Some Contributions to Parameterized Complexity

Ignasi Sau

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

Université de Montpellier

Speciality: Computer Science

Habilitation à Diriger des Recherches (HDR)

Ignasi Sau Valls

Some Contributions to Parameterized Complexity

Quelques Contributions en Complexité Paramétrée

Version of June 21, 2018 – Defended on June 25, 2018

Committee:

Reviewers:	Michael R. Fellows	-	University of Bergen (Norway)
	Fedor V. Fomin	-	University of Bergen (Norway)
	Rolf Niedermeier	-	Technische Universität Berlin (Germany)
Examinators:	Jean-Claude Bermond	-	CNRS, U. de Nice-Sophia Antipolis (France)
	Marc Noy	-	Univ. Politècnica de Catalunya (Catalonia)
	Dimitrios M. Thilikos	-	CNRS, Université de Montpellier (France)
	Gilles Trombettoni	-	Université de Montpellier (France)

Contents

Chapter 0

Résumé et projet de recherche

Nous présentons ici un petit résumé de ce manuscrit, ainsi que quelques pistes de recherche pour les années à venir, en relation avec le contenu scientifique exposé dans le manuscrit.

Résumé

Ce document contient une synthèse de mes principales contributions scientifiques après ma soutenance de thèse, qui a eu lieu en octobre 2009, et surtout pendant le temps que j'ai passé au LIRMM (Montpellier, France) en tant que Chargé de Recherche au CNRS.

Au lieu d'une compilation exhaustive de toutes mes contributions, ce qui serait certainement trop long et trop hétérogène, ce manuscrit contient deux parties principales. Dans la première, on présente une description courte de toutes mes contributions, en les mettant en contexte et en énonçant les résultats principaux, mais sans rentrer dans les détails scientifiques. Cette description permet d'avoir une vision globale sur l'ensemble de mes contributions. Dans la seconde partie, on présente quatre de ces contributions avec tous les détails. Ces quatre contributions ont été choisies avec le critère d'être représentatives à la fois sur les thématiques sur lequelles elles portent, et aussi sur le type de techniques utilisées pour obtenir les résultats.

En rentrant sur la thématique de mes recherches, la grande plupart de mes contributions se placent dans le cadre de la *complexité paramétrée*. Cette théorie a été développée par Downey et Fellows dans les années 90, et aujourd'hui est déjà un domaine de recherche complètement établi avec, par exemple, quatre bouquins publiés sur le sujet, et de centaines d'articles acceptés annuellement dans les plus prestigieuses conférences internationales en informatique théorique. Nous allons maintenant introduire de manière succincte les idées principales de la complexité paramétrée, pour pouvoir mettre en contexte mes contributions scientifiques.

La théorie de la NP-complétude "classique" considère qu'un problème est *facile* s'il admet un algorithme polynomial pour le résoudre. Mais beaucoup de problèmes sont NP-difficiles, c'est-à-dire, ils n'admettent pas de tels algorithmes à moins que $\mathsf{P} = \mathsf{NP}$, une hypothèse que, au jour d'aujourd'hui, est considérée de manière presque unanime comme essez peu probable dans la communauté.

Cependant, une fois qu'un problème est montré NP-difficile, que faire si on doit malgré tout le résoudre? Plusieurs méthodologies peuvent être envisagées, les plus connues étant les suivantes: algorithmes d'approximation, algorithmes randomisés, heuristiques, ou bien algorithmes exponentiels exacts. Logiquement, chacune de ces méthodes a des avantages et désavantages.

L'algorithmique à paramètre fixée se place dans l'approche des algorithmes exponentiels exacts, notamment en proposant une analyse de la complexité *bidimensionnelle*, et donc plus fine que l'analyse classique. La motivation principale est d'essayer d'obtenir une meilleure vision de la complexité d'un problème en explorant comment certains paramètres, spécifiques du problème ou des instances, influencent cette complexité.

Idéalement, on cherche à identifier un paramètre k tel que si k est petit alors le problème peut être résolu *efficacement*. Mais qu'est-ce qu'on veut dire par "efficacement"? La notion fondamentale de la théorie paramétrée est la définition suivante d'algorithme "efficace": un algorithme tel que son temps d'exécution peut être borné par $f(k) \cdot |x|^{\mathcal{O}(1)}$, où f est une fonction quelconque qui ne dépend que du paramètre k, et $|x|$ représente la taille de l'entrée du problème. Les problèmes paramétrés qui possèdent un tel algorithme sont appelés *solubles a paramètre fixé* (en anglais, *fixed-parameter tractable*), et notés FPT.

Il existe de techniques très puissantes et très générales pour démontrer que de familles entières de problèmes paramétrés sont FPT. Par exemple, le théorème de Courcelle sur la logique monadique du second ordre et les graphes de largeur arborescente bornée, ou le théorème de Robertson et Seymour sur les mineurs de graphes, impliquent l'*existence* d'algorithmes FPT pour tout problème qui satisfait certaines conditions générales. Mais le prix qu'on doit payer pour une telle généralité est que les algorithmes FPT qui découlent de ces "meta-théorèmes" ont un temps d'exécution extrêmement élevé. En plus, souvent on ne connaît même pas une borne supérieure sur le temps d'exécution, ou encore pire, on peut ne même pas connaître explicitement quel est l'algorithme qu'on cherche; on sait juste qu'il *existe*.

Motivé par la problematique évoquée ci-dessus, une bonne partie de mes recherches porte sur la conception d'algorithmes FPT qui soient, souhaitablement, *efficaces*, *constructifs*, et avec de temps d'exécution *explicites*. Une de mes contributions principales dans cet axe de recherche concerne le problème de Planar $\mathcal{F}$-Deletion: pour une famille finie et fixée de graphes $\mathcal{F}$ contenant au moins un graphe planaire, et étant donné un graphe G et un entier k en tant que paramètre, le but est de décider si l'on peut supprimer au plus k sommets de G pour obtenir un graphe qui ne contient aucun des graphes dans $\mathcal{F}$ en tant que mineur. Ce problème généralise, par exemple, les problèmes de Vertex Cover et Feedback Vertex Set. Nous avons obtenu le premier algorithme pour résoudre Planar $\mathcal{F}$-Deletion, pour une famille quelconque $\mathcal{F}$ contenant un graphe planaire, en temps *exponentiel simple*, c'est-à-dire, en temps $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, où n est le nombre de sommets de G. En plus, cet algorithme est asymptotiquement optimal si l'on suppose l'*hypothèse de temps exponentiel* (ETH en anglais).

