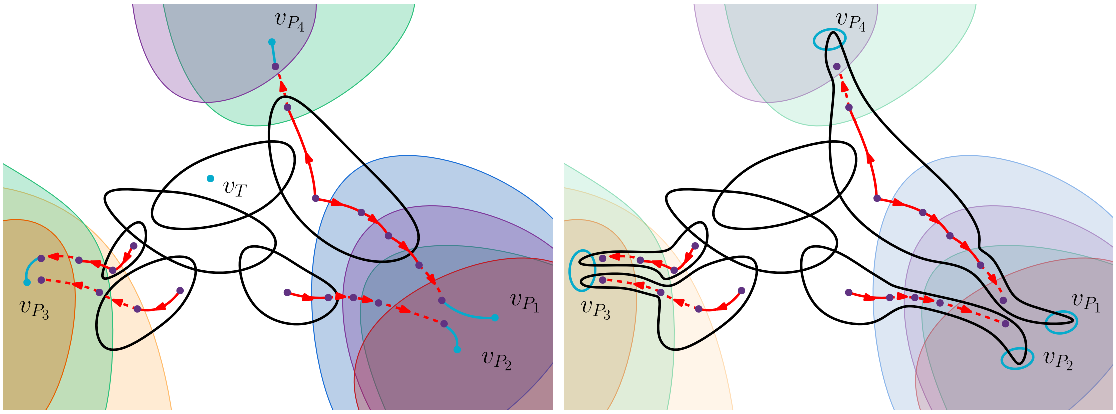

Figure 7.16: Construction of G^*.

3. Any (prolonged) disk $\mathcal{D}_u$ of a vertex $u \in B_{M'}$ intersects any disk $\mathcal{D}_{v_P}$, such that $P \in \mathsf{types}(u)$ (possible by Claim 7.63). Furthermore, each component of $\mathcal{D}_{v_P} \cap \mathcal{D}_u$ is a lens, that is, it is bordered by the union of a sub-arc of $\partial\mathcal{D}_{v_P}$ and a sub-arc of $\partial\mathcal{D}_u$.

Hence, we deduce that this intersection system is indeed a weak pseudo-disk one. Furthermore, since this representation has ply two, the graph G^* is planar [Kra96].

As $|V(G')| \leq |M'| + |V(G^*)|$, we are going to bound $|V(G')|$ by bounding $|V(G^*)|$. The main difficulty is to establish that for any tree T of $G' - M' - I_{M'}$ we have $|V(T)| = \mathcal{O}(\#\mathsf{types}(T))$. Before proving this in Claim 7.76, we have to introduce preliminary notions.

Consider an arbitrary connected component T of $G - M' - I_{M'}$ such that $\mathsf{types}(T) = \{P_1, \dots, P_d\} \subseteq \mathbb{T}$, for some $d \geq 2$. Then, root T at an arbitrary vertex r. For any vertex $v \in V(T)$ denote by T_v the rooted subtree of T rooted at v and let us denote by $\mathsf{types}^T(v)$ the set $\bigcup_{u \in V(T_v)} \mathsf{types}(u)$. Note that $\mathsf{types}^T(v) \subseteq \{P_1, \dots, P_d\}$. The vertex of G^* corresponding to a path $P \in \mathbb{T}$ will be denoted v_P. Two vertices u, v of T are said ***incomparable*** if $u \notin T_v$ and $v \notin T_u$.

The graph G^* being planar we have the following two properties.

Claim 7.72. *For any three, pairwise incomparable, vertices u, v, w of T, there is no pair $i, j \in \{1, \dots, d\}$ such that $\{P_i, P_j\}$ is a subset of $\mathsf{types}^T(u)$, $\mathsf{types}^T(v)$, and $\mathsf{types}^T(w)$.*

Proof. Indeed, otherwise by contracting T_u, T_v, T_w, and the rest of T on u, v, w, and r, respectively, these four vertices with the vertices v_{P_i} and v_{P_j} would form a $K_{3,3}$ minor in G^*. □

Claim 7.73. *There are at most* $3d$ *vertices* v *in* T *that are pairwise incomparable and that have distinct pairs* $\{P_i, P_j\}$ *in their set* $\mathsf{types}^T(v)$.

Proof. Indeed, otherwise one could construct a minor of G^*, with vertex set $\{v_{P_1}, \ldots, v_{P_d}\}$ and more than $3d$ edges, which is impossible for a planar graph. ◻

Let $L \subseteq V(T)$ (L for large) be the set of vertices v such that $|\widetilde{N}^T(v)| \geq 2$.

Claim 7.74. *If* $d \geq 2$*, we have* $|L| = \mathcal{O}(d)$.

Proof. Observe that if $v \in L$ and $u \in V(T)$ is an ancestor of v (i.e., $v \in T_u$), then $u \in L$. Also, as $d \geq 2$, $r \in L$. Thus, $T[L]$ is connected and the leaves of $T[L]$ form a set of incomparable vertices. By Claim 7.72 and Claim 7.73, we deduce that $T[L]$ has at most $6d$ leaves. This implies that the number of vertices of $T[L]$ with degree at least three (in $T[L]$) is also bounded by $6d$. The same bound of $6d$ holds on the number of connected components in the subgraph of $T[L]$ induced by the vertices having degree two in $T[L]$. In the following, let us refer to these connected components as ***long paths***.

To bound the number of vertices of degree two (in $T[L]$), let us now cut each long path into order-60 subpaths whenever possible, and let us refer to these paths as the ***short paths***. Each long path leading to at most one short path of order smaller than 60, these short paths contribute in total for at most $6d \cdot 59 = \mathcal{O}(d)$ vertices in $T[L]$. Let us now bound the number of short paths of order exactly 60. By (R8), for every such short path P we have that $\#\mathsf{types}(P) \geq 3$. This implies that the minor of G^* obtained by contracting these short paths and keeping the vertices in $\{v_{P_1}, \ldots, v_{P_d}\}$ has at most $2d$ contracted vertices (by Lemma 7.55). So, there are at most $2d$ short paths of length 60, and in total, the size of L is bounded by $6d + 6d + (6d \cdot 59) + (2d \cdot 60) = \mathcal{O}(d)$. ◻

Claim 7.75. *For every vertex* $v \in L$ *and every* $P \in \mathsf{types}^T(v)$*, the subtree* T' *of* T *rooted at* v *and containing every tree* T_u *such that* u *is a son of* v *and* $\mathsf{types}^T(u) = P$ *has constant size.*

Proof. Let $K_\alpha = (u_1, \ldots, u_q)$, for some $q \leq p$, be the clique corresponding to the border α. As all the leaves of T' have a non-empty K_α-monotone neighborhood. Rules (R5) and (R6) imply that any vertex v of T' has at most three sons in T'. Indeed, with four sons we would have either three leaves among them, or two subtrees with at least two vertices each. In the first case, Rule (R6) should have been applied. In the second case, by Rule (R2), the two non-trivial subtrees have at least two leaves each. Since the leaves are connected to u_1 (by Rule (R1)), these subtrees contain two disjoint paths connecting neighbors of u_1. In that case, Rule (R5) should have been applied.

Also, T' has depth at most four, as otherwise T' would contain two disjoint paths avoiding v and such that their endpoints are adjacent to u_1, and in such case Rule (R5) should have applied. If there was a path $P = (v, t_1, \dots, t_5)$, there would be four vertex disjoint paths from t_1, t_2, t_3, t_4 to leaves in $T' - \{v\}$. Indeed, for any $i \in \{1,2,3,4\}$, the vertex t_i has degree at least three (by Rule (R2)), this third neighbor (distinct from t_{i-1} and t_{i+1}, with $t_0 = v$) is either u_1 or it belongs to T', and we denote it x_i. The subtree of T' rooted at x_i contains a path linking x_i to a leaf, which is adjacent to u_1. Hence, in any case we have the path from t_i to a leaf.

Finally, T' having bounded degree and bounded depth, it has bounded size. $\square$

Let us define the graph G^*_L from $G^*[V(T) \cup \{v_{P_1}, \dots, v_{P_d}\}]$ by contracting every edge uv such that $|\,\mathsf{types}^T(u)| = 1$. As a minor of G^*, this graph is planar and has $\mathcal{O}(d)$ vertices by Claim 7.74 (since $V(G^*_L) = L \cup \{v_{P_1}, \dots, v_{P_d}\}$). Being planar, its number of edges is also $\mathcal{O}(d)$. Given a vertex $v \in L$ and a path $P \in \mathsf{types}^T(v)$, such that v has a son u with $\mathsf{types}^T(u) = P$, then v and v_P are adjacent in G^*_L. Hence, the number of vertices of $V(T) \setminus L$ is bounded by the number of edges in G^*_L, times the constant of Claim 7.75. This set has thus also size $\mathcal{O}(d)$. We can thus conclude the following.

Claim 7.76. *For any tree T of $G' - M' - I_{M'}$ such that $\#\mathsf{types}(T) \geq 2$, we have $|V(T)| = \mathcal{O}(\#\mathsf{types}(T))$.*

By Claim 7.59, $|I_{M'}| = \mathcal{O}(p^4 k)$. By Claim 7.70, the set $\bigcup_T V(T)$, where the union is taken over all the trees $T \in G' - M' - I_{M'}$ such that $\#\mathsf{types}(T) \leq 1$, has size $\mathcal{O}(p|M'|) = \mathcal{O}(p^3 k)$. By Claim 7.65, Claim 7.71, and Claim 7.76, the set $\cup_T V(T)$, where the union is taken over all the trees $T \in G' - M' - I_{M'}$ such that $\#\mathsf{types}(T) = 2$, has size $\mathcal{O}(p^2|M'|) = \mathcal{O}(p^4 k)$. By Claim 7.66 and Claim 7.76, the set $\bigcup_T V(T)$, where the union is taken over all the trees $T \in G' - M' - I_{M'}$ such that $\#\mathsf{types}(T) \geq 3$, has size $\mathcal{O}(p|M'|) = \mathcal{O}(p^3 k)$. Hence, the total number of vertices in G' is $\mathcal{O}(p^4 k)$. This concludes the proof of Lemma 7.50.

7.4.5 Contact segment graphs as pseudo-disk graphs

Recall that in a segment contact representation, any point belonging to two segments must be an endpoint of at least one of these segments. If a point belongs to several segments, the above property must hold for every pair of them. This defines ***contact segment graphs***. This class of graphs is a subclass of pseudo-disk graphs. This, up to the authors knowledge, does not appear in the literature. We hence prove it formally here.

Theorem 7.77. *Every contact segment graph is a pseudo-disk graph.*

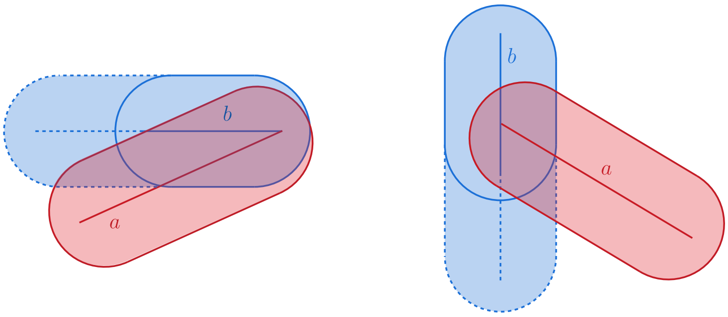

Figure 7.17: (left) Two segments a, b intersecting at their endpoints, and the corresponding pseudo-disks. According to the lengths of a and b, and the angle they form, the intersection points of $\partial\mathcal{D}_a$ and $\partial\mathcal{D}_b$ occur on their straight side or on their rounded extremity. (right) Two segments a, b intersecting at an endpoint of a, and inside b. The corresponding pseudo-disks with border $\partial\mathcal{D}_a$ and $\partial\mathcal{D}_b$ intersecting on two points.

Proof. Consider any contact system of segments $\mathcal{S}$, and let $\epsilon > 0$ be a sufficiently small value such that (1) every segment has length at least ϵ, and (2) any two points in two non-intersecting segments are at distance more than 2ϵ from each other.

The construction of the equivalent system of pseudo-disks goes in two steps. First, let us shorten every segment by $\epsilon/2$. We denote $\mathcal{S}'$ the obtained system of segments. Note that in $\mathcal{S}'$ all the segments are disjoint, but that $a', b' \in \mathcal{S}'$ are at distance at most ϵ, if and only if the corresponding original segments in $\mathcal{S}$, a, b, were intersecting. Secondly, for each segment $a' \in \mathcal{S}'$, let $\mathcal{D}_a$ be the set of points at distance at most ϵ from a point of a'. Clearly, every set $\mathcal{D}_a$ is homeomorphic to a disk, and any two such sets $\mathcal{D}_a, \mathcal{D}_b$ intersect if and only if the segments $a, b \in \mathcal{S}$ intersect. Hence, we thus just have to show that if $\mathcal{D}_a$ and $\mathcal{D}_b$ intersect, then $\partial\mathcal{D}_a$ and $\partial\mathcal{D}_b$ intersect on two points exactly. Let us consider the cases where a and b intersect at their endpoints, and the case where their intersection is an inner point for one of them. In both cases, Figure 7.17 clearly shows that there are only two intersection points. $\square$

Theorem 7.78. *There is an algorithm that, given a n-vertex contact segment graph with a representation and a parameter k, solves FVS in time $2^{\mathcal{O}\left(k^{5/6}\log k\right)} n^{\mathcal{O}(1)}$.*

Proof. The algorithm is the same as the one of Theorem 7.49, but the analysis changes in a few places. First, one should note that given an instance (G, k) the branching at the beginning (obtained from Lemma 3.4) produces a set $\mathcal{C}$ of instances (G', k', M), where G' is an induced subgraph of G, we can hence assume that G' is provided with a contact segment representation. Let us see now where the proof of Theorem 7.49 can be improved.

Note that by taking a sufficiently small ϵ in the proof of Theorem 7.77 one gets a pseudo-disk representation where every pseudo-disk contains a point of ply one. This implies that the set of inner-M'-vertices is necessarily empty, that is $|I_{M'}| = 0$.

Note that in the contact segment representation of G', there are at most $2|M|$ points where segments of M intersect, and all the segments going through such point but at most one belong to M (because M hits every triangle at least twice). Furthermore, every segment of $M' \setminus M$ intersects at least one of these points (assuming again that ϵ is sufficiently small in the construction of Theorem 7.77). Hence, we have that $|M'| = \mathcal{O}(|M|) = \mathcal{O}(pk)$.

In the proof of Lemma 7.50 it is shown that after the kernelization, the reduced instance has size $|I_{M'}| + p^2|M'|$ (last paragraph of the proof), which is $\mathcal{O}(p^3k)$ for contact segment graphs. Plugging this value in the proof of Theorem 7.49, one gets the claimed complexity. □

A finer analysis could probably lead to a time complexity of $2^{\mathcal{O}\left(k^{4/5}\log k\right)}n^{\mathcal{O}(1)}$. In any case, it still misses the lower bound of $2^{o\left(\sqrt{n}\right)}$ obtained for contact 2-DIR graphs assuming ETH.

Chapter 8
Hitting cliques

Contents

In this chapter, we present algorithms for solving the K_r-HITTING where $r \geq 2$ is a fixed integer. Observe that VERTEX COVER is the case $r = 2$ and TRIANGLE HITTING the case $r = 3$.

Organization of the chapter The first section gives a general algorithm for solving the K_r-HITTING problem for $r \geq 2$ such that for graph classes having some properties, satisfied for example by pseudo-disk graphs, this algorithm runs in subexponential FTP time. The second section deals more specifically with the K_3-HITTING problem a.k.a. TH. Again sufficient conditions are given for a graph class to admit a subexponential FPT algorithm solving this problem.

8.1 Solving the K_r-Hitting problem

Before introducing the new algorithm, we will first show that using arguments developed in the context of approximation [FLRS11], the following subexponential FPT algorithm can be obtained for apex-minor-free graphs.[1]

Theorem 8.1 (from [FLRS11])**.** *For every apex graph H and every $r \in \mathbb{N}$ there is an algorithm solving K_r-Hitting in H-minor-free graphs in time $2^{\mathcal{O}_{H,r}(\sqrt{k})} \cdot n^{\mathcal{O}_r(1)}$.*

We will need the following bound[2] on the treewidth of H-minor-free graphs:

Lemma 8.2 (Corollary 4.1. in [FLRS11])**.** *Let $\mathcal{G}$ be a graph class excluding a fixed graph H as a minor and $\eta > 0$. There exists a constant β, such that for any graph $G \in \mathcal{G}$ and $X \subseteq V(G)$ where $\mathbf{tw}(G \setminus X) \leq \eta$, we have $\mathbf{tw}(G) = \eta + \beta\sqrt{|X|}$.*

Another needed result is a connection between the treewidth of our graph and the size of a minimum dominating set. A ***dominating set*** in a graph G is a set X of vertices such that every vertex of $G - X$ has a neighbor in X. Given a graph G we will denote $\pi_{\mathrm{DOM}}(G)$ the minimum size of a dominating set.

Lemma 8.3 (Lemma 5.1. in [FLRS11])**.** *Let H be a fixed apex graph. Then there exists a constant η_H such that, for all graphs G excluding H as a minor, there exists a set X of size $\mathcal{O}_H(\pi_{DOM}(G))$ such that $\mathbf{tw}(G \setminus X) \leq \eta_H$.*

We are now ready to prove the claimed result.

Proof of Theorem 8.1. For an instance (G, k), with G a H-minor-free graph for some fixed apex graph H, we first remove all the vertices that do not appear in a clique of size r of G. This can be done in time $n^{\mathcal{O}(r)}$ and this does not change the size of an optimal solution. Denoting G' the obtained graph, observe that for S a K_r-Hitting set of G', each vertex $v \in V(G')$ is part of a clique of size r hit by S, so S is a dominating set of G'. Therefore a minimum K_r-Hitting set has size at least $\pi_{\mathrm{DOM}}(G)$. Then by greedily computing an r-approximate K_r-hitting set M in time $n^{\mathcal{O}(r)}$, and comparing its size to kr, we can either immediately conclude that we have a no-instance, or obtain that $\pi_{\mathrm{DOM}}(G) \leq rk$. In the second case, by applying Lemma 8.2 with Lemma 8.3 we get that $\mathbf{tw}(G') = \mathcal{O}_{H,r}(\sqrt{k})$. The problem can then be solved by dynamic programming on an approximate tree-decomposition in $2^{\mathcal{O}(\mathbf{tw}(G'))} n^{\mathcal{O}(1)}$ time by noting that every r-clique has to be contained in a bag. This gives an overall complexity of $2^{\mathcal{O}_{H,r}(\sqrt{k})} n^{\mathcal{O}_r(1)}$ as wanted. $\square$

[1] A graph is *apex* if the deletion of some vertex yields a planar graph.

[2] We use a corrected statement available in the full version of [FLRS11].

Our contribution

Our main result is the following subexponential parameterized algorithm for K_r-Hitting in graph classes satisfying two conditions related to cliques and treewidth. Notice that the statement of the following theorem is a simplified version of the actual Theorem 8.22 that we prove in Subsection 8.1.4.

Theorem 8.4. *Let $r \in \mathbb{N}$, $\alpha \in (0,1)$, $\mu \in \mathbb{R}_{>0}$ and let $\mathcal{G}$ be a hereditary graph class where every $G \in \mathcal{G}$ with n vertices and clique number ω has $\mathcal{O}_r(\omega^\mu n)$ cliques of order less than r and treewidth $\mathcal{O}_r(\omega^\mu n^\alpha)$. There exists $\varepsilon < 1$ and an algorithm that solves K_r-Hitting on $\mathcal{G}$ in time $2^{k^\varepsilon} \cdot n^{\mathcal{O}_r(1)}$.*

One additional motivation for this work was to generalize to K_r-Hitting the techniques used in previous work to solve Triangle Hitting (in specific graph classes) and to extract the minimal requirements for such an approach to work in more general settings. We believe we met this goal as we actually describe a single generic approach that solves K_r-Hitting on any input graph, for any r. The properties of the class in which the inputs are taken are only used to bound its running time. Such a generalization effort can be fruitful and indeed it allowed us afterwards to identify natural graph classes where subexponential algorithms exist as a consequence of our general result, as we detail now.

In Subsection 8.1.5 we derive from Theorem 8.22 the following applications.

Theorem 8.5. *There is an algorithm solving K_r-Hitting in pseudo-disk graphs in time $2^{\mathcal{O}_r\left(k^{(r+1)/(r+2)} \log k\right)} \cdot n^{\mathcal{O}_r(1)}$.*

Disk graphs and contact segment graphs are pseudo-disk graphs, so Theorem 8.5 applies to the two settings handled by the algorithms of [ACO23] for disk graphs and Chapter 6 for contact segment graphs. Another application is the following:

Theorem 8.6. *There is an algorithm solving K_r-Hitting in map graphs in time $2^{\mathcal{O}_r\left(k^{(r+1)/(r+2)} \log k\right)} \cdot n^{\mathcal{O}_r(1)}$.*

We cannot expect a similar consequence for the more general class of string graphs. Indeed, there are n-vertex string graphs that are triangle-free and have treewidth $\Omega(n)$, for instance, the balanced bicliques.[3] Note that such graphs prevent string graphs from satisfying the requirement of Theorem 8.4. Also, and more importantly, by Theorem 5.10 under ETH there is no $2^{o(n)}$-time algorithm for K_3-Hitting in 2-DIR graphs, a restricted subclass of string graphs. As we will show, large bicliques are the only obstructions in the sense that forbidding them in

[3] $K_{n,n}$ can be drawn as a 2-DIR graph with n horizontal disjoint segments that are all crossed by n vertical disjoint segments.

string graphs allows us to solve the problem in subexponential FPT time. For this, we use the following light version of Theorem 8.4 (which is also a consequence of Theorem 8.22) suited for classes where the clique number is already bounded.

Theorem 8.7. *Let $r \in \mathbb{N}$, $\alpha \in (0,1)$ and let $\mathcal{G}$ be a hereditary graph class where every $G \in \mathcal{G}$ with n vertices has $\mathcal{O}(n)$ cliques of order less than r and treewidth $\mathcal{O}(n^\alpha)$. There exists an algorithm that solves K_r-Hitting on $\mathcal{G}$ in time $2^{\mathcal{O}_r(k^{2/(1+1/\alpha)} \log k)} \cdot n^{\mathcal{O}_r(1)}$.*

As a consequence, we obtain a subexponential FPT algorithm for string graphs excluding large bicliques.

Theorem 8.8. *There is an algorithm solving K_r-Hitting in $K_{t,t}$-subgraph-free string graphs in time $2^{\mathcal{O}_{t,r}(k^{2/3} \log k)} \cdot n^{\mathcal{O}_r(1)}$.*

We note that under ETH the contribution of k cannot be improved to $2^{o(\sqrt{k})}$, according to Theorem 5.10. Finally, we observe that Theorem 8.4 can also be applied to certain classes of sparse graphs.

Theorem 8.9. *For every graph H, there is an algorithm solving K_r-Hitting in H-minor-free graphs in time $2^{\mathcal{O}_{H,r}(k^{2/3} \log k)} \cdot n^{\mathcal{O}_r(1)}$.*

This is a more general statement than Theorem 8.1 in the sense that we are not limited to apex-minor-free graphs, with the price of a slightly larger time complexity.

Our techniques

Our subexponential algorithm for K_r-Hitting of Theorem 8.4 is obtained as follows. Given (G, k), we first perform in Subsection 8.1.2 a preliminary branching step whose objective is to get rid of large cliques (i.e., cliques of order at least k^ε for some $\varepsilon \in (0,1)$ that we will fix later). This step is a folklore technique that is frequently used for any problem where a solution has to contain almost all vertices of a large clique, like Triangle Hitting (or in general K_r-Hitting), Feedback Vertex Set, or Odd Cycle Transversal. After this preprocessing has been performed we can assume that the instances to solve have no clique on more than k^ε vertices. Then, we greedily compute an r-approximate K_r-hitting set M. If $|M| > kr$ we can already answer negatively, so in the following, we may assume $|M| \leq kr$ and will use the fact that there is no K_r in $G - M$.

Now, the crucial part of the algorithm is Subsection 8.1.3. Informally, the goal of the algorithms described in this section is to extend M into a superset M' together with a new parameter $k' \leq k$, such that $|M'| = O\left(k^{1+c\varepsilon}\right)$ (for some $c > 0$) and that vertices of $V(G) \setminus M'$ are irrelevant for the problem of hitting r-cliques.

In that way, we can remove them, and it remains only to solve $(G[M'], k')$, whose treewidth can be typically bounded by $\sqrt{\omega(G)|M'|}$ in the graph classes we consider. As $\sqrt{\omega(G)|M'|} = O\left(k^{1/2+\frac{c+1}{2}\varepsilon}\right)$, this leads to a subexponential algorithm. We stress that the high-level description above is not a kernelization because first we actually do not produce a single reduced instance but instead, we have to branch and obtain a subexponential (in k) number of sub-instances and second because the reduction steps are not carried out in polynomial time, but in subexponential (in k) time.

To obtain this set M', we use lemmatas 8.14 and 8.21 that are inspired from the following "virtual branching" procedure of [LPS⁺22, Lemma 6.5]. This routine was introduced for Triangle Hitting and works as follows. It starts with a triangle hitting set M (obtained by greedily packing disjoint triangles) and outputs a slightly larger superset M' such that vertices in $G - M'$ are *almost* useless, in the sense that every triangle has at least two vertices in M' (we do not detail here how to handle the triangles with exactly two vertices in M' and refer to [LPS⁺22]). This is done as follows. For a vertex $v \in M$, consider a maximum matching $M(v) \subseteq N(v) \cap (V(G) \setminus M)$. If for every $v \in M$ such a matching is small, meaning $|M(v)| \leq k^\varepsilon$, then we can define $M' = M \cup \bigcup_{v \in M} M(v)$. We are done as $|M'| = \mathcal{O}(k^{1+\varepsilon})$ remains small, and there is no longer a $v \in M'$ with an edge in $N(v) \setminus M'$ (as this would form a triangle outside M). Otherwise, if for some $v \in M$, $|M(v)| > k^\varepsilon$, a solution of Triangle Hitting should either take v, or otherwise hits all edges of $M(v)$. In the second case, it would be too costly to guess which vertex is taken in each $e \in M(v)$, so instead the procedure "absorbs" $M(v)$ by defining $M' = M \cup M(v)$. This absorption increases the size of M, but "virtually" decreases the parameter k, as it increases by $|M(v)|$ the size of a matching the solution will have to hit. This leads to a running time typically dominated by the recurrence $f(k) = f(k-1) + f(k - k^\varepsilon)$, which is subexponential in k.