En complexité paramétrée, un autre concept central et en forte relation avec les algorithmes FPT est celui de *kernelization*, qui pourrait être traduit en français par *extraction de noyau*. Formellement, un algorithme de kernelization, ou juste *kernel* (en français, *noyau*), pour un problème paramétrée Π, reçoit une instance (x, k) du problème et, en temps polynomial en $|x|+k$, calcule une instance (x', k') telle que $|x'|, k' \leqslant g(k)$ pour une certaine fonction g, et $(x, k) \in \Pi$ si et seulement si $(x', k') \in \Pi$. La fonction g est connue comme la *taille* du noyau, et peut être interprétée comme le degré de "compressibilité" d'un problème en utilisant de règles de pretraitement polynomiales.

Un noyau est *polynomial* (resp. *linéaire*) si la fonction g est une fonction polynomiale (resp. linéaire) en k. Il est bien connu que tout problème décidable est FPT si et seulement s'il admet un noyau. Mais typiquement la taille des noyaux qu'on obtient de cette manière est exponentielle en fonction du paramètre k. Dans ce contexte, un problème naturel est de décider si un problème FPT admet on noyau polynomial ou pas. Comme c'est le cas des algorithmes FPT, il existe de "méta-noyaux" pour déduire l'*existence* de noyaux polynomiaux ou linéaires pour tout problème et classe de graphes qui satisfont certaines conditions générales. Mais encore une fois, souvent ces méta-noyaux ne permettent pas d'obtenir une fonction g *explicite*, ou même de *construire* le noyau qu'on cherche.

Une bonne partie de mes contributions en kernelization se placent dans cette problématique. Un de mes résultats principaux ici est une méthodologie pour obtenir de noyaux linéaires, de manière *constructive* et avec de constantes *explicites*, pour des familles très larges de problèmes dans les graphes *sparses*, c'est-à-dire, de graphes qui ont un nombre faible d'arêtes par rapport à leur nombre de sommets, comme les graphes planaires, les graphes qui peuvent être dessinés sur une surface fixée, ou les graphes qui excluent un graphe fixé en tant que mineur ou mineur topologique. Notre méthodologie est basée sur une formalisation de la technique de programmation dynamique sur une décomposition arborescente du graphe en entrée, ce qui permet d'abstraire les propriétés critiques qu'il faut préserver lors qu'on veut résoudre un certain problème dans ces classes de graphes, pour obtenir de manière constructive un noyau avec de constantes explicites.

Même si mon domaine principal de recherche est celui de la complexité paramétrée, j'ai aussi investi une bonne partie de mon temps à l'étude de problèmes purement *combinatoires*, la plupart motivés par l'étude de problèmes paramétrés. À titre d'exemple, on dit qu'une classe de graphes $\mathcal{G}$ satisfait la *propriété d'Erdős-Pósa* s'il existe une fonction $f_{\mathcal{G}} : \mathbb{N} \to \mathbb{N}$ telle que, pour tout graphe G et tout entier positif k, soit G contient k sous-graphes deux à deux disjoints en sommets, tous isomorphes à un graphe dans la classe $\mathcal{G}$, soit il contient un ensemble S de sommets, avec $|S| \leqslant f_{\mathcal{G}}(k)$, tel que $G - S$ ne contient aucun sous-graphe isomorphe à un graphe dans $\mathcal{G}$. Quand cette propriété est satisfaite pour une classe $\mathcal{G}$, la fonction $f_{\mathcal{G}}$ est connue comme le *gap* de la propriété d'Erdős-Pósa pour la classe $\mathcal{G}$. Avec ces définitions, le théorème classique d'Erdős et Pósa dit que la classe contenant tous les cycles satisfait la propriété d'Erdős-Pósa avec gap $\mathcal{O}(k \cdot \log k)$.

Quand on sait que une classe de graphes $\mathcal{G}$ satisfait la propriété d'Erdős-Pósa, le problème naturel associé d'un point de vu combinatoire est de trouver le plus petit gap $f_{\mathcal{G}}$ possible. En particulier, j'ai étudié le cas où la classe de graphe $\mathcal{G}$ est définie comme tous les graphes qui contient un graphe fixé H en tant que mineur. Ce cas a été beaucoup étudié dans

la littérature, et Robertson et Seymour ont demontré que la propriété d'Erdős-Pósa est satisfaite si et seulement si le graphe H est planaire. Cependant, on ne sait pas encore quel est le plus petit gap f_H possible pour un graphe planaire quelconque H. En utilisant de techniques de décomposition de graphes, nous avons trouvé les meilleurs gaps possibles (asymptotiquement) quand H appartient à une de ces deux familles de graphes planaires: les *citrouilles* et les *roues*.

Finalement, je me suis aussi intéressé à de problèmes qui sont motivés par des *applications pratiques*. Par exemple, de problèmes issus des réseaux de télécommunications, ou de problèmes qui viennent de la bioinformatique. Mes contributions dans ce genre de problèmes sont diverses: algorithmes polynomiaux et FPT, ou bien preuves de NP-complétude et de non-existence d'algorithmes FPT.

Pour conclure ce petit résumé, j'aimerais rajouter que toutes mes recherches ont été effectuées en collaboration avec mes nombreux collègues. Un grand merci à vous tous!

Projet de recherche

Dans les anées a venir, je compte continuer à travailler en complexité paramétrée, en particulier sur des algorithmes FPT et la kernelization. J'aimerais aussi continuer l'étude des problèmes combinatoires auxquels je me suis attaqué.

En particulier, la liste qui suit contient quelques problèmes particuliers que j'ai rencontré lors de mes travaux de recherche, et je que voudrais essayer de résoudre, ou étudier en détail:

1. On a exploré le rôle joué par la planarité dans les problèmes de connexité paramétrés par la largeur arborescente, comme Longest Path ou Cycle Packing. Un problème notoire reste ouvert: est-ce qu'on peut résoudre Planar Disjoint Paths paramétré par la largeur arborescente en temps exponentiel simple dans les graphes planaires? Dans les graphes généraux, on sait résoudre le problème en temps $2^{\mathcal{O}(\mathsf{tw}\log\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, et que ce temps est optimal si on suppose l'ETH, mais on ne connaît pas la réponse dans les graphes planaires.