Now, coming back to K_r-Hitting, given a set M, let us say[4] that a ***type-i clique*** is an r-clique X such that $|X \cap M| = i$. We could remove type-1 cliques by using the previous virtual branching procedure, defining now $M(v)$ as a packing of $r-1$ cliques instead of a packing of edges. However, the problem is then that, if we want to obtain a set M' as promised (where vertices of $V(G) \setminus M'$ are useless), we also have to remove type-i cliques for $i \in \{2, \ldots, r-1\}$. However, there is a first obstacle to removing such type-i cliques: as the part common with M (which was before a single vertex in M) is now an i-clique, we cannot afford to enumerate all possible choices X' of such an i-clique in M. Indeed, already for $i = 2$, we would possibly consider a quadratic number of sets X', so absorbing every packing (of $(r-i)$-cliques) $M(X')$ (in the unfortunate case where these are all small) would

[4]This notion of type-i clique will not be used later and is just introduced for this sketch.

result in a set $M' = M \cup \bigcup_{X' \subseteq M, |X'|=2} M(X')$ with $|M'| = \Omega(k^2)$, which is too large for our purpose. To circumvent this issue, we identified a key property that holds in many geometric graph classes like pseudo-disk or $K_{t,t}$-subgraph-free string graphs: in such graphs, there is only a linear (in the number of vertices) number of i-cliques for fixed i, and for fixed clique number ω. In our case, as ω is small, and $i \leq r$ is fixed, this implies that there are $\mathcal{O}(|M|)$ such i-cliques in M. Hence, it allows us to control the size of M'. Of course, dealing with r-cliques instead of triangles also raises other problems, in particular related to the way we hit their intersection with M which is no longer a single vertex but a clique. To deal with this issue we had to introduce an annotated variant of the problem where additional sets of vertices have to be hit besides r-cliques.

Observe that the result presented in Subsection 3.2.1 generalizes our work both by the problems considered and the graph classes where it can be applied. However, compared to this new result our algorithm has two advantages: it is conceptually much simpler, and the running time is explicit.

Organization of the chapter

In Subsection 8.1.1 we give the necessary definitions. We describe the first branching in Subsection 8.1.2 and the second in Subsection 8.1.3. The algorithm is given in Subsection 8.1.4. We give applications to selected graph classes in Subsection 8.1.5.

8.1.1 Preliminaries

To simplify the presentation we will assume that r is a fixed constant instead of explicitly giving it as a parameter in all our algorithms and lemmas.

For any $i \in \mathbb{N}$, an ***i*-clique** is a clique on i vertices and a $(< i)$-clique is a clique on less than i vertices.

Hypergraphs and hitting sets

A ***hypergraph*** is simply a collection of sets, where any set is referred to as an ***hyperedge***. Therefore, $|\mathcal{D}|$ refers to the number of sets in the hypergraph $\mathcal{D}$ and we define $V(\mathcal{D}) = \bigcup_{D \in \mathcal{D}} D$. By $\mathcal{K}_r(G)$ (resp. $\mathcal{K}_{<r}(G)$) we denote the hypergraph of r-cliques (resp. $(< r)$-cliques) of G, i.e. $\mathcal{K}_r(G) = \{X \subseteq V(G) : X \text{ is an } r\text{-clique}\}$.

A ***hitting set*** of $\mathcal{D}$ is a subset $X \subseteq V(\mathcal{D})$ that intersects every hyperedge of $\mathcal{D}$. A ***matching*** of $\mathcal{D}$ is a collection of disjoint hyperedges. The maximum size of a matching in $\mathcal{D}$ is denoted by $\nu(\mathcal{D})$. Note that a hitting set of $\mathcal{D}$ has always size at least $\nu(\mathcal{D})$ as it needs to intersect each of the elements of a maximum matching, which are disjoint.

8.1.2 Dealing with large cliques

A K_r-hitting set can be useful to detect large cliques, as we explain now. This will allow us to identify cliques on which to branch.

Lemma 8.10. *Given a graph G, a K_r-hitting set M, and an integer $p > r$, one can find a p-clique of G, or correctly conclude none exists, in $\mathcal{O}(p^2|M|^p n^{r-1})$ steps.*

Proof. For each choice of $p - r + 1$ vertices of M and $r - 1$ other vertices of G, we check whether they form a clique (which takes $\mathcal{O}(p^2)$ time), in which case return it and stop. If no clique is found we return that G is K_p-free. The correctness follows from the following observation: as $G - M$ is K_r-free, every p-clique of G has at least $p - r + 1$ of its vertices in M. □

As noted in the proof of Lemma 8.10, every large clique of the input graph will have most of its vertices in a K_r-hitting set so we can branch on which these are (see Subsection 3.1.1 for more details).

Lemma 8.11. *There is an algorithm that, given two integers $r \geq 1$ and $p > r$, an instance (G, k) of K_r-Hitting and a K_r-hitting set M of G, runs in $2^{kr(\log p)/p}|M|^p n^{\mathcal{O}(r)}$ steps and returns a collection $\mathcal{Y}$ of $2^{kr(\log p)/p} \cdot |M|^p$ instances of the same problem such that:*

1. *(G, k) is a yes-instance if and only if $\mathcal{Y}$ contains a yes-instance;*
2. *for every $(G', k') \in \mathcal{Y}$, G' has no p-clique.*

8.1.3 Picking petals

Let $i \in \{1, \dots, r - 1\}$ and let X be an i-clique in a graph G. An ***r*-petal** of X is a subset of vertices of $G - X$ that together with X forms an r-clique. We denote by $\mathsf{Petals}_r(G, X)$ the hypergraph of r-petals of X, i.e., $\mathsf{Petals}_r(G, X) = \{Y \subseteq V(G) \setminus X : X \cup Y \in \mathcal{K}_r(G)\}$. See Figure 8.1 for an illustration.

To deal more easily with the recursive steps in our algorithms we introduce Ann.-K_r-Hitting, an annotated version of K_r-Hitting where several choices have already been made, which is recorded by extra vertex subsets that the solution is required to hit.

In the Ann.-K_r-Hitting problem, one is given a triple $(G, \mathcal{D}, k)$ where G is a graph, $\mathcal{D} \subseteq \mathcal{K}_{<r}(G)$ and k is an integer. A ***solution*** to this instance is a set of vertices that hits $\mathcal{K}_r(G)$ and $\mathcal{D}$ and has at most k vertices. The question is whether the input instance admits a solution.

In the forthcoming algorithms, it will also be more convenient to consider, along with an instance, a non-optimal solution M. This motivates the following

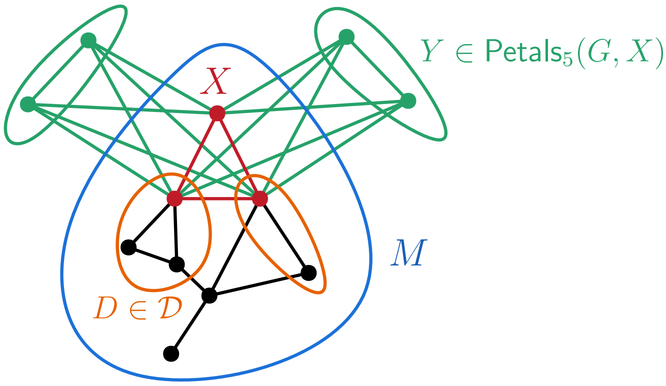

Figure 8.1: A 3-clique X with two 5-petals that live in $G - M$. Here $\mathcal{D}$ contains two hyperedges, represented in orange. Observe that no hyperedge of $\mathcal{D}$ is contained in X, so X is a lush 3-clique.

definition. A ***context*** is a pair $((G, \mathcal{D}, k), M)$ where $(G, \mathcal{D}, k)$ is an instance of Ann.-K_r-Hitting, M is a K_r-hitting set (possibly larger than k), and $V(\mathcal{D}) \subseteq M$. We say that a context is ***positive*** if the instance of Ann.-K_r-Hitting it contains is a yes-instance, and ***negative*** otherwise.

Given a context $((G, \mathcal{D}, k), M)$ and $i \in \{1, \ldots, r-1\}$, a ***lush i-clique*** is an i-clique X of $G[M]$ that has an r-petal in $G - M$ and such that no $D \in \mathcal{D}$ is subset of $V(X)$. See Figure 8.1 for an example.

Informally, if X is a lush clique then we are not guaranteed that hitting $\mathcal{D}$ alone will always also hit the cliques induced by X and its petals, so we have to take care of them separately. Ideally, we would like to get rid of lush cliques so that we can focus on $\mathcal{D}$ to solve the problem. We say that the context $((G, \mathcal{D}, k), M)$ is ***i-stripped*** if for every $i' < i$ it has no lush i'-clique. These notions are motivated by the following easy lemma.

Lemma 8.12. *Let $((G, \mathcal{D}, k), M)$ be an r-stripped context. The instances $(G, \mathcal{D}, k)$ and $(G[M], \mathcal{D}, k)$ of Ann.-K_r-Hitting are equivalent.*

Proof. First, recall that as $((G, \mathcal{D}, k), M)$ is a context, $V(\mathcal{D}) \subseteq M$ so $(G[M], \mathcal{D}, k)$ is indeed a valid instance of Ann.-K_r-Hitting. Also as $\mathcal{K}_r(G[M]) \subseteq \mathcal{K}_r(G)$, if $(G, \mathcal{D}, k)$ has a solution then $(G[M], \mathcal{D}, k)$ does. Therefore, we only need to show the other direction. Let $v \in V(G) \setminus M$ and suppose that G contains some r-clique X with $v \in X$. The set $M \cap X$ is not empty because M is a K_r-hitting set. Note that $X \cap M$ has a petal $X \setminus M$ disjoint from M. It cannot form a lush i-clique (for $i = |M \cap X| < r$) as this would contradict the assumption that the considered context is r-stripped, so there is some $D \in \mathcal{D}$ that is a subset of $X \cap M$. Therefore every solution of $(G[M], \mathcal{D}, k)$ does hit X in G, as it hits D. As this holds for every v and X as above, every solution of $(G[M], \mathcal{D}, k)$ is a solution of $(G, \mathcal{D}, k)$, as desired. $\square$

By the above lemma, if we manage to get rid of lush cliques, we can obtain an equivalent instance whose graph is not larger than $G[M]$. As we will see, we are able to handle lush cliques with the price of slightly increasing the size of M and producing several instances to represent the solutions of the original instance.

Lemma 8.13. *There is an algorithm that, given an integer $i < r$ and an i-stripped context $((G, \mathcal{D}, k), M)$, runs in time $n^{\mathcal{O}(r)}$ and either correctly concludes that the context is $(i+1)$-stripped, or returns a lush i-clique X.*

Proof. The input context is i-stripped so we only have to check whether it contains a lush i-clique. We iterate over the i-cliques of $G[M]$. For every such clique X, we first check if $D \subseteq V(X)$ for some $D \in \mathcal{D}$. If so X is not a lush i-clique so we can move to the next choice of X. Otherwise, we check if the common neighborhood of the vertices of X in $G - M$ has an $(r-i)$-clique. If so this is an r-petal so X is lush and we can return it. Otherwise, we continue to the next choice of X. If the iteration terminates without detecting a lush i-clique, we can safely return that the input context is $(i+1)$-stripped. The complexity bound follows from the fact that $|\mathcal{D}| = n^{\mathcal{O}(r)}$ and that $G[M]$ has $n^{\mathcal{O}(r)}$ i-cliques. $\square$

The following lemma is the key branching step in our subexponential algorithms for K_r-Hitting.

Lemma 8.14. *There is an algorithm that, given $i \in \{1, \dots, r-1\}$ and $\lambda \in \{1, \dots, k\}$ and an i-stripped context $((G, \mathcal{D}, k), M)$ where we denote by ζ the number of i-cliques in $G[M]$, runs in time $2^{\mathcal{O}((k/\lambda)\cdot\log\zeta)} \cdot n^{\mathcal{O}(r)}$ and returns a collection $\mathcal{Z}$ of size $2^{\mathcal{O}((k/\lambda)\cdot\log\zeta)}$ of $(i+1)$-stripped contexts such that:*

(I) $((G, \mathcal{D}, k), M)$ is a positive context if and only $\mathcal{Z}$ contains one; and

(II) for every $((G', \mathcal{D}', k'), M') \in \mathcal{Z}$, $M \subseteq M'$, $|M'| \leq |M| + r(\lambda\zeta + k)$, and $k' \leq k$.

Proof. Let us define an auxiliary algorithm that takes as input $(i, \lambda, (G, \mathcal{D}, k), M, \mathcal{P}^*)$, where $(i, \lambda, (G, \mathcal{D}, k), M)$ is as specified in the statement of the lemma, and $\mathcal{P}^*$ is a matching of $\mathcal{D}$, and output the promised collection $\mathcal{Z}$. Such an auxiliary algorithm will imply the lemma, as we will run it with parameters $(i, \lambda, (G, \mathcal{D}, k), M, \emptyset)$. For simplicity, as i and λ will not change between recursive calls: we denote by $((G, \mathcal{D}, k), M, \mathcal{P}^*)$ the parameter of this auxiliary algorithm, which is defined as follows:

1. If $k < |\mathcal{P}^*|$, we can immediately return $\mathcal{Z} = \emptyset$.

2. Run the algorithm of Lemma 8.13 on i and $((G, \mathcal{D}, k), M)$. If no lush i-clique is found then the input context is already $(i+1)$-stripped so we return $\mathcal{Z} = \{((G, \mathcal{D}, k), M)\}$. Otherwise, let X denote the lush i-clique we found.

3. Construct the hypergraph $\mathcal{B}_X$ of those r-petals of X that are subset of $V(G) \setminus M$.

4. Greedily compute a maximal matching $\mathcal{P}$ in $\mathcal{B}_X$.

5. If $\tilde{v}_X \leq \lambda$, we return the result of the recursive call with parameters $((G, \mathcal{D}, k), M \cup V(\mathcal{P}), \mathcal{P}^*)$ (and quit).

6. Otherwise, we investigate the different ways to hit the r-cliques induced by X and $\mathcal{B}_X$ (via X or via $\mathcal{B}_X$) as follows.