2. On a étudié la complexité du problème $\mathcal{F}$-Deletion paramétrée, encore une fois, par la largeur arborescente. Ce problème est défini comme Planar-$\mathcal{F}$-Deletion, mais sans la condition que la famille $\mathcal{F}$ doit contenir un graphe planaire. Trouver la complexité "exacte" de ce problème (dans le sens de trouver un temps d'exécution qui soit optimal en supposant l'ETH) est un projet de rechercher à la fois ambitieux et passionant. Les deux questions suivantes pourraient être un peu plus faciles à répondre, mais elles ont aussi l'air bien difficiles:

   - On ne sait pas s'il existe une famille $\mathcal{F}$ pour laquelle $\mathcal{F}$-Deletion ne puisse pas être résolu, en supposant l'ETH, en temps $2^{\mathrm{poly}(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. Les résultats récents de Kociumaka and Pilipczuk pourraient aider à trouver la réponse, qui paraît avoir plus de chances d'être négative.
   - De l'autre côté, la cas où $\mathcal{F}$ contient un graphe planaire, c'est-à-dire, le problème Planar-$\mathcal{F}$-Deletion, est déjà extrêmement intéressant. On sait que Planar-$\mathcal{F}$-Deletion peut être résolu en temps $2^{\mathcal{O}(\mathsf{tw}\log\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, et que dans quelques cas particuliers il peut être résolu en temps $2^{\mathcal{O}(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. On croit que cette dernière possibilité arrive si et seulement si la famille $\mathcal{F}$ contient un des graphes P_2, P_3, P_4, C_3, C_4, ou $K_{1,3}$. On travaille actuellement pour essayer de démontrer ce fait.

3. Dans un de mes articles récents, on a étudié comment un paramètre structurel de graphes, la *treedepth*, peut être utilisé pour obtenir de noyaux polynomiaux dans les graphes denses (en arêtes) pour de problèmes classiques comme Vertex Cover ou Dominating Set. Deux directions de recherche naturelles apparaissent:

   - Y a-t-il de paramètres naturels qui permettent d'obtenir de résultats de *méta-kernelization* dans les graphes denses? Ici, par méta-kernelization on veut dire

un résultat qui garantit l'existence d'un type particulier de noyau pour tout problème satisfaisant certaines conditions génériques.

- Pour le cas particulier de la treedepth, et en continuant la ligne de recherche qu'on a initié dans notre article, quels sont les problèmes qui admettent de noyaux polynomiaux paramétrées par la taille d'un *modulateur de treedepth*? Un modulateur de treedepth dans un graphe est un ensemble de sommets tel que si l'on le supprime, le graphe qu'on obtient a treedepth bornée par une constante.

4. La question suivante est simple à poser, mais probablement compliquée à répondre: est-ce que le problème Minimum Colored Cut est NP-complet? Dans ce problème, étant donné un graphe avec une coloration sur les arêtes, le but est de trouver une coupe d'arêtes qui utilise le nombre minimum de couleurs. Sans les couleurs, le problème peut être résolu en temps polynomial en utilisant la dualité flot max-coupe min, mais sa complexité dans le cas avec couleurs reste un problème ouvert.

5. On dit qu'un graphe est (r, ℓ) s'il admet une (r, ℓ)-partition, c'est-à-dire, une partition de son ensemble de sommets en r ensembles indépendants et ℓ cliques. Un graphe est *bien-couvert* si tout ensemble indépendant maximal est aussi maximum. Un graphe est (r, ℓ)-*bien-couvert* s'il est à la fois (r, ℓ) et bien-couvert. On a réussi classer la complexité du problème de décider si un graphe donné avec une (r, ℓ)-partition est bien-couvert pour toutes les valeurs de r et ℓ, sauf pour $\ell = 0$ et $r \geqslant 3$. En particulier, étant donné un graphe triparti G avec une tripartition de $V(G)$, le problème de décider si G est bien-couvert est NP-complet?

6. En changeant vers de questions purement combinatories, nous avons démontré que le nombre $T_{n,k}$ de graphes étiquetés à n sommets satisfait

$$\left(c \cdot \frac{k\ 2^k n}{\log k}\right)^n 2^{-\frac{k(k+3)}{2}} k^{-2k-2} \ \leqslant\ T_{n,k}\ \leqslant\ \left(k\ 2^k n\right)^n 2^{-\frac{k(k+1)}{2}} k^{-k},$$

pour $k > 1$ et une constante explicite absolue $c > 0$. Trouver la bonne valeur de $T_{n,k}$ reste un problème ouvert très intéressant.

On croit fortement qu'il existe une constante universelle d et une fonction f, avec $k^{-2k-2} \leqslant f(k) \leqslant k^{-k}$ pour tout entier positif k, tels que

$$T_{n,k}\ \geqslant\ (d \cdot k \cdot 2^k \cdot n)^n \cdot 2^{-\frac{k(k+1)}{2}} \cdot f(k).$$

7. Finalement, sur la propriété d'Erdős-Pósa pour les mineurs de graphes, on sait que quand H est un graphe planaire, le *gap* f_H de la propriété d'Erdős-Pósa pour les H-mineurs satisfait $f_H(k) = \mathcal{O}(k \cdot \log^{\mathcal{O}(1)} k)$ et que, si H contient un cycle, alors $f_H(k) = \Omega(k \cdot \log k)$. On a prouvé récemment que $f_H(k) = \Theta(k \cdot \log k)$ quand H est une roue, et on conjecture que cette fonction est le bon *gap* pour tout graphe planaire H. De nouvelles techniques et idées ont l'air d'être nécessaires pour résoudre cette conjecture.

Chapter 1

Introduction

This document contains an overview of the research activity that I have conducted in the last years, after I defended my Ph.D in October 2009, and mostly during the time that I have spent at the LIRMM (Montpellier, France) as a *Chargé de Recherche* of the CNRS.

Rather than an exhaustive summary of my articles, this document contains mainly two parts, aside from my curriculum vitae, the scientific preliminaries, and some questions for further research. Namely, on the one hand, we first provide a list of "stuffed" abstracts of my contributions, in order to provide a global picture of them. On the other hand, the document contains four chapters corresponding to four selected articles that, in my opinion, provide a good flavor of the topics I work on, with all the scientific details.

All my research has been carried out with the invaluable help of my collaborators, which I will not list here. Without them, I would probably not find a sense to my everyday activity. Thanks a lot to all of you!!!

The remainder of this introduction is devoted to a brief contextualization of my research, a description of several recent and significant scientific collaborations, and a more detailed explanation about how this manuscript is organized.

Contents

1.1 Contextualization

I will start by briefly describing my current research field and how it has evolved since I started my Ph.D in 2006. The title of my Ph.D was "*Optimization in Graphs under Degree Constraints Application to Telecommunication Networks*" (the full manuscript can be found at `www.lirmm.fr/~sau/PhD_Ignasi.pdf`).

During my first two years of research I devoted most of the time to a problem called *traffic grooming* that, loosely speaking, consists in partitioning the edges of a request graph under some constraints while optimizing the total number of vertices (or edges) in the decomposition. This problem has a number of different variants, and arises from optical networks, where traffic grooming has been used extensively to reduce the equipment cost of telecommunication networks.

The techniques and problems that I faced while working on traffic grooming attracted my interest, and I started to work also on *complexity* and *approximation* of problems arising purely from graph theory, mostly concerning optimization problems under degree constraints. In particular, one of the aspects that I started to study about these problems is their *parameterized complexity*, which has steadily gained important among my research interests until becoming nowadays my main area of expertise. In particular, I am interested in two of the main subareas of parameterized complexity: FPT *algorithms* and *kernelization*.

It is worth saying that some of my recent contributions have a more *combinatorial* flavor, in the sense that the results do not necessarily lead to algorithms or to complexity results. Finally, while most of my research focuses on purely graph-theoretic problems, in the last years I have also applied my expertise to problems arising from several *applications*, such as communication networks or bioinformatics.

The field of parameterized complexity originated in the 90's after the seminal work of Downey and Fellows (cf. for instance the foundational articles [113, 114]), and it has become nowadays one of the most active fields in theoretical computer science. While an introduction to parameterized complexity can be found in Section 3.2, we will provide here just a succinct introduction to the field in order to provide a closer zoom to the areas where most of my articles contribute.

Namely, parameterized complexity deals with algorithms for decision problems whose instances consist of a pair (x, k), where k is a secondary measurement of the input known as the *parameter*. A major goal in parameterized complexity is to investigate whether a problem with parameter k admits an algorithm with running time $f(k) \cdot |x|^{\mathcal{O}(1)}$, where f is a function depending only on the parameter and $|x|$ represents the input size. Parameterized problems that admit such algorithms are called *fixed-parameter tractable*, and the class of all such problems is denoted FPT.

During the last decades, parameterized complexity theory has brought forth several algorithmic meta-theorems that imply that a wide range of problems are in FPT. For instance, Courcelle's theorem [90] states that every decision problem expressible in Monadic Second Order Logic can be solved in linear time when parameterized by the treewidth of the input graph. At the price of generality, such algorithmic meta-theorems may suffer from the fact that the function $f(k)$ is huge [144, 189] or non-explicit [90, 226]. Therefore, it has become a central task in parameterized complexity to provide FPT algorithms such that the behavior of the function $f(k)$ is *reasonable*; in other words, a function $f(k)$ that could lead to a practical algorithm. Most of my contributions to the area of FPT algorithms fit into this programme: even if one knows that a problem is FPT, which is the *fastest* FPT algorithm one could hope for, subject to reasonable complexity assumptions?

A concept closely related to FPT algorithms is that of *kernelization*. A kernelization algorithm, or just *kernel*, for a parameterized problem Π takes an instance (x, k) of the problem and, in time polynomial in $|x| + k$, outputs an instance (x', k') such that $|x'|, k' \leqslant g(k)$ for some function g, and $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$. The function g is called the *size* of the kernel and may be viewed as a measure of the "compressibility" of a problem using polynomial-time preprocessing rules. A kernel is called *polynomial* (resp.

linear) if the function $g(k)$ is a polynomial (resp. linear) function in k. As mentioned by Fellows in [123], it is often (inaccurately) attributed to "folklore" that a decidable problem is in FPT if and only if it has a kernelization algorithm; this foundational observation appeared first in [76, 116]. However, the kernel that one obtains in this way is typically of size at least exponential in the parameter. A natural problem in this context is to find polynomial or linear kernels for problems that are in FPT, and most of my contributions to the area of kernelization are along this line of research.

1.2 Scientific collaborations

Most of the research I have carried out since 2010 has been in collaboration with my colleagues in the AlGCo team in Montpellier, specially with Dimitrios M. Thilikos and Christophe Paul, with whom I have co-authored a number of articles.

Beside my group in Montpellier, I collaborate regularly with researchers from other groups, both in Europe and in other continents. The following are my most active collaborations during the last years, which have given me the opportunity to start working on many interesting new problems:

- Since 2014, I have very active collaborations with two research groups in **Brazil**:

  1. **Fortaleza**: *Júlio Araujo* and *Ana Karolinna Maia*, from the *Departamento de Matemática e Computaçao da Universidade Federal do Ceará (UFC)* in Fortaleza, did their Ph.D at the same place where I did mine: the MASCOTTE project of INRIA Sophia Antipolis. Since we knew each other quite well, they invited me in Fortaleza in 2015 to collaborate with their research group. This collaboration was very successful, and this is the main reason that motivated me to spend the academic year 2016/2017 as *Professor Visitante* in Fortaleza. This year has been an amazing experience for me, both from a personal and a professional point of view. Concerning the latter, this position has given me the opportunity to teach regularly during a whole academic year, and to start several new projects with my colleagues in Fortaleza that have substantially broadened the topics I usually work on. In particular, we have been working on several problems concerning *graph coloring*. In the next years, we plan to continue the collaboration with the members of the ParGO research group in Fortaleza, in particular *Júlio Araujo*, *Ana Karolinna Maia*, *Ana Shirley*, *Victor A. Campos*, *Fabrício Benevides*, and *Cláudia Linhares-Sales*.

  2. **Rio de Janeiro**: Funded by the COFECUB project MA 622-08, entitled "Connexité et séparateurs", I visited in December 2014 the *Departamento de Ciência da Computaçao da Universidade Federal do Rio de Janeiro* (UFRJ), to work with *Sulamita Klein* and *Luerbio Faria*. This collaboration turned out to be very productive, and we have currently a number of joint articles. I visited them again in 2016 and 2017, and Luerbio came also once to visit me in Montpellier. During my trips to Rio de Janeiro, I also started a fruitful collaboration with *Uéverton Souza* from *Universidade Federal Fluminense*.

- Since I did a postdoc in the Technion (Haifa, **Israel**) in 2009-2010, I collaborate regularly with *Mordechai Shalom*. Given that Mordechai is Turkish, our friendship was at the origin of the joint CNRS-TUBITAK project (grant 114E731) between France and **Turkey** for the period 2015-2016, of which I have been the coordinator of the French side. The Turkish colleagues of the project are *Didem Gözüpek* and *Sibel Özkan* from the Computer Engineering Department of Gebze Technical University, Istanbul. The visits funded by this project have already lead to two publications.

- *Marc Noy*, from the Department of Applied Mathematics of Universitat Politècnica de Catalunya (UPC), Barcelona, **Catalonia**. I did my whole undergraduate studies and my Ph.D (in joint supervision) in the UPC, so for me it is something like my "home university". I keep in touch regularly with the group working on Graph Theory and Combinatorics. In particular, I have organized twice the JCALM (Journées Combinatoire et Algorithmes du Littoral Méditerranéen) in Barcelona, in June 2011 and October 2013. Also, I give one talk per year (in average) in the seminar of their group (see `www.lirmm.fr/~sau/Talks.html`), and I visit them quite often. Concerning research, we have recently conducted a research project with Marc Noy (UPC Barcelona) and my Ph.D student Julien Baste, which is discussed in detail in Chapter 7.