   (a) Solutions containing some (yet unspecified) vertex of X: this is done by a recursive call with parameters $((G, \mathcal{D} \cup \{X\}, k), M, \mathcal{P}^*)$. We call $\mathcal{Z}_1$ the resulting family.

   (b) Solutions hitting $\mathcal{P}$.[5] As $\mathcal{P}$ is disjoint from M (hence from $\mathcal{P}^*$), such solutions exist only if $k \geq |\mathcal{P}^*| + |\mathcal{P}|$. In this case, we define $\mathcal{Z}_2$ as the result of the recursive call with parameters

   $$((G, \mathcal{D} \cup \mathcal{P}, k), M \cup V(\mathcal{P}), \mathcal{P}^* \cup \mathcal{P}).$$

   Otherwise, we set $\mathcal{Z}_2 = \emptyset$.

7. We return $\mathcal{Z}_1 \cup \mathcal{Z}_2$.

Observe first that in step (6b), the last parameter $\mathcal{P}^* \cup \mathcal{P}$ is a matching as required, as in particular $V(\mathcal{P}) \subseteq V(G) - M$ and $V(\mathcal{P}^*) \subseteq M$, by definition. We first prove the following fragment of item ((II)).

Claim 8.15. *For every $((G', \mathcal{D}', k'), M') \in \mathcal{Z}$, $M \subseteq M'$ and $k' \leq k$.*

Proof. The only places where M is updated are steps 5 and 6b, where new vertices are added to it. Besides we never change the value of the parameter k. □

Let us describe the recursion tree T of the above algorithm on some input $((G, \mathcal{D}, k), M, \mathcal{P})$. The nodes of this tree are inputs. The root is $((G, \mathcal{D}, k), M, \emptyset)$ and a node s' is a child of a node s if a call to the above algorithm on the input s triggers a call on the input s'. Therefore, the leaves of this tree are the inputs that do not trigger any recursive call.

[5]Notice that some of these solutions have possibly already been investigated in the previous step. For our purpose, it is not an issue however to consider several times the same solution.

Let us consider a path from the root of T to some leaf. We denote by

$$((G_j, \mathcal{D}_j, k_j), M_j, \mathcal{P}_j^*)_{j\in\{1,\dots,\ell\}}$$

the inputs along this path, and by $C_j = ((G_j, \mathcal{D}_j, k_j), M_j)$ the corresponding contexts, with $((G_1, \mathcal{D}_1, k_1), M_1, \mathcal{P}_1^*) = ((G, \mathcal{D}, k), M, \emptyset)$ and $((G_\ell, \mathcal{D}_\ell, k_\ell), M_\ell, \mathcal{P}_\ell^*)$ corresponding to the aforementioned leaf. Also, for every $j \in \{1, \dots, \ell - 1\}$ we denote by X_j the lush i-clique of C_j that is considered in the corresponding call.

We first show that all lush i-cliques considered along this path belong to the hitting set $M = M_1$ of the initial context.

Claim 8.16. *For every $j \in \{1, \dots, \ell - 1\}$, $X_j \subseteq M$.*

Proof. Suppose towards a contradiction that for some j, $X_j \not\subseteq M$. Observe that since C_j is not a leaf, $\mathcal{B}_{X_j}$ is not empty so X_j induces an r-clique together with some r-petal $B \in \mathcal{B}_{X_j}$. Recall that B is disjoint from M_j, by the definition of $\mathcal{B}_{X_j}$. As M_j is a superset of M (Claim 8.15), B is disjoint from M as well. Therefore, M intersects X_j as otherwise the r-clique $X_j \cup B$ would not be hit by M. Let $i' = |X_j \cap M|$. Note that $X_j \cap M$ is a lush i'-clique of C_1 since it has an r-petal $B \cup (X \setminus M)$ disjoint from M (the fact that no set of $\mathcal{D}$ is subset of $X_j \cap M$ follows from the fact that this property holds for X_j). By our initial assumption and as $|X_j| = i$, we have $i' < i$. This contradicts the fact that C_1 is i-stripped. □

With a similar proof, we can show the following (so the recursive calls are indeed made on valid inputs).

Claim 8.17. *For every $i \in \{1, \dots, \ell\}$, C_i is i-stripped.*

Let us now show that each lush clique is only considered once.

Claim 8.18. *For every distinct $j, j' \in \{1, \dots, \ell - 1\}$, $X_j \neq X_{j'}$.*

Proof. Suppose towards a contradiction that for some $j < j'$ we have $X_j = X_{j'}$. Then in the call on the context C_j, the next context C_{j+1} was obtained at step 5 or 6b (since at step 6a we would include X_j in $\mathcal{D}_j$, preventing it to be considered in future calls). In any of these two possibilities, we set $M_{j+1} = M_j \cup V(\mathcal{P}_j)$, where $\mathcal{P}_j$ denotes the maximal matching of step 4 in the call on context C_j.

Besides, as X_j is a lush i-clique in the context $C_{j'}$, then it has some r-petal B subset of $V(G_{j'}) - M_{j'}$. As $M_{j+1} \supseteq M_{j'}$ (Claim 8.15), B is also an r-petal of X_j and is subset of $V(G_j) - M_j$ and by the above observation, it is disjoint from $\mathcal{P}_j$. This contradicts the maximality of $\mathcal{P}_j$. □

Recall that the number of i-cliques in M is ζ. As a consequence of Claim 8.16 and Claim 8.18, we get the following.

Claim 8.19. *The recursion tree has depth at most ζ.*

We can now conclude the proof of item (II).

Claim 8.20. *For every* $((G', \mathcal{D}', k'), M') \in \mathcal{Z}$, $|M'| \leq |M| + r(\lambda\zeta + k)$.

Proof. When considering the context C_j, and given the chosen maximal matching $\mathcal{P}_j$ of the petals of X_j, the set M_{j+1} is defined from M_j by:

- either adding the at most $\lambda(r-1)$ new vertices of $\mathcal{P}_j$, if we make the recursive call at step 5,
- or by adding the vertices of the petals of $\mathcal{P}_j$, if we recurse at step 6b. Recall that in this case we also have $\mathcal{D}_{j+1} = \mathcal{D}_j \cup \mathcal{P}_j$ and $\mathcal{P}^*_{j+1} = \mathcal{P}^*_j \cup \mathcal{P}_j$

In the latter case, the number of added vertices is not directly bounded however we have $|\mathcal{P}^*_{j+1}| = |\mathcal{P}^*_j| + |\mathcal{P}_j|$. Because of the stopping condition of step 1, we will overall (from C_1 to C_ℓ) add at most k petals to the hypergraph and each petal has at most $r-1$ vertices. Hence we get $|M'| \leq |M| + r(\lambda\zeta + k)$, as claimed. □

Let us now show that the algorithm is correct, i.e., item (I) of the statement of the lemma. The proof is by induction on the depth of the recursion tree (i.e., $\ell - 1$ with the notation above). When the depth is 0, there is no recursive call. This corresponds to the two base cases in this algorithm: step 1, when the "budget" k is insufficient, and step 2, when the input context is already $(i+1)$-stripped. Clearly, the outputs in these cases satisfy item (I).

We now consider the case of a run of the algorithm where the depth of the recursion tree is at least 1 and suppose that item (I) holds for all runs with recursion trees of smaller depth.

- If the recursive call is made at step 5 then item (I) trivially holds because the instance is unchanged.
- Otherwise, note that any solution has to hit the r-cliques induced by X and $\mathcal{B}_X$. Thus, any solution either contains a vertex of X, or hits $\mathcal{B}_X$ (or both). These are exactly the two branches that are explored in steps 6a and 6b, respectively, by our induction hypothesis.

The above shows that the algorithm is correct. It remains to prove that it has the claimed running time. Note that we do not update the graph nor the parameter between recursive calls: we will always work on the graph G with n vertices and with the parameter k. Therefore, the induction proving the time bound will use two different parameters as we explain now. For every $x, y \in \mathbb{N}$, let us denote by $T(x, y)$ the worst-case running time of the above algorithm on an input $((G, \mathcal{D}, k), M, \mathcal{P}^*)$ with $|G| = n$ such that there is at most x (non-necessarily disjoint) lush i-cliques in $G[M]$ and such that $|\mathcal{P}^*| \geq y$. Let $S_r(n)$ be the sum of the worst-case complexity of all subroutines needed in the different steps of item 1 to item 7 to the exception of recursive calls (one such subroutine is the algorithm of Lemma 8.13, another one is the construction of the hypergraph $\mathcal{B}_X$). Observe that $T(x, y) \leq S_r(n)$ when $y > k$ (as we fall into the base case of step 1) or when $x = 0$ (as we fall into the base case of step 2). Notice also that by definition we have $T(x, y) \leq T(x', y)$ for any $x \leq x'$, and $T(x, y) \leq T(x, y')$ for any $y' \leq y$.

If we make a recursive call at step 5, we return in time at most $T(x-1, y)$ (by induction) as, by Claim 8.18, no lush i-clique of the original instance is considered twice. Otherwise, we will make recursive calls in step 6 which by induction take time at most $T(x-1, y) + T(x-1, y+|\mathcal{P}|)$ as, in the first branch 6a of recursion, Claim 8.18 implies again that no lush i-clique is considered two times, and in the second branch 6b, we know in addition that the size of the matching given as parameter increases by $|\mathcal{P}|$. Thus, in both cases (and including the other computation steps which take time $S_r(n)$), we obtain the upper bound:

$$\begin{aligned} T(x, y) &\leq T(x-1, y) + T(x-1, y+|\mathcal{P}|) + S_r(n) \\ &\leq T(x-1, y) + T(x, y+\lambda) + S_r(n). \end{aligned}$$

(For the last line recall that T is anti-monotone with respect to its second parameter.)

Let us now show that for every $x, y \in \mathbb{N}$, $T(x, y) \leq xT(x, y+\lambda) + (x+1)S_r(n)$. The proof is by induction on x. The base case $x = 0$ holds as observed above. Suppose the inequality holds for $x - 1$. As proved above

$$\begin{aligned} T(x, y) &\leq T(x-1, y) + T(x, y+\lambda) + S_r(n) \\ &\leq (x-1)T(x-1, y+\lambda) + xS_r(n) && \text{(by induction)} \\ &\quad + T(x, y+\lambda) + S_r(n) \\ &\leq xT(x, y+\lambda) + (x+1)S_r(n), && \text{as claimed.} \end{aligned} \tag{8.1}$$

We now prove that for every $x, y \in \mathbb{N}$,

$$T(x, y) \leq x^{\frac{k+1-y}{\lambda}} + \left(1 + \frac{k+1-y}{\lambda}\right)(x+1)S_r(n).$$

This time the induction is on y. The base case $y > k$ holds as observed above. Let $x \geq 1$ and $y \leq k$ and suppose the inequality holds for any pair (x', y') with $y' > y$. Then as proved above in Eq. 8.1,

$$\begin{aligned} T(x, y) &\leq xT(x, y+\lambda) + (x+1)S_r(n) \\ &\leq x \cdot x^{\frac{k+1-y-\lambda}{\lambda}} + \left(1 + \frac{k+1-y-\lambda}{\lambda}\right)(x+1)S_r(n) + (x+1)S_r(n) \\ &\leq x^{\frac{k+1-y}{\lambda}} + \left(1 + \frac{k+1-y}{\lambda}\right)(x+1)S_r(n). \end{aligned}$$

Observe that $S_r(n)$ is dominated by the time spent in the algorithm of Lemma 8.13, and thus $S_r(n) = n^{\mathcal{O}(r)}$. As we initially have $x \le \zeta$ and $y \ge 0$, we obtained the claimed running time. A similar analysis can be used to bound the size of the output family $\mathcal{Z}$. □

By iterating the algorithm of Lemma 8.14 for increasing values of i we can obtain a collection of r-stripped contexts, as we explain now.

Lemma 8.21 (picking petals)**.** *There is an algorithm that, given $i \in \{1, \ldots, r\}$, $\lambda \in \mathbb{R}_{\ge 1}$, a context $((G, \mathcal{D}, k), M)$, where we denote by ζ the number of ($< r$)-cliques in $G[M]$, runs in time $2^{\mathcal{O}((i \cdot k/\lambda) \cdot \log \zeta)} \cdot n^{\mathcal{O}(r)}$ and returns a collection $\mathcal{Z}$ of size $2^{\mathcal{O}((i \cdot k/\lambda) \cdot \log \zeta)}$ of i-stripped contexts such that:*

1. *$(G, \mathcal{D}, k)$ is a yes-instance if and only if some input of $\mathcal{Z}$ contains one; and*
2. *for every $((G', \mathcal{D}', k'), M') \in \mathcal{Z}$, $M \subseteq M'$, $|M'| \le |M| + ir(\lambda\zeta + k)$, and $k' \le k$.*

Proof. Again for the sake of clarity, we assume λ is a fixed constant.

The proof is by induction on i. For the base case $i = 1$ we simply observe that $((G, \mathcal{D}, k), M)$ is already 1-stripped so there is nothing to do.