- *Gwenaël Joret* from the Département d'Informatique of Université Libre de Bruxelles, **Belgium**. Some years ago, Gwenaël visited our group in Montpellier to give a talk, and since them we collaborate regularly, as our research interests are quite close and we enjoy working together. We have visited each other several times in the last years, and our collaboration has already resulted in two joint publications.

1.3 Organization of the manuscript

The remainder of this manuscript is organized as follows:

- Chapter 2 contains my curriculum vitae, including a full list of publications, supervised students, participation into projects, teaching activity, and research visits.
- In Chapter 3 we provide the required preliminaries needed to read the subsequent chapters.
- Chapter 4 is devoted to a summary of my contributions in the four aforementioned main axes: FPT algorithms, kernelization, combinatorics, and applications.

The next four chapters present four contributions in detail, corresponding to four significant articles providing good examples of the four main axes discussed above. Namely, Chapters 5 and 6 contain contributions to the areas to FPT algorithms and kernelization, Chapter 7 presents a combinatorial result, and Chapter 8 deals with a problem arising from applications in telecommunication networks. Let us discuss more in detail what is inside each of these four chapters:

- Chapter 5 uses the technique of so-called protrusion decompositions to provide linear kernels for a wide family of problems on graphs excluding topological minors, and the first single-exponential FPT algorithm for the Planar-$\mathcal{F}$-Deletion Problem.
- Chapter 6 presents a framework to produce constructive linear kernels with explicit constants via dynamic programming. This is the main result of the Ph.D of my student Valentin Garnero.
- Chapter 7 is devoted to counting the number of labeled graphs of bounded treewidth. The bounds presented in this chapter constitute of the main results in the Ph.D of my student Julien Baste.
- Chapter 8 presents an accurate analysis of the parameterized complexity of an optimization problem that is part of the family of reload cost problems in telecommunication networks. These results are also part of the Ph.D of my student Julien Baste.

- In Chapter 9 we present several open problems for further research. We would like to mention that this chapter contains open questions that have *not* been already mentioned at the end of Chapters 5-6-7-8.
- Finally, the manuscript ends with the bibliography that has been cited throughout the document. This bibliography distinguishes between the personal publications of the author and other articles (tagged as "General Bibliography"). The personal bibliography is classified into international journals (refs. [J1-J21]), international conferences that have not yet appeared in a journal (refs. [C22-C34]), and articles currently submitted for publication (refs. [S35-S38]).

The figures that have not been made by the author of this manuscript are tagged with the symbol '©' indicating the corresponding author.

Chapter 2

Curriculum vitae

Ignasi Sau Valls

Date of birth: May 14, 1982
Place of birth: Barcelona, Catalunya
E-mail: ignasi.sau@gmail.com
Homepage: www.lirmm.fr/∼sau

2.1 Education and positions

8/2016-7/2017 **Visiting Professor** at *Department of Mathematics* of UFC[1], Fortaleza, Brazil, within ParGO[2] team.

Since 10/2010 **Chargé de Recherche CNRS[3] (Junior Researcher)** at LIRMM[4], Montpellier, France, within AlGCo[5] team.

11/2009-7/2010 **Postdoc** at *Computer Science Department* of *Technion*, Haifa, Israel.

2/2006-10/2009 **Ph.D student**, defended on October 16, 2009.
Optimization in Graphs under Degree Constraints. Application to Telecommunication Networks.
Ph.D in Computer Science and Mathematics in **joint supervision**:

Since 10/2006: at MASCOTTE[6] team of INRIA[7]/CNRS/UNS[8], Sophia-Antipolis, France.
Under the supervision of Jean-Claude Bermond and David Coudert.
Funded by a BDI (CNRS/région PACA) grant.
Teaching assistant at the department of Computer Science at UNS.

[1] UFC: Universidade Federal do Ceará
[2] ParGO: Paralelismo, Grafos e Otimização combinatória
[3] CNRS: Centre National de la Recherche Scientifique
[4] LIRMM: Laboratoire d'Informatique, Robotique et Microélectronique de Montpellier
[5] AlGCo: Algorithmes, Graphes et Combinatoire
[6] MASCOTTE: Méthodes Algorithmiques, Simulation et Combinatoire pour l'OpTimisation des TElécommunications
[7] INRIA: Institut National de Recherche en Informatique et en Automatique
[8] UNS: Université de Nice-Sophia Antipolis

Since 2/2006: at *Departament de Matemàtica Aplicada IV* of UPC[9], Barcelona, Catalunya.
Under the supervision of Xavier Muñoz.

9/2005–3/2006 **Erasmus Program** at *Université de Nice Sophia Antipolis* (Nice, France).

9/2000–7/2006 **Graduate in Telecommunications Engineering.**
ETSETB[10] and CFIS[11] of UPC, Barcelona, Catalunya. 5-years course, equivalent to M2.

9/2000–6/2005 **Graduate in Mathematics.**
FME[12] and CFIS of UPC, Barcelona, Catalunya. 5-years course, equivalent to M2.

2.2 Full list of publications

International journals

2008

[J1] I. Sau and J. Žerovnik.
An Optimal Permutation Routing Algorithm for Full-Duplex Hexagonal Mesh Networks.
*Discrete Mathematics and Theoretical Computer Science (**DMTCS**), 10(3):49-62, **2008**.*

2009

[J2] O. Amini, S. Pérennes, and I. Sau.
Hardness and Approximation of Traffic Grooming.
*Theoretical Computer Science (**TCS**), 410(38-40):3751-3760, **2009**.*

[J3] F. Huc, I. Sau, and J. Žerovnik.
(ℓ, k)-Routing on Plane Grids.
*Journal of Interconnection Networks (**JOIN**), 10(1-2):27-57, **2009**.*

[J4] G. B. Mertzios, I. Sau, and S. Zaks.
A New Intersection Model and Improved Algorithms for Tolerance Graphs.
*SIAM Journal on Discrete Mathematics (**SIDMA**), 23(4):1800-1813, **2009**.*

[9] UPC: Universitat Politècnica de Catalunya (Polytechnical University of Catalonia)
[10] ETSETB: Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, UPC
[11] CFIS: Centre de Formació Interdisciplinària Superior de la UPC, see `www-cfis.upc.es`
[12] FME: Facultat de Matemàtiques i Estadística, UPC

2010

[J5] J.-C. Bermond, C. J. Colbourn, L. Gionfriddo, G. Quattrocchi, and I. Sau.
Drop Cost and Wavelength Optimal Two-Period Grooming with Ratio 4.
*SIAM Journal on Discrete Mathematics (**SIDMA**), 24(2): 400-419, **2010**.*

[J6] I. Sau and D. M. Thilikos.
Subexponential Parameterized Algorithms for Degree-constrained Subgraph Problems on Planar Graphs.
*Journal of Discrete Algorithms (**JDA**), 8(3): 330-338, **2010**.*

[J7] D. Coudert, F. Giroire, and I. Sau.
Circuits in Graphs Through a Prescribed Set of Ordered Vertices.
*Journal of Interconnection Networks (**JOIN**), 11(3-4): 121-141, **2010**.*

2011

[J8] J.-C. Bermond, X. Muñoz, and I. Sau.
Traffic Grooming in Bidirectional WDM Ring Networks.
***Networks**, 58(1): 20-35, **2011**.*

[J9] G. B. Mertzios, I. Sau, and S. Zaks.
The Recognition of Tolerance and Bounded Tolerance Graphs.
*SIAM Journal on Computing (**SICOMP**), 40(5): 1234-1257, **2011**.*

[J10] I. Sau and D .M. Thilikos.
On Self-duality of Branchwidth on Graphs of Bounded Genus.
*Discrete Applied Mathematics (**DAM**), 159(17): 2184-2186, **2011**.*

[J11] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos.
Faster Parameterized Algorithms for Minor Containment.
*Theoretical Computer Science (**TCS**), 412(50): 7018-7028, **2011**.*

[J12] X. Muñoz, Z. Li, and I. Sau.
Edge-partitioning Regular Graphs for Ring Traffic Grooming with a Priori Placement of the ADMs.
*SIAM Journal on Discrete Mathematics (**SIDMA**), 25(4): 1490-1505, **2011**.*

2012

[J13] J.-C. Bermond, D. Coudert, J. Moulierac, S. Pérennes, I. Sau, and F. Solano Donado.
GMPLS Label Space Minimization through Hypergraph Layouts.
*Theoretical Computer Science (**TCS**), 444:3-16, **2012**.*

[J14] I. Sau, P. Šparl, and J. Žerovnik.
Simpler Multicoloring of Triangle-free Hexagonal Graphs.
*Discrete Mathematics (**DM**), 312(1): 181-187, **2012**.*

[J15] O. Amini, I. Sau, and S. Saurabh.
Parameterized Complexity of Finding Small Degree-constrained Subgraphs.
*Journal of Discrete Algorithms (**JDA**), 10: 70-83, **2012**.*

[J16] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos.
Fast Minor Testing in Planar Graphs.
***Algorithmica**, 64(1): 69-84, **2012**.*

[J17] G. B. Mertzios, I. Sau, M. Shalom, and S. Zaks.
Placing Regenerators in Optical Networks to Satisfy Multiple Sets of Requests.
*IEEE/ACM Transactions on Networking (**ToN**), 20(6): 1870-1879, **2012**.*

[J18] O. Amini, D. Peleg, S. Pérennes, I. Sau, and S. Saurabh.
On the approximability of some degree-constrained subgraph problems.
*Discrete Applied Mathematics (**DAM**), 160(12): 1661-1679, **2012**.*

2013

[J19] J. Rué, I. Sau, and D. M. Thilikos.
Asymptotic Enumeration of Non-crossing Partitions on Surfaces.
*Discrete Mathematics (**DM**), 313(5-6): 635-649, **2013**.*

[J20] D. Peleg, I. Sau, and M. Shalom.
On approximating the d-girth of a graph.
*Discrete Applied Mathematics (**DAM**), 161(16-17): 2587-2596, **2013**.*

2014

[J21] N. Bousquet, D. Gonçalves, G. B. Mertzios, C. Paul, I. Sau, and S. Thomassé.
Parameterized Domination in Circle Graphs.
*Theory of Computing Systems (**TOCS**), 54(1): 45-72, **2014**.*

[J22] J. Rué, I. Sau, and D. M. Thilikos.
Dynamic Programming for Graphs on Surfaces.
*ACM Transactions on Algorithms (**TALG**), 10(2): 8, **2014**.*

[J23] G. Joret, C. Paul, I. Sau, S. Saurabh, and S. Thomassé.
Hitting and Harvesting Pumpkins.
*SIAM Journal on Discrete Mathematics (**SIDMA**), 28(3): 1363-1390, **2014**.*

2015

[J24] J. Baste and I. Sau.
The role of planarity in connectivity problems parameterized by treewidth.
*Theoretical Computer Science (**TCS**), 570: 1-14, **2015**.*

[J25] V. Garnero, C. Paul, I. Sau, and D.M. Thilikos.
Explicit linear kernels via dynamic programming.
*SIAM Journal on Discrete Mathematics (**SIDMA**), 29(4): 1864-1894, **2015**.*

2016

[J26] E. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, and S. Sikdar.
Linear kernels and single-exponential algorithms via protrusion decompositions.
*ACM Transactions on Algorithms (**TALG**), 12(2): 21, **2016**.*

[J27] J.-F. Raymond, I. Sau, and D. M. Thilikos.
An edge variant of the Erdős-Pósa property.
*Discrete Mathematics (**DM**), 339(8): 2027-2035, **2016**.*

2017

[J28] E. Kim, C. Paul, I. Sau, and D. M. Thilikos.
Parameterized Algorithms for Min-Max Multiway Cut and List Digraph Homomorphism.
*Journal of Computer and System Sciences (**JCSS**), 86: 191-206, **2017**.*

[J29] L. P. Montejano and I. Sau.
On the complexity of computing the k-restricted edge-connectivity of a graph.
*Theoretical Computer Science (**TCS**), 662: 31-39, **2017**.*

[J30] J. Baste, F. Beggas , H. Kheddouci, and I. Sau.
On the Parameterized Complexity of the Edge Monitoring Problem.
*Information Processing Letters (**IPL**), 121: 39-44, **2017**.*

[J31] V. Garnero, I. Sau, and D.M. Thilikos.
A linear kernel for planar red-blue dominating set.
*Discrete Applied Mathematics (**DAM**), 217: 536-547, **2017**.*

[J32] J. Baste, C. Paul, I. Sau, and C. Scornavacca.
Efficient FPT algorithms for (strict) compatibility of unrooted phylogenetic trees.
*Bulletin of Mathematical Biology (**BMAB**), 79(4): 920-938, **2017**.*