Let us now suppose that $i > 1$ and that the statement holds for $i - 1$. Therefore, from the input context $((G, \mathcal{D}, k), M)$ and $i - 1$ we can use the induction hypothesis to construct a collection $\mathcal{Z}_{i-1}$ of $(i - 1)$-stripped contexts satisfying the statement for $i - 1$. Now we apply the algorithm of Lemma 8.14 to $i - 1$ and each context in $\mathcal{Z}_{i-1}$ and call $\mathcal{Z}_i$ the union of the obtained collections. Item 1 follows from the properties of $\mathcal{Z}_{i-1}$ and the correctness of the algorithm of Lemma 8.14. Constructing $\mathcal{Z}_{i-1}$ takes time $2^{\mathcal{O}(((i-1) \cdot k/\lambda) \cdot \log \zeta)} \cdot n^{\mathcal{O}(r)}$ (by induction) and then we run the $\left(2^{\mathcal{O}((k/\lambda) \cdot \log \zeta)} \cdot n^{\mathcal{O}(r)}\right)$-time algorithm of Lemma 8.14 on each of its $2^{\mathcal{O}(((i-1) \cdot k/\lambda) \cdot \log \zeta)}$ contexts. This results in an overall running time of $2^{\mathcal{O}((i \cdot k/\lambda) \cdot \log \zeta)} \cdot n^{\mathcal{O}(r)}$, as claimed. In each context of $\mathcal{Z}_{i-1}$ the set M' has size at most $|M| + (i - 1)r(\lambda\zeta + k)$ (by induction) and after the run of the algorithm Lemma 8.14, the corresponding set in the produced instances has at most $r(\lambda\zeta + k)$ vertices more so we get the desired bound. Finally, as in the proof of Lemma 8.14 the value of k never changes. □

8.1.4 Kick the cliques

For every $\phi, \gamma \in \mathbb{R}_{\ge 0}$ and $\alpha \in (0, 1)$, we say that graph class $\mathcal{G}$ has property $P_r(\phi, \gamma, \alpha)$ if there are functions $f(x) = \mathcal{O}_r(x^{\phi})$ and $g(x) \in \mathcal{O}_r(x^{\gamma})$ such that for $G \in \mathcal{G}$ with n vertices and clique number less than ω,

(P1) G has at most $f(\omega) \cdot n$ cliques of order less than r; and

(P2) $\mathbf{tw}(G) \le g(\omega) n^{\alpha}$.

Our main contribution is the following.

Theorem 8.22. *For every hereditary graph class $\mathcal{G}$ that has property $P_r(\phi, \gamma, \alpha)$ for some $\alpha \in (0,1)$, $\phi, \gamma \in \mathbb{R}_{\ge 0}$, there is an algorithm that solves K_r-*Hitting *on $\mathcal{G}$ in time*

$$2^{\mathcal{O}_{r,\phi}(k^{\varepsilon}\log k)} \cdot n^{\mathcal{O}_r(1)} \qquad \text{with} \quad \varepsilon = \frac{\gamma + \alpha(\phi+2)}{\gamma + \alpha(\phi+1)+1} < 1,$$

i.e., subexponential FPT time.

Proof. Let $f\colon x \mapsto c_f \cdot x^{\phi}$ and $g\colon x \mapsto c_g \cdot x^{\gamma}$ (for some $c_f, c_g > 0$) be as in the definition of property $P_r(\phi, \gamma, \alpha)$. For the sake of readability, we will here use (when deemed useful) exp to denote the function $x \mapsto 2^x$ defined over reals. Given an instance (G, k) of K_r-Hitting, we consider the context $((G, \mathcal{D}, k), M)$ where $\mathcal{D} = \emptyset$ and M is an r-approximation of a K_r-hitting set computed in $n^{\mathcal{O}(r)}$ time by greedily packing disjoint r-cliques. If $|M| > kr$ we can already answer negatively, so in what follows we suppose that $|M| \le kr$. We run the algorithm of Lemma 8.11 with $p = \lceil k^{\varepsilon} \rceil$ for some constant $\varepsilon \in (0,1)$ that we will fix later. In time

$$2^{\mathcal{O}(rk^{1-\varepsilon}\log k)} n^{\mathcal{O}(r)}$$

we obtain a set $\mathcal{Y}$ of $2^{\mathcal{O}(rk^{1-\varepsilon}\log k)}$ contexts that have no p-clique. For each such context, we apply the petal-picking algorithm of Lemma 8.21 with

$$\begin{aligned} \lambda &= k^{\varepsilon} \qquad \text{and} \\ \zeta &= f(p) \cdot |M| \\ &= \mathcal{O}_r\left(k^{1+\varepsilon\phi}\right). \end{aligned}$$

Let $\mathcal{Z}$ denote the union of the output families of these algorithms. Computing this set then takes time

$$\begin{aligned} |\mathcal{Y}| \cdot 2^{\mathcal{O}((r\cdot k/\lambda)\cdot\log\zeta)} \cdot n^{\mathcal{O}(r)} &= 2^{\mathcal{O}(r\cdot k^{1-\varepsilon}(\log k + \log\zeta))} \cdot n^{\mathcal{O}(r)} \\ &= 2^{\mathcal{O}_r((1+\varepsilon\phi)k^{1-\varepsilon}\log k)} \cdot n^{\mathcal{O}(r)}. \end{aligned}$$

For every $((G', \mathcal{D}', k), M') \in \mathcal{Z}$ we have

$$\begin{aligned} |M'| &\le |M| + r^2(\lambda\zeta + k) && \text{by Lemma 8.21} \\ &= \mathcal{O}_r\left(kr + r^2(k^{1+\varepsilon\phi+\varepsilon} + k)\right) \\ &= \mathcal{O}_r\left(k^{1+\varepsilon(\phi+1)}\right). \end{aligned}$$

Therefore, given any such context, we can decide whether it is positive or not in time

$$\begin{aligned} 2^{g(p)|M'|^\alpha} n^{\mathcal{O}(1)} &= \exp\left(\mathcal{O}_r\left(p^\gamma \cdot \left(k^{1+\varepsilon(\phi+1)}\right)^\alpha\right)\right) \cdot n^{\mathcal{O}_r(1)} \\ &= \exp\left(\mathcal{O}_r\left(k^{\varepsilon\gamma+\alpha(1+\varepsilon(\phi+1))}\right)\right) \cdot n^{\mathcal{O}_r(1)} \end{aligned}$$

as follows: first we use Lemma 8.12 to delete irrelevant vertices and obtain an equivalent instance H on $|M'|$ vertices, then we use property $P_r(\phi, \gamma, \alpha)$ to bound the treewidth of H and then we solve the problem by dynamic programming on an approximate tree-decomposition in $2^{\mathcal{O}(\mathbf{tw}(H))} n^{\mathcal{O}(1)}$ time by noting that every r-clique and every $D' \in \mathcal{D}'$ (which is an $(< r)$-clique) has to be contained in a bag.

Thus, overall, computing $\mathcal{Z}$ and solving the problem in each sub-instance takes time

$$\begin{aligned} &\exp\left(\mathcal{O}_r\left((1+\varepsilon\phi)k^{1-\varepsilon}\log k + k^{\varepsilon\gamma+\alpha(1+\varepsilon(\phi+1))}\right)\right) \cdot n^{\mathcal{O}_r(1)} \\ = &\exp\left(\mathcal{O}_r\left((1+\varepsilon\phi)k^{1-\varepsilon}\log k + k^{\varepsilon(\gamma+\alpha+\alpha\phi)+\alpha}\right)\right) \cdot n^{\mathcal{O}_r(1)}. \end{aligned}$$

As we aim for algorithms where the contribution of k to the time complexity is of the form $2^{o(k)}$, the above bound sets the following constraint: $\varepsilon < \frac{1-\alpha}{\gamma+\alpha+\alpha\phi}$. Let $\varepsilon = \frac{1-\alpha-\delta}{\gamma+\alpha+\alpha\phi}$ for some constant $\delta \in (0,1)$ that we will fix later. Then the above complexity becomes:

$$\exp\left(\mathcal{O}_r\left((1+\varepsilon\phi)k^{1-\varepsilon}\log k + k^{1-\delta}\right)\right) \cdot n^{\mathcal{O}_r(1)}.$$

We optimize (ignoring logarithmic factors) by choosing the value of δ so that $1-\varepsilon = 1-\delta$, i.e., $\delta = \frac{1-\alpha}{\gamma+\alpha(\phi+1)+1}$. This gives the following overall running time bound:

$$2^{\mathcal{O}_r\left((1+\varepsilon\phi)k^{\varepsilon'}\log k\right)} \cdot n^{\mathcal{O}_r(1)} \qquad \text{with } \varepsilon' = \frac{\gamma+\alpha(\phi+2)}{\gamma+\alpha(\phi+1)+1}.$$

As $\alpha < 1$ we have $\varepsilon' < 1$ so the algorithm runs in subexponential FPT time, as desired. □

8.1.5 Applications

In this section, we give applications of Theorem 8.22 to specific graph classes. First, we have to show that the considered classes satisfy property P_r (recall that this property is defined at the beginning of Subsection 8.1.4).

Pseudo-disk graphs and map graphs

In this subsection, we will prove the following lemma.

Lemma 8.23. *Pseudo-disk graphs and map graphs have the property* $P_r(r-2, 1/2, 1/2)$.

As a consequence, we get the two following results.

Theorem 8.5. *There is an algorithm solving K_r-Hitting in pseudo-disk graphs in time* $2^{\mathcal{O}_r\left(k^{(r+1)/(r+2)}\log k\right)} \cdot n^{\mathcal{O}_r(1)}$.

Theorem 8.6. *There is an algorithm solving K_r-Hitting in map graphs in time* $2^{\mathcal{O}_r\left(k^{(r+1)/(r+2)}\log k\right)} \cdot n^{\mathcal{O}_r(1)}$.

To prove Lemma 8.23, we first need to state some external results.

In order to bound the number of small cliques in the considered graphs we can bound their degeneracy and then rely on the following result of Chiba and Nishizeki.

Theorem 8.24 ([CN85])**.** *Any string graph G with n vertices and degeneracy d has* $\mathcal{O}(id^{i-1}n)$ *i-cliques.*

Actually [CN85] gives a time bound for the enumeration of i-cliques in graphs of arboricity d. As arboricity and degeneracy are linearly bounded by each other and since the time bound implies a bound on the number of enumerated objects (up to a constant factor), we get the above statement.

Theorem 8.25 ([CGP02])**.** *Map graphs on n vertices with clique number ω have at most $7\omega n$ edges. In particular, they are (7ω)-degenerate.*

To bound the treewidth of map graphs and pseudo-disk graphs, we use the following combination of results on balanced separators of string graphs of Lee and the links between separators and treewidth of Dvořák and Norin.

As a consequence of the bounds in Theorem 8.25 and Theorem 2.9, using Theorem 2.12 we get the following.

Corollary 8.26. *Map graphs and pseudo-disk graphs on n vertices with clique number ω have treewidth* $\mathcal{O}(\sqrt{\omega n})$.

We are now ready to prove the wanted properties.

Proof of Lemma 8.23. Let G be a pseudo-disk graph with n vertices and clique number ω. By Theorem 2.9 the pseudo-disk graphs with n vertices and clique number ω are $(c\omega)$-degenerate for some constant c, so by Theorem 8.24 they have $\mathcal{O}((r-1)^2(c\omega)^{r-2}n)$ cliques of order less than r. Thus, property (P1) holds with $\phi = r-2$. By Corollary 8.26 property (P2) is satisfied with $\alpha = 1/2$ and $\gamma = 1/2$. The proof for map graphs is very similar, using Theorem 8.25 and Corollary 8.26. ◻

String graphs

We now move to string graphs where forbidding large bicliques is necessary. Actually for $K_{t,t}$-subgraph free string graphs the branching of Lemma 8.10 to reduce the clique number is not necessary in the algorithm of Theorem 8.22 as the number of small cliques and the treewidth are already suitably bounded. This explains the zeroes in Lemma 8.27 hereafter. In this subsection, we will prove the following lemma.

Lemma 8.27. *$K_{t,t}$-free string graphs have the property $P_r(0, 0, 1/2)$.*

As a consequence, we get the following result.

Corollary 8.28. *There is an algorithm solving K_r-Hitting in $K_{t,t}$-free string graphs in time*

$$2^{\mathcal{O}_{t,r}(k^{2/3}\log k)} \cdot n^{\mathcal{O}_r(1)}.$$

The contributions of t and r to the complexity in Corollary 8.28 are not explicit due to the way we stated the bound of Theorem 8.22 but can be extracted from the proof of this theorem.

Proof of Lemma 8.27. Combining Theorem 8.24 and Theorem 2.11 we get that the number of $(< r)$-cliques in $K_{t,t}$-free string graphs is $\mathcal{O}_r((t\log t)^{r-2}n)$, i.e. (P1) holds with $\phi = 0$. Theorem 2.12 gives (P2) with $\alpha = 1/2$ and $\gamma = 0$. ◻

Minor-closed classes

In this subsection, we will prove the following lemma.

Lemma 8.29. *For every graph H, H-minor-free graphs have the property $P_r(0, 0, 1/2)$.*

As a consequence, we get the following result.

Theorem 8.9. *For every graph H, there is an algorithm solving K_r-Hitting in H-minor-free graphs in time $2^{\mathcal{O}_{H,r}(k^{2/3}\log k)} \cdot n^{\mathcal{O}_r(1)}$.*

To prove Lemma 8.29, we first need to state some external results.

Theorem 8.30 (see [Woo07])**.** *Every d-degenerate graph with $n \geq d$ vertices has at most $2^d(n-d+1)$ cliques.*

Theorem 8.31 ([Kos82, Tho84])**.** *For every h-vertex graph H there is a constant $d = O\left(h\sqrt{\log h}\right)$ such that H-minor-free graphs are d-degenerate.*

Theorem 8.32 ([AST90]). *For every graph H, n-vertex H-minor-free graphs have treewidth $\mathcal{O}_H(\sqrt{n})$.*

Proof of Lemma 8.29. As a consequence of Theorem 8.31 and Theorem 8.30, H-minor-free graphs have a linear number of cliques (regardless of their clique number). Hence they satisfy (P1) with $\phi = 0$. By Theorem 8.32 they also satisfy (P2) with $\alpha = 1/2$ and $\gamma = 0$. □

8.2 The K_3-Hitting problem revisited

In this section, we provide a subexponential parameterized algorithm for TH in $K_{t,t}$-free d-DIR graphs, $K_{t,t}$-free string graphs, and contact segment graphs.