[J33] L. Faria, S. Klein, I. Sau, and R. Sucupira.
Improved Kernels for Signed Max Cut ATLB on (r, ℓ)-graphs.
*Discrete Mathematics & Theoretical Computer Science (**DMTCS**), 19(1), **2017**.*

[J34] D. Chatzidimitriou, J.-F. Raymond, I. Sau, and D. M. Thilikos.
Minors in graphs of large θ_r-girth.
*European Journal of Combinatorics (**EJC**), 65:106-121, **2017**.*

[J35] D. Gözüpek, S. Özkan, C. Paul, I. Sau, and M. Shalom.
Parameterized complexity of the MINCCA problem on graphs of bounded decomposability.
*Theoretical Computer Science (**TCS**), 690: 91-103, **2017**.*

[J36] J. Baste, L. Faria, S. Klein, and I. Sau.
Parameterized Complexity Dichotomy for (r, ℓ)-Vertex Deletion.
*Theory of Computing Systems (**TOCS**), 61(3): 777-794, **2017**.*

[J37] N. Cohen, D. Gonçalves, E. Kim, C. Paul, I. Sau, D. M. Thilikos, and M. Weller.
A Polynomial-time Algorithm for Outerplanar Diameter Improvement.
*Journal of Computer and System Sciences (**JCSS**), 89: 315-327, **2017**.*

2018

[J38] S. Oum, E. Kim, C. Paul, I. Sau, and D. M. Thilikos.
An FPT 2-Approximation for Tree-Cut Decomposition.
***Algorithmica**, 80(1): 116-135, **2018**.*

[J39] H. Perret du Cray and I. Sau.
Improved FPT algorithms for weighted independent set in bull-free graphs.
*Discrete Mathematics (**DM**), 341(2): 451-462, **2018**.*

[J40] D. Chatzidimitriou, J.-F. Raymond, I. Sau, and D. M. Thilikos.
An $O(\log \mathrm{OPT})$-approximation for covering and packing minor models of θ_r.
***Algorithmica**, 80(4): 1330-1356, **2018**.*

[J41] J. Baste, M. Noy, and I. Sau.
On the number of labeled graphs of bounded treewidth.
*European Journal of Combinatorics (**EJC**), 71: 12-21, **2018**.*

To appear

[J42] J. Araújo, J. Baste, and I. Sau.
Ruling out FPT algorithms for Weighted Coloring on forests.
To appear in *Theoretical Computer Science (**TCS**).*

International conferences

[C1] J.-C. Bermond, D. Coudert, X. Muñoz, and I. Sau.
Traffic Grooming in Bidirectional WDM Ring Networks.
*In Proc. of the IEEE-LEOS International Conference on Transparent Optical Networks (**ICTON**), volume 3, pages 19-22, Nottingham, UK, June **2006**.*

[C2] I. Sau and J. Žerovnik.
Optimal Permutation Routing on Mesh Networks.
*In Proc. of the International Network Optimization Conference (**INOC**), Spa, Belgium, April **2007**.*

[C3] O. Amini, S. Pérennes, and I. Sau.
Hardness and Approximation of Traffic Grooming.
*In Proc. of the 18th International Symposium on Algorithms and Computation (**ISAAC**), volume 4835 of LNCS, pages 561-573, Sendai, Japan, December **2007**.*

[C4] O. Amini, I. Sau, and S. Saurabh.
Parameterized Complexity of the Smallest Degree-Constrained Subgraph Problem.
*In Proc. of the International Workshop on Parameterized and Exact Computation (**IWPEC**), volume 5008 of LNCS, pages 13-29, Victoria, Canada, May **2008**.*

[C5] X. Muñoz and I. Sau.
Traffic Grooming in Unidirectional WDM Rings with Bounded-Degree Request Graph.
*In Proc. of the 34th International Workshop on Graph-Theoretic Concepts in Computer Science (**WG**), volume 5344 of LNCS, pages 300-311, Durham University, U.K., July **2008**.*

[C6] O. Amini, D. Peleg, S. Pérennes, I. Sau, and S. Saurabh.
Degree-Constrained Subgraph Problems: Hardness and Approximation Results.
*In Proc. of the 6th Workshop on Approximation and Online Algorithms (**WAOA**), volume 5426 of LNCS, pages 29-42, Universität Karlsruhe, Germany, September **2008**.*

[C7] I. Sau and D. M. Thilikos.
Subexponential Parameterized Algorithms for Bounded-Degree Connected Subgraph Problems on Planar Graphs.
*In Proc. of DIMAP workshop on Algorithmic Graph Theory (**AGT**), Electronic Notes in Discrete Mathematics, 32:59-66, University of Warwick, U.K., March **2009**.*

[C8] J.-C. Bermond, D. Coudert, J. Moulierac, S. Pérennes, H. Rivano, I. Sau, and F. Solano.
MPLS label stacking on the line network.
*In Proc. of **IFIP Networking**, volume 5550 of LNCS, pages 809-820, Aachen, Germany, May **2009**.*

[C9] J.-C. Bermond, D. Coudert, J. Moulierac, S. Pérennes, I. Sau, and F. Solano.
Designing Hypergraph Layouts to GMPLS Routing Strategies.
*In Proc. of the 16th International Colloquium on Structural Information and Communication Complexity (**SIROCCO**), volume 5869 of LNCS, pages 57-71, Piran, Slovenia, May **2009**.*

[C10] I. Sau and D. M. Thilikos.
On Self-duality of Branchwidth in Graphs of Bounded Genus.
*In Proc. of the 8th Cologne Twente Workshop on Graphs and Combinatorial Optimization (**CTW**), pages 12-22. Paris, France, June **2009**.*

[C11] G. B. Mertzios, I. Sau, and S. Zaks.
A New Intersection Model and Improved Algorithms for Tolerance Graphs.
*In Proc. of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (**WG**), volume 5911 of LNCS, pages 285-295, Montpellier, France, June **2009**.*

[C12] Z. Li and I. Sau.
Graph Partitioning and Traffic Grooming with Bounded Degree Request Graph.
*In Proc. of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (**WG**), volume 5911 of LNCS, pages 250-261, Montpellier, France, June **2009**.*
Best student paper award.