8.2.1 A subexponential FPT algorithm in classes with sublinear separators

The goal of this section is to prove the following result (it will be restated later in a more accurate form as Theorem 8.34).

Theorem 8.33. *If $\mathcal{G}$ is a hereditary graph class having strongly sublinear separators, then there is a subexponential time FPT algorithm for TH in $\mathcal{G}$.*

The proof of the following result follows the same steps as the proof in [LPS+22] for disk graphs, but as our statement is more general, we prefer to include it. It is of interest to us as one of its consequences (described below) is that $K_{t,t}$-free string graphs admit subexponential parameterized algorithm for TH, which is a positive counterpart to the lower-bound in Theorem 5.18.

Theorem 8.34. *If $\mathcal{G}$ is a hereditary graph class that has strongly sublinear separators, then there is a subexponential time algorithm for TH in $\mathcal{G}$.*
More precisely, if the size of the separators in $\mathcal{G}$ is $f(n) = \beta n^{\delta}$ for some $\delta < 1, \beta > 0$ and the ratio of neighborhood complexity is c, then the algorithm runs on n-vertex graphs of $\mathcal{G}$ with parameter k in time $2^{\mathcal{O}(\beta\gamma^{\delta}c^{\delta}k^{2\delta/(1+\delta)}\log k)}n^{\mathcal{O}(1)}$ for some universal constant γ.

Notice that by Lemma 4.2, the ratio of neighborhood complexity c used in the above statement is always defined for graph classes with polynomially bounded expansion.

Proof of Theorem 8.34. We prove the statement for the (more general) Weighted Triangle Hitting problem. This is the weighted version of TH where the vertices of the input graph have positive weights and the aim is to decide if there is a triangle hitting set whose sum of weights is at most the input parameter. Let $\alpha = \frac{1-\delta}{1+\delta}$. We apply Lemma 3.4 with $p = k^{\alpha}$, and thus in $2^{\mathcal{O}(k^{1-\alpha}\log k)}n^{\mathcal{O}(1)}$ time we generate the $2^{\mathcal{O}(k^{1-\alpha}\log k)}$ instances (G', k', M) as described in the aforementioned corollary. Let us now describe how we solve these instances.

By definition of c the vertices of $G' - M$ can be partitioned into at most $c|M|$ classes $V_1, \ldots, V_r$ such that two vertices in the same class have the same neighborhood in M. For every $i \in \{1, \ldots, r\}$ we delete all but one vertex of V_i, which we call v_i. Let G'' be the obtained graph. Let w be the weight function of the input instance. In order to keep track of the deleted vertices we define a new weight function $w'' \colon V(G'') \to \mathbb{N}$ as $w''(v_i) = \sum_{v \in V_i} w(v)$ for every $i \in \{1, \ldots, r\}$ and $w''(v) = w(v)$ for every $v \in M$. The produced instance is (G'', w'', k'); we call it a ***translation*** of the instance (G', k', M). Observe that it can be computed in polynomial time, and that it is an instance of Weighted Triangle Hitting.

Claim 8.35. *(G', w, k') is a yes-instance or TH $\Leftrightarrow$ (G'', w'', k') is a yes-instance of Weighted TH.*

Proof. Observe first the property $(P1)$ that, as for any $v \in M$, $N(v) \setminus M$ is an independent set, and M is a triangle hitting, any triangle Δ of G' (or G'') has $|\Delta \cap M| \geq 2$.

Direction "$\Leftarrow$". Let S'' be a solution of (G'', w'', k'). Observe that every vertex of S'' either belongs to M or is of the form v_i for some $i \in \{1, \ldots, r\}$ (using the same notation as above). Let $I \subseteq \{1, \ldots, r\}$ be the set of integers such that $S'' \setminus M = \{v_i : i \in I\}$. We define $S' = (S'' \cap M) \cup \bigcup_{i \in I} V_i$. By the definition of k' and w, we have $\sum_{v \in S'} w(v) \leq k'$. In order to show that S' is a triangle hitting set of G', we assume toward a contradiction that $G' - S'$ has a triangle Δ. If $|\Delta \cap M| = 3$ then Δ would be a triangle in $G'' - S''$, a contradiction, so by property $(P1)$ we have $|\Delta \cap M| = 2$, and thus let v_i be the representative in G'' of the vertex in $\Delta \setminus M$. By construction of S', this implies that $v_i \notin S''$, and even that $\Delta \cap S'' = \emptyset$, a contradiction.

Direction "$\Rightarrow$". Let S' be a subset-minimal solution of (G', w, k'). Observe that for every $i \in \{1, \ldots, r\}$, either $V_i \subseteq S'$ or $V_i \cap S' = \emptyset$. Indeed, if $v \in V_i$ belongs to S', by minimality of this set there is a triangle Δ containing v in $G' - (S' \setminus \{v\})$, and by property $(P1)$ it has its two other vertices in M. By the definition of V_i, every $u \in V_i$ is a neighbor of these two vertices. So, as S' is a triangle hitting set, $u \in S'$. Let $I = \{i \in \{1, \ldots, r\} : S \cap V_i \neq \emptyset\}$. Let $S'' = (S' \cap M) \cup \{v_i : i \in I\}$, and notice that this set has weight at most k'. Accordingly, it remains to show that S'' is a triangle hitting set of G''. Suppose toward a contradiction that $G'' - S''$ has a triangle Δ. As previously, if $|\Delta \cap M| = 3$ we have a contradiction, and thus $|\Delta \cap M| = 2$, and let v_i be the vertex in $\Delta \setminus M$. By definition of S'', $v_i \notin S'$ and thus Δ is a triangle of $G' - S'$, a contradiction.

□

In remains now to solve such an instance (G'', w'', k') of Weighted Triangle Hitting. Let $t = \mathbf{tw}(G'')$. We first compute in $2^{\mathcal{O}(t)}|G'|$ steps a tree decomposition of width $\mathcal{O}(t)$ (for instance using the approximation algorithm of Korhonen in Theorem 2.2). We then solve the problem in $2^{\mathcal{O}(t)}|G'|^{\mathcal{O}(1)}$ steps via a standard dynamic programming algorithm on the tree decomposition by observing that every triangle lies in some bag (as observed by [CMPP17] for the general problem of hitting cliques).

According to Lemma 3.4, $|M| \leq 11k^{1+\alpha}$ so $|G''| \leq 11(c+1)k^{1+\alpha}$. Recall that $\mathcal{G}$ is hereditary and G'' is an induced subgraph of G' that is an induced subgraph of G, so $G'' \in \mathcal{G}$. As $\mathcal{G}$ has strongly sublinear separators with bound f, by Theorem 2.4 the treewidth t of G'' is at most $15\beta|G''|^{\delta}$, so $t \leq 15\beta\gamma^{\delta}c^{\delta}k^{(1+\alpha)\delta}$ for some constant $\gamma > 0$. The number of instances generated by Lemma 3.4 is at most $2^{\mathcal{O}(k^{1-\alpha}\log k)}$ so overall the running time is

$$2^{\mathcal{O}(\beta\gamma^{\delta}c^{\delta}k^{(1+\alpha)\delta}+k^{1-\alpha}\log k)}n^{\mathcal{O}(1)} = 2^{\mathcal{O}(\beta\gamma^{\delta}c^{\delta}k^{2\delta/(1+\delta)}\log k)}n^{\mathcal{O}(1)},$$

as claimed. □

8.2.2 Application to $K_{t,t}$-free d-DIR, $K_{t,t}$-free string graphs and contact segment graphs

Corollary 2.13 and Theorem 8.34 applied with $\delta = 1/2$, $\beta = \mathcal{O}(\sqrt{t\log t})$ and for c any bound on the neighborhood complexity (that is bounded from above by a function δ, according to Lemma 4.2) lead to the following result.

Theorem 8.36. *For every integer t there is a constant $c_t > 0$ such that the following holds. There is an algorithm for TH in string graphs that runs on n-vertex instances with parameter k in time $2^{c_t k^{2/3}\log k}n^{\mathcal{O}(1)}$, where t is the minimum integer such that G is $K_{t,t}$-free, even when neither t nor a string representation are given as input.*

In Chapter 4 an explicit bound c_t was provided in $K_{t,t}$-free d-DIR and contact segment graphs. In order to remove big bicliques from graphs of these classes it is enough to remove big cliques as shown below.

Lemma 8.37. *A contact segment graph with clique number ω does not contain a $K_{12\omega,12\omega}$ as a subgraph.*

Proof. Towards a contradiction, assume there is a contact segment graph H with clique number ω containing a $K_{12\omega,12\omega}$-subgraph, with vertex sets $A = \{a_1,\dots,a_{12\omega}\}$ and $B = \{b_1,\dots,b_{12\omega}\}$. We denote by α_A and α_B the sizes of a maximum independent set in $H[A]$ and $H[B]$, respectively. As these graphs are not complete, we have $\alpha_A \geq 2$ and $\alpha_B \geq 2$.

Every triangle-free contact segment graph (as it only contains trivial contact points) is planar (see Lemma 2 in [DGMR18]), so in particular $K_{3,3}$ is not a contact segment graph. Therefore there is no induced $K_{3,3}$ in H, which implies that $\alpha_A \leq 2$ or $\alpha_B \leq 2$. In the following, we assume that $\alpha_A = 2$ and $\alpha_B \geq 2$. Let $b_1, b_2 \in B$ be non-adjacent.

We consider a maximal subset $A' \subseteq A$ whose segments touch b_1 on the same side, and touch b_2 on the same side. This set has size at least 2ω, as there are at most 4ω segments in A touching b_1 or b_2 on their endpoints, and as the other segments can be partitioned into four sets according to the sides of b_1 and b_2 they reach these segments.

As H is a contact segment graph, the segments in A' can be ordered $a'_1, \ldots, a'_{2\omega}, \ldots$ so that a'_i and a'_j, with $i < j$ do not intersect if $j - i \geq \omega - 1$ (otherwise, all the segments in-between and b_1 would intersect and form a $(\omega+1)$-clique). Thus, a'_1, a'_ω, and $a'_{2\omega}$ form an independent set, a contradiction to $\alpha_A = 2$. □

This leads to the following result, observe that in the case of contact segment graphs, an algorithm with the same running time for the more general case of pseudo-disk graphs was already obtained in Section 7.3.

Corollary 8.38. *There is an algorithm that solves TH in time*

- $2^{\mathcal{O}(k^{2/3}(\log k)\sqrt{d}t^2 \log t)} n^{\mathcal{O}(1)}$ *in $K_{t,t}$-free d-DIR graphs,*
- $2^{\mathcal{O}(k^{3/4} \log k)} n^{\mathcal{O}(1)}$ *in contact segment graphs,*

even when no representation is given.

Proof. For the first result, Corollary 2.13, together with Theorem 8.34 with $\delta = 1/2$, $\beta = \sqrt{t \log t}$, $c = \mathcal{O}(dt^3 \log t)$ (Lemma 4.7), implies the claimed running time.

For the second result, given an instance (G, k) of TH in contact segment graphs, we consider $p = k^\alpha$ (with α to be chosen), and proceed as in the proof of Theorem 8.34 by first generating the $2^{\mathcal{O}(k^{1-\alpha} \log(k))}$ instances (G'', k, w) of Weighted Triangle Hitting. As contact segment has linear neighborhood complexity, we have $|G''| = \mathcal{O}(|M|)$. Moreover, as G' is the result of applying Lemma 3.4, we have $\omega(G') \leq p$, and Lemma 8.37 implies that G' is $K_{t,t}$-free for $t = \mathcal{O}(p)$. By Corollary 2.13, this implies that $\mathbf{tw}(G'') = \mathcal{O}\left(\sqrt{|M| t \log(t)}\right) = \mathcal{O}\left(p\sqrt{k \log(k)}\right)$. By choosing $\alpha = 1/4$, and solving the instances in the same way as in Theorem 8.34, we obtain the wanted result. □

Chapter 9
Conclusion and research directions

9.1 Concluding remarks

In this thesis, progress was made on the optimality program for FPT subexponential algorithms in vertex deletion problems when restricted to geometric intersection graph classes.

The first direction of progress was to obtain new lower bounds for the complexity of solving cycle and clique-hitting problems, even when restricted to subclasses of segment graphs. The constructions give a good hint of the differences between hitting cycles and cliques, and why forbidding large bicliques can improve the running time of algorithms solving the considered problems.

A second direction of progress was to develop frameworks for obtaining subexponential FPT time algorithms, and solving vertex deletion problems for geometric graph classes. Instead of considering specific graph classes, the frameworks were developed for all geometric graph classes satisfying specific conditions. Those conditions take many forms, but most of them ask to bound some parameters of the considered graphs. Some parameters have a geometric definition, such as the local radius in Chapter 6 and the number of intersections between a pair of strings in Section 7.1. Another kind of considered parameters are the one that have a structural definition, with for example the size of a balanced separator, the tree neighborhood complexity in Section 7.1, or on the number of cliques in Chapter 8. In fact, for the frameworks developed in this thesis where no graphic representation are used, the graph classes for which the framework can be applied may go further than the scope of geometric intersection graphs classes.

9.2 Research directions

In this section, we highlight some questions whose answer would extend in some ways the results presented in this thesis.

9.2.1 Neighborhood complexity

Many of our algorithmic results rely on the result Theorem 4.12 from [ADEP21, Kes20], but it is not known if the bound is tight. Improving the upper bound on the number of classes would improve the running time of the algorithms solving FEEDBACK VERTEX SET for pseudo-disks and s-strings presented in Chapter 7. Another solution would be to get better bounds by using additional hypothesis satisfied by the (pseudo-)disks families used in our proof. Example of such properties satisfied are bounded ply, not being two disjoint families of pseudo-disks but only one, and having one of the family is independent.

Question 9.1. *Is the bound $\mathcal{O}(k^3)$ in Theorem 4.12 tight? What about variations of the graph classes considered (disks, only one family of pseudo-disks, one of the two families being an independent, bounded ply, etc...).*

In Chapter 4 we introduce a new kind of neighborhood complexity, namely subgraph neighborhood complexity, used in Section 7.1 when applied to s-string graphs and pseudo-disk graphs. It would be interesting to find new graph classes having the wanted properties, and to find new algorithmic application of this parameter as well.

9.2.2 Lower bounds

By looking at our results and constructions for lower bounds, one could conjecture that the deciding element for H-HITTING problem to not admit algorithms in time $2^{o(n)}$ is whether H is bipartite or not.

Question 9.2. *Is there a family $\mathcal{F}$ (not necessarily finite) containing a bipartite graph and such that solving $\mathcal{F}$-HITTING in time $2^{o(n)}$ for string graphs would contradict ETH?*

It could also be interesting to obtain lower bounds on the running time of problems for outerstring graphs and intersection graphs of circles.

9.2.3 Extending the ASQGM framework

This thesis introduced the new framework of Almost-SQGM, and examples of applications were given for the classes of square graphs and contact-segment graphs.

It would be interesting to find new graph classes being ASQGM, parameterized by the clique number and the local radius as in the two aforementioned examples or for other parameters of interest. A first step could be to extend the proof used for square graphs to disk graphs, where some of the geometric properties used in the proof for the squares no longer hold. Going further, it would be interesting to manage to unify the two given exemples by proving that pseudo-disk graphs have the ASQGM property. For proving this results many challenges would have to be tackled. A first one is that in order to apply Theorem 6.6, we need to approximate the biggest clique up to a constant. As discussed in Subsection 3.1.1, constant approximations are not known for pseudo-disk graphs. However when the problem we want to solve admits a $\mathcal{O}(1)$-approximation, such a clique approximation is not needed. A second obstacle would be to define a subneighborhood function having the wanted property. Ideas could be to look in the direction of combinatorial properties of the considered graph classes, such as weak coloring number (see [NOdM12] for the definition) or finding non-trivial generalization of disk properties to pseudo-disks, similar for example to the result in [Pin14], where a property known to be satisfied for any finite family of disks using its smallest element was generalized to any finite families of pseudo disks.

9.2.4 FPT algorithms for cycle hitting problems.

Odd Cycle Transversal In this thesis the main focus was on Feedback Vertex Set, but we almost ignored the similar Odd Cycle Transversal. If they are both hitting cycle problems of some kind, previous results as well as ours show that when it is question of subexponential algorithms, they differ a lot. Recall it was known that OCT for segments did not have subexponential time algorithms under ETH, and we proved the same result in contact segment graphs which are both segment graphs and pseudo-disk graphs. On the contrary in Chapter 7 we have provided algorithms solving FVS, both in pseudo-disk graphs and segment graphs (as 1-string graphs). In a sense the complexity of OCT seems more similar to the complexity of Triangle Hitting for example.

Balanced separators vs. FVS in subexponential FPT time.

Question 9.3. *What are examples of hereditary graph classes $\mathcal{G}$ with separators of size $\omega(G)^c n^\eta$ with $\eta < 1$ for n-vertices graphs, but such that FVS cannot be solved in subexponential FPT time for $\mathcal{G}$ assuming ETH?*

For FVS, the d-dimensional ball intersection graphs are example of such classes, for the dimension $d \geq 3$ [FLS18]. The case $d = 2$ do admit subexponential FPT algorithm as proved in [LPS+22].

FVS in string graphs. In Section 7.1 we gave general sufficient conditions for the existence of subexponential parameterized algorithms for the Feedback Vertex Set problem. Our main theorem unifies the previously known results on several classes of graphs such as planar graphs, map graphs, unit disk graphs, disk graphs, or more generally pseudo-disk graphs, and string graphs of bounded edge-degree. It also applies to classes where such algorithms were not known to exist, notably intersection graphs of thin objects such as segment graphs and more generally s-string graphs.

Question 9.4. *Can Feedback Vertex Set be solved in subexponential FPT time for string graphs?*

9.2.5 FPT algorithms for subgraph hitting

Our main results of Section 8.1 on a subexponential FPT algorithm solving K_r-Hitting could potentially be improved in many ways. A first idea would be to generalize the algorithm by removing the condition about the number of cliques, without worsening too much the complexity, in order to obtain a result with conditions similar to the more general Theorem 3.21 but with a better running time. Another way of improving the result is to remove the dependency in r in the running time.

For the question of existence of subexponential FPT time algorithms solving the H-Hitting problem for some graph H, Theorem 3.21 summarizes the vast majority of known results on the subject. Finding cases where such algorithm exists but where Theorem 3.21 does not apply would then be an important step for finding new conditions for the existence of a subexponential FPT algorithm in specific graph classes. One example of such algorithm out of the scope of Theorem 3.21 is Vertex-Cover for the class of outerstring graphs, which admit an algorithm with running time $2^{\widetilde{\mathcal{O}}(\sqrt{k})}n^{\mathcal{O}(1)}$ [AOX24]. Another class that could be of interest where the general theorem of Lokshtanov et al. do not apply are intersection graph of circles, and more generally intersection graph of sphere in dimension $d \geq 3$. More precisely, very recently, it was proved that for $d \geq 2$ the $K_{t,t}$-free n-vertex intersection graphs of d-dimensional sphere admit separators of size $\mathcal{O}\left(t^{10}n^{1-\frac{1}{2d+8}}\right)$ [DGHM25]. Note the in the case of FVS for $d = 2$ we obtain the graph intersection of circles, which are 2-strings graphs and so admit subexponential FPT algorithms by Corollary 7.6, and for $d \geq 3$ there is no subexponential FPT time algorithm solving FVS for intersection graphs of d-dimensional, by [FLS18] and with the observation that the intersection of unit balls is the intersection of unit spheres as well.

9.2.6 Subexponential vs. FPT subexponential

As we saw in Chapter 3, many problems have algorithms running in time $2^{\mathcal{O}(n^{1-\varepsilon})}$ for n vertices graphs with $\varepsilon > 0$. Getting a kernel of size $\mathcal{O}(k^{\varepsilon'})$ stable for the geometric intersection graph class considered with $\varepsilon' < \frac{1}{1-\varepsilon}$ would then yield a subexponential FPT algorithm. Some of the intersection graph classes we consider are stable by edge contraction. A kernel which is an induced minor is then also of interest.

Question 9.5. *For which graph class does FVS admit a (linear) kernel which is an induced minor graph of the original graph?*

For intermediate questions, the case of the class of outer-string graphs is once again interesting, as it admits an algorithm running in time $n^{\widetilde{\mathcal{O}}(\sqrt{k})}$, however, not only it is unknown if it admits a subexponential FPT algorithm, but our method presented in Theorem 7.5 cannot be applied. Indeed, when excluding a $K_{t,t}$ an outerstring graph has not necessarily linear neighborhood complexity with ratio $t^{\mathcal{O}(1)}$ (as shown in Figure 4.1), but this is a mandatory condition for Theorem 7.5 to apply.

Question 9.6. *Does* Feedback Vertex Set *admit a subexponential FPT algorithm for outer-string graphs?*

To answer this question, it would suffice to find a polynomial kernel for FVS in outer-string graphs, such that the obtained graph is still in the graph class. Indeed one could then use the $n^{\mathcal{O}(\sqrt{k})}$ algorithm for obtaining an $k^{\widetilde{\mathcal{O}}\left(\sqrt{k}\right)}n^{\mathcal{O}(1)}$-time algorithm. As the class of outer-string graphs is stable by taking an induced minor graph, a positive answer to the following question would then be sufficient for answering by the positive Question 9.6.

Question 9.7. *Does* Feedback Vertex Set *admit a polynomial kernel that is an induced minor of the original graph?*

Question 9.8. *Can* Vertex Cover *be solved in time $2^{O\left(\sqrt{k}\right)}n^{\mathcal{O}(1)}$ for string graphs?*

Observe that this question is open even for disks if no representation is given.

9.2.7 More problems and more parameters

Subgraph and cycle hitting problems are not the only (parameterized) problems of interest, and questions similar to the ones asked in this thesis could be asked for those problems as well. A first example could be Induced-$\mathcal{F}$-Hitting for a family of graph $\mathcal{F}$. For $\mathcal{F} = \{K_r\}$ for some r, there is no difference with the problem K_r-Hitting investigated in Chapter 8. In the case of $\mathcal{F}$ finite, the authors of [LPS+24b] made the observation that for $k = 0$ and the graph H being an independent set of size p, the Induced-H-Hitting problem is equivalent to decide if there exists an independent set of size p.

FVS is the Minor-H-Hitting problem with $H = K_3$. Other problems of this family would be interesting to study. Observe that if H is planar, Minor-H-Hitting is a bidimensional problem.

Other parameters can also be considered, with for example a recent idea of considering modifications of the geometric representation instead of removing vertices, in the considered case changing the radius of the disks in a representation of disk graph (see [FGI+25]).

Bibliography

[ACO23] Shinwoo An, Kyungjin Cho, and Eunjin Oh. Faster algorithms for cycle hitting problems on disk graphs. In *Algorithms and Data Structures: 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 – August 2, 2023, Proceedings*, page 29–42, Berlin, Heidelberg, 2023. Springer-Verlag. Cited on pages 45, 50, 129, 130, and 161.

[ADEP21] Boris Aronov, Anirudh Donakonda, Esther Ezra, and Rom Pinchasi. On pseudo-disk hypergraphs. *Computational Geometry*, 92:101687, 2021. Cited on pages 56 and 182.

[AO21] Shinwoo An and Eunjin Oh. Feedback Vertex Set on Geometric Intersection Graphs. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, volume 212 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:12, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Cited on pages 43, 76, and 94.

[AOX24] Shinwoo An, Eunjin Oh, and Jie Xue. Sparse Outerstring Graphs Have Logarithmic Treewidth. In Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Cited on pages 46, 47, and 184.

[AST90] Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990. Cited on page 177.

[BBB+21] Marthe Bonamy, Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Panos Giannopoulos, Eun Jung Kim, Paweł Rzążewski, Florian

Sikora, and Stéphan Thomassé. Eptas and subexponential algorithm for maximum clique on disk and unit ball graphs. *J. ACM*, 68(2), jan 2021. Cited on pages 38 and 129.

[BBF99] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999. Cited on page 27.

[BBGR24a] Gaétan Berthe, Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond. Kick the Cliques. In Édouard Bonnet and Paweł Rzążewski, editors, *19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*, volume 321 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Cited on page 20.

[BBGR24b] Gaétan Berthe, Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond. Subexponential Algorithms in Geometric Graphs via the Subquadratic Grid Minor Property: The Role of Local Radius. In Hans L. Bodlaender, editor, *19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024)*, volume 294 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Cited on page 20.

[BBGR25a] Gaétan Berthe, Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond. Feedback vertex set for pseudo-disk graphs in subexponential fpt time. In Daniel Kráľ and Martin Milanič, editors, *Graph-Theoretic Concepts in Computer Science*, pages 65–78, Cham, 2025. Springer Nature Switzerland. Cited on page 20.

[BBGR25b] Gaétan Berthe, Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond. Pushing the Frontiers of Subexponential FPT Time for Feedback Vertex Set. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*, volume 334 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:15, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Cited on page 20.

[BG96] Ann Becker and Dan Geiger. Optimization of pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83(1):167–188, 1996. Cited on page 27.

[BGM20] Édouard Bonnet, Nicolas Grelier, and Tillmann Miltzow. Maximum clique in disk-like intersection graphs. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, December 14-18, 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference)*, volume 182 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. Cited on pages 38, 82, and 83.

[BLL+22] Sayan Bandyapadhyay, William Lochet, Daniel Lokshtanov, Saket Saurabh, and Jie Xue. True Contraction Decomposition and Almost ETH-Tight Bipartization for Unit-Disk Graphs. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Cited on page 46.

[BR19] Édouard Bonnet and Paweł Rzążewski. Optimality program in segment and string graphs. *Algorithmica*, 81:3047–3073, 2019. Cited on pages 36, 48, 62, and 94.

[BR24] Edouard Bonnet and Paweł Rzążewski. An 11/6-approximation algorithm for vertex cover on string graphs, 2024. https://arxiv.org/abs/2409.18820, to appear in SOCG 2025. Cited on page 33.

[BST19] Julien Baste, Ignasi Sau, and Dimitrios M Thilikos. Hitting minors on bounded treewidth graphs. iv. an optimal algorithm. *arXiv preprint arXiv:1907.04442*, 2019. Cited on page 75.

[BT22] Julien Baste and Dimitrios M Thilikos. Contraction bidimensionality of geometric intersection graphs. *Algorithmica*, 84(2):510–531, 2022. Cited on pages 43, 74, 80, and 94.

[CFJ04] Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $o(n^2)$ steps. In Juraj Hromkovic, Manfred Nagl, and Bernhard Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science, 30th International*

Workshop,WG 2004, Bad Honnef, Germany, June 21-23, 2004, Revised Papers, volume 3353 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2004. Cited on page 44.

[CFK+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015. Cited on pages 27, 28, 36, 37, 124, 131, 134, and 138.