[C13] D. Coudert, F. Giroire, and I. Sau.
Edge-Simple Circuits Through 10 Ordered Vertices in Square Grids.
*In Proc. of the 20th International Workshop on Combinatorial Algorithms (**IWOCA**), volume 5874 of LNCS, pages 134-145, Opava, Czech Republic, June **2009**.*

[C14] G. B. Mertzios, I. Sau, and S. Zaks.
The Recognition of Tolerance and Bounded Tolerance Graphs.
*In Proc. of the 27th International Symposium on Theoretical Aspects of Computer Science (**STACS**), volume 5 of LIPIcs, pages 858-596, Nancy, France, March **2010**.*

[C15] I. Sau, M. Shalom, and S. Zaks.
Traffic Grooming in Star Networks via Matching Techniques.
*In Proc. of the 17th International Colloquium on Structural Information and Communication Complexity (**SIROCCO**), volume 6058 of LNCS, pages 41-56, Sirince, Turkey, June **2010**.*

[C16] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos.
Faster Parameterized Algorithms for Minor Containment.
*In Proc. of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (**SWAT**), volume 6139 of LNCS, pages 322-333, Bergen, Norway, June **2010**.*

[C17] G. B. Mertzios, I. Sau, M. Shalom, and S. Zaks.
Placing Regenerators in Optical Networks to Satisfy Multiple Sets of Requests.
*In Proc. of the 37th International Colloquium on Automata, Languages and Programming (**ICALP**), volume 6199 of LNCS, pages 333-344, Bordeaux, France, July **2010**.*
Best paper award of Track C.

[C18] J. Rué, I. Sau, and D.M. Thilikos.
Dynamic Programming for Graphs on Surfaces.

*In Proc. of the 37th International Colloquium on Automata, Languages and Programming (**ICALP**), volume 6198 of LNCS, pages 372-383, Bordeaux, France, July **2010**.*

[C19] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos.
Fast Minor Testing in Planar Graphs.
*In Proc. of the 18th Annual European Symposium on Algorithms (**ESA**), volume 6346 of LNCS, pages 97-109, Liverpool, U.K., September **2010**.*

[C20] D. Peleg, I. Sau, and M. Shalom.
On approximating the d-girth of a graph.
*In Proc. of the 37th International Conference on Current Trends in Theory and Practice of Computer Science (**SOFSEM**), volume 6453 of LNCS, pages 467-481, Novy Smokovec, Slovakia, January **2011**.*

[C21] G. Joret, C. Paul, I. Sau, S. Saurabh, and S. Thomassé.
Hitting and Harvesting Pumpkins.
*In Proc. of the 19th Annual European Symposium on Algorithms (**ESA**), volume 6942 of LNCS, pages 394-407, Saarbrücken, Germany, September **2011**.*

[C22] J. Rué, I. Sau, and D. M. Thilikos.
Dynamic Programming for H-minor-free Graphs (Extended Abstract).
*In Proc. of the 18th International Conference on Computing and Combinatorics (**COCOON**), volume 7434 of LNCS, pages 86-97, Sydney, Australia, August **2012**.*

[C23] N. Bousquet, D. Gonçalves, G. B. Mertzios, C. Paul, I. Sau, and S. Thomassé.
Parameterized Domination in Circle Graphs.
*In Proc. of the 38th International Workshop on Graph-Theoretic Concepts in Computer Science (**WG**), volume 7551 of LNCS, pages 308-319, Jerusalem, Israel, June **2012**.*

[C24] V. Garnero, I. Sau, and D. M. Thilikos.
A linear kernel for planar red-blue dominating set.
*In Proc. of the 12th Cologne Twente Workshop on Graphs and Combinatorial Optimization (**CTW**), pages 117-120, Enschede, The Netherlands, May **2013**.*

[C25] E. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, and S. Sikdar.
Linear kernels and single-exponential algorithms via protrusion decompositions.
*In Proc. of the 40th International Colloquium on Automata, Languages and Programming (**ICALP**), volume 7965 of LNCS, pages 613-624, Riga, Latvia, July **2013**.*

[C26] V. Garnero, C. Paul, I. Sau, and D. M. Thilikos.
Explicit linear kernels via dynamic programming.
*In Proc. of the 31st Symposium on Theoretical Aspects of Computer Science (**STACS**), volume 25 of LIPIcs, pages 312-324, Lyon, France, March **2014**.*

[C27] D. Chatzidimitriou, J.-F. Raymond, I. Sau, and D. M. Thilikos.
Covering and packing pumpkin models.
*In Proc. of the 9th International Colloquium on Graph Theory and Combinatorics (**ICGT**), Grenoble, France, June-July **2014**.*

[C28] J.-F. Raymond, I. Sau, and D. M. Thilikos.
An edge variant of the Erdős-Pósa property.
*In Proc. of the 9th International Colloquium on Graph Theory and Combinatorics (**ICGT**), Grenoble, France, June-July **2014**.*

[C29] H. Perret du Cray and I. Sau.
Improved FPT algorithms for weighted independent set in bull-free graphs.
*In Proc. of the 9th International Symposium on Parameterized and Exact Computation (**IPEC**), volume 8894 of LNCS, pages 282-293, Grenoble, Warsaw, Poland, September **2014**.*

[C30] J. Baste and I. Sau.
The role of planarity in connectivity problems parameterized by treewidth.
*In Proc. of the 9th International Symposium on Parameterized and Exact Computation (**IPEC**), volume 8894 of LNCS, pages 63-74, Grenoble, Warsaw, Poland, September **2014**.*

[C31] L. P. Montejano and I. Sau.
On the complexity of computing the k-restricted edge-connectivity of a graph.
*In Proc. of the 41st International Workshop on Graph-Theoretic Concepts in Computer Science (**WG**), volume 9224 of LNCS, pages 219-233, Munich, Germany, June **2015**.*

[C32] N. Cohen, D. Gonçalves, E. Kim, C. Paul, I. Sau, D. M. Thilikos, and M. Weller.
A Polynomial-time Algorithm for Outerplanar Diameter Improvement.
*In Proc. of the 10th International Computer Science Symposium in Russia (**CSR**), volume 9139 of LNCS, pages 123-142, Listvyanka, Russia, July **2015**.*

[C33] S. Oum, E. Kim, C. Paul, I. Sau, and D. M. Thilikos.
An FPT 2-Approximation for Tree-Cut Decomposition. *In Proc. of the 13th Workshop on Approximation and Online Algorithms (**WAOA**), volume 9499 of LNCS, pages 35-46, Patras, Greece, September **2015**.*

[C34] D. Chatzidimitriou, J.-F. Raymond, I. Sau, and D. M. Thilikos.
An $O(\log \mathrm{OPT})$-approximation for Covering/Packing minor models of θ_r.
*In Proc. of the 13th Workshop on Approximation and Online Algorithms (**WAOA**), volume 9499 of LNCS, pages 122-132, Patras, Greece, September **2015**.*

[C35] E. Kim, C. Paul, I. Sau, and D. M. Thilikos.
Parameterized Algorithms for Min-Max Multiway Cut and List Digraph Homomorphism.
*In Proc. of the 10th International Symposium on Parameterized and Exact Computation (**IPEC**), volume