[CGP02] Zhi-Zhong Chen, Michelangelo Grigni, and Christos H Papadimitriou. Map graphs. *Journal of the ACM (JACM)*, 49(2):127–138, 2002. Cited on page 175.

[CJ03] Liming Cai and David Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):789–807, 2003. Cited on page 63.

[CKU98] Jurek Czyzowicz, Evangelos Kranakis, and Jorge Urrutia. A simple proof of the representation of bipartite planar graphs as the contact graphs of orthogonal straight line segments. *Inf. Process. Lett.*, 66(3):125–126, 1998. Cited on page 69.

[CMPP17] Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michał Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. *Information and Computation*, 256:62–82, 2017. Cited on pages 75 and 179.

[CN85] Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223, 1985. Cited on page 175.

[CS89] Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, ii. *Discrete & Computational Geometry*, 4(5):387–421, Oct 1989. See `https://courses.grainger.illinois.edu/cs498sh3/sp2023/lec/25/depth.pdf` for a comprehensive presentation. Cited on page 29.

[CT21] Julia Chuzhoy and Zihan Tan. Towards tight (er) bounds for the excluded grid theorem. *Journal of Combinatorial Theory, Series B*, 146:219–265, 2021. Cited on page 42.

[dBBKB+20] Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for exponential-time-hypothesis–tight algorithms and lower bounds in geometric intersection graphs. *SIAM Journal on Computing*, 49(6):1291–1331, 2020. Cited on pages 44 and 61.

[dBKBMT23] Mark de Berg, Sándor Kisfaludi-Bak, Morteza Monemizadeh, and Leonidas Theocharous. Clique-based separators for geometric intersection graphs. *Algorithmica*, 85(6):1652–1678, 2023. Cited on pages 44 and 130.

[DFHT05] Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *Journal of the ACM (JACM)*, 52(6):866–893, 2005. Cited on pages 41, 42, and 43.

[DGHM25] James Davies, Agelos Georgakopoulos, Meike Hatzel, and Rose McCarty. Strongly sublinear separators and bounded asymptotic dimension for sphere intersection graphs, 2025. Cited on page 184.

[DGMR18] Zakir Deniz, Esther Galby, Andrea Munaro, and Bernard Ries. On contact graphs of paths on a grid. In *Graph Drawing and Network Visualization: 26th International Symposium, GD 2018, Barcelona, Spain, September 26-28, 2018, Proceedings 26*, pages 317–330. Springer, 2018. Cited on page 180.

[DH08] Erik D Demaine and MohammadTaghi Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008. Cited on page 41.

[DN16] Zdenek Dvorák and Sergey Norin. Strongly sublinear separators and polynomial expansion. *SIAM Journal on Discrete Mathematics*, 30(2):1095–1101, 2016. Cited on page 26.

[DN19] Zdeněk Dvořák and Sergey Norin. Treewidth of graphs with balanced separations. *Journal of Combinatorial Theory, Series B*, 137:137–144, 2019. Cited on page 26.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. Cited on page 24.

[ER18] Louis Esperet and Jean-Florent Raymond. Polynomial expansion and sublinear separators. *European Journal of Combinatorics*, 69:49–53, 2018. Cited on page 26.

[FGI+25] Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, Saket Saurabh, and Meirav Zehavi. Parameterized Geometric Graph Modification with Disk Scaling. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, volume 325 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:17, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Cited on page 186.

[FLP+19] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. *Discrete & Computational Geometry*, 62:879–911, 2019. Cited on pages 43 and 76.

[FLRS11] Fedor V Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 748–759. SIAM, 2011. Full version available at `https://arxiv.org/abs/1005.5449`. Cited on page 160.

[FLS18] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Excluded grid minors and efficient polynomial-time approximation schemes. *Journal of the ACM (JACM)*, 65(2):1–44, 2018. Cited on pages 41, 43, 61, 75, 76, 183, and 184.

[FLST20] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. *SIAM Journal on Computing*, 49(6):1397–1422, 2020. Cited on page 43.

[FP11] Jacob Fox and János Pach. Computing the independence number of intersection graphs. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1161–1165. SIAM, 2011. Cited on page 36.

[GKT14] Alexander Grigoriev, Athanassios Koutsonas, and Dimitrios M Thilikos. Bidimensionality of geometric intersection graphs. In *SOFSEM 2014: Theory and Practice of Computer Science: 40th International Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 26-29, 2014, Proceedings 40*, pages 293–305. Springer, 2014. Cited on pages 43, 75, and 76.

[GL20] Peter Gartland and Daniel Lokshtanov. Independent set on P_k-free graphs in quasi-polynomial time. In *2020 IEEE 61st Annual*

Symposium on Foundations of Computer Science (FOCS), pages 613–624, 2020. Cited on page 62.

[GOR+19] Carla Groenland, Karolina Okrasa, Paweł Rzążewski, Alex Scott, Paul Seymour, and Sophie Spirkl. H-colouring pt-free graphs in subexponential time. *Discrete Applied Mathematics*, 267:184–189, 2019. Cited on pages 48 and 62.

[Gre22] Nicolas Grelier. Computing a maximum clique in geometric superclasses of disk graphs. *Journal of Combinatorial Optimization*, 44(4):3106–3135, Nov 2022. Cited on page 38.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. Cited on page 27.

[JR24] Gwenaël Joret and Clément Rambaud. Neighborhood complexity of planar graphs. *Combinatorica*, 44(5):1115–1148, Oct 2024. Cited on page 50.

[Kes20] Balázs Keszegh. Coloring intersection hypergraphs of pseudo-disks. *Discret. Comput. Geom.*, 64(3):942–964, 2020. Cited on pages 56 and 182.

[KK06] Jan Kára and Jan Kratochvíl. Fixed parameter tractability of independent set in segment intersection graphs. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 166–174. Springer, 2006. Cited on page 62.

[KLPS86] Klara Kedem, Ron Livne, János Pach, and Micha Sharir. On the union of jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete & Computational Geometry*, 1(1):59–71, Mar 1986. Cited on page 29.

[KM94] Jan Kratochvíl and Jirí Matousek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994. Cited on page 34.

[KM24] J. Mark Keil and Debajyoti Mondal. The maximum clique problem in a disk graph made easy. *ArXiv*, abs/2404.03751, 2024. Cited on page 38.

[Koe36] Paul Koebe. Kontaktprobleme der konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physikalische Klasse*, 88:141–164, 1936. Cited on pages 10, 16, and 32.

[Kor22] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–192. IEEE, 2022. Cited on page 24.

[Kos82] A Kostochka. On the minimum of the Hadwiger number for graphs with given average degree. *Metody Diskret. Analiz.*, 38:37–58, 1982. English translation: AMS Translations (2), 132:15-32, 1986. Cited on page 176.

[Kra96] Jan Kratochvíl. Intersection graphs of noncrossing arc-connected sets in the plane. In Stephen C. North, editor, *Graph Drawing, Symposium on Graph Drawing, GD '96, Berkeley, California, USA, September 18-20, Proceedings*, volume 1190 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 1996. Cited on pages 138 and 154.

[Lee17] James R. Lee. Separators in Region Intersection Graphs. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:8, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. Cited on pages 32 and 33.

[Lic82] David Lichtenstein. Planar formulae and their uses. *SIAM journal on computing*, 11(2):329–343, 1982. Cited on page 68.

[LPS+22] Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. Subexponential parameterized algorithms on disk graphs (extended abstract). In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2031. SIAM, 2022. Full version available at `https://sites.cs.ucsb.edu/~daniello/papers/subexpDiskOCTandFriends.pdf`. Cited on pages 39, 40, 44, 45, 50, 76, 98, 115, 127, 129, 131, 163, 177, and 183.

[LPS+23] Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. A framework for approximation schemes on disk graphs. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2228–2241. SIAM, 2023. Cited on pages 24, 74, 79, and 80.

[LPS+24a] Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. Bipartizing (Pseudo-)Disk Graphs: Approximation with a Ratio Better than 3. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024)*, volume 317 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:14, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Cited on page 32.

[LPS+24b] Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. Subexponential parameterized algorithms for hitting subgraphs, 2024. Accepted in STOC 2025. Cited on pages 47 and 186.

[LT79] Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. Cited on page 26.

[LY80] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980. Cited on page 61.

[Mar05] Dániel Marx. Efficient approximation schemes for geometric problems? In Gerth Stolting Brodal and Stefano Leonardi, editors, *Algorithms – ESA 2005*, pages 448–459, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. Cited on page 62.

[Mar06] Dániel Marx. Parameterized complexity of independence and domination on geometric graphs. In *Proceedings of the Second International Conference on Parameterized and Exact Computation*, IWPEC'06, page 154–165, Berlin, Heidelberg, 2006. Springer-Verlag. Cited on page 62.

[Mat14] Jirí Matousek. Near-optimal separators in string graphs. *Comb. Probab. Comput.*, 23(1):135–139, 2014. Cited on page 33.

[MTTV97] Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, January 1997. Cited on page 33.

[MU04] Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In Torben Hagerup and Jyrki Katajainen, editors, *Algorithm Theory - SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, Humlebaek, Denmark, July 8-10, 2004, Proceedings*, volume 3111 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 2004. Cited on page 90.

[NOdM12] Jaroslav Neetil and Patrice Ossona de Mendez. *Sparsity: Graphs, Structures, and Algorithms*. Springer Publishing Company, Incorporated, 2012. Cited on pages 25 and 183.

[NOP+21] Jana Novotná, Karolina Okrasa, Michał Pilipczuk, Paweł Rzążewski, Erik Jan van Leeuwen, and Bartosz Walczak. Subexponential-time algorithms for finding large induced sparse subgraphs. *Algorithmica*, 83:2634–2650, 2021. Cited on pages 48 and 61.

[OR20] Karolina Okrasa and Paweł Rzążewski. Subexponential algorithms for variants of the homomorphism problem in string graphs. *Journal of Computer and System Sciences*, 109:126–144, 2020. Cited on pages 48, 62, and 63.

[Pin14] Rom Pinchasi. A finite family of pseudodiscs must include a "small" pseudodisc. *SIAM Journal on Discrete Mathematics*, 28(4):1930–1934, 2014. Cited on page 183.

[PPR21] Marcin Pilipczuk, Michal Pilipczuk, and Pawel Rzazewski. Quasi-polynomial-time algorithm for independent set in P_t-free graphs via shrinking the space of induced paths. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 204–209. SIAM, 2021. Cited on page 62.

[PRS94] Serge Plotkin, Satish Rao, and Warren D. Smith. Shallow excluded minors and improved graph decompositions. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '94, page 462–470, USA, 1994. Society for Industrial and Applied Mathematics. Cited on page 26.

[RVS19] Felix Reidl, Fernando Sánchez Villaamil, and Konstantinos Stavropoulos. Characterising bounded expansion by neighbourhood complexity. *European Journal of Combinatorics*, 75:152–168, 2019. Cited on page 50.

[Tho84] Andrew Thomason. An extremal function for contractions of graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 95, pages 261–265. Cambridge University Press, 1984. Cited on page 176.

[Wag37] K. Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–590, Dec 1937. Cited on page 25.

[Woo07] David R Wood. On the maximum number of cliques in a graph. *Graphs and Combinatorics*, 23(3):337–352, 2007. Cited on page 176.

Problem definitions

Vertex deletion problems

Given Π a graph property:

Π-Vertex delete	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \leq k$ and for any $H \in \mathcal{F}$, the graph $G - S$ satisfies the property Π?

Given $\mathcal{F}$ a family of graphs:

$\mathcal{F}$-Hitting	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \leq k$ and for all $H \in \mathcal{F}$, $G - S$ does not contain H as a subgraph?

For H a graph, the $\{H\}$-Hitting problem is denoted H-Hitting.
The K_3-Hitting may be denoted Triangle Hitting.

$\mathcal{F}$-minor-Hitting	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \leq k$ and for any $H \in \mathcal{F}$, the graph $G - S$ does not contain H as a minor?

Vertex Cover	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \leq k$ and $G - S$ is edgeless?

Feedback Vertex Set	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \leq k$ and $G - S$ is a forest?

Odd Cycle Transversal	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \leq k$ and $G - S$ is bipartite?

Pseudo Forest Del	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \leq k$ and the connected components of $G - S$ have at most one cycle?

General problems

Maximum Induced Matching	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \geq k$ and the connected components of $G - S$ are all of size 2?

Dominating Set	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \leq k$ and S is a dominating set?

Independent Set	
Input:	A graph G
Parameter:	k
Question:	Is there a set $S \subseteq V(G)$ such that $\lvert S \rvert \leq k$ and S is an independent set?

k-Coloring	
Input:	A graph G
Question:	Is there a function $\varphi : V(G) \to \{1, \ldots, k\}$ such that for $uv \in E(G)$, $\varphi(u) \neq \varphi(v)$?

Given a graph G, a function $w : (V(G) \times V(H)) \cup (E(G) \times E(H)) \to \mathbb{R}$, and an integer k. For a homomorphism $h : G \to H$ the weight of h is defined as $\sum_{x \in V(G) \cup E(G)} w(x, h(x))$.

WHom(H)	
Input:	A graph G, a weight function $w : (V(G) \times V(H)) \cup (E(G) \times E(H)) \to \mathbb{R}$
Parameter:	k
Question:	Is there a homomorphism $h : G \to H$ whose total weight is at least k?

Logic problems

k-SAT	
Input:	A propositional formula φ, in conjunctive normal form, where every clause consists of at most k literals.
Question:	Does there exist a satisfying assignment for φ?

Planar-k-SAT	
Input:	A propositional formula φ, in conjunctive normal form, where every clause consists of at most k literals. Moreover the graph with vertices the variable and an edge between two variables appearing in a same clause is planar.
Question:	Does there exist a satisfying assignment for φ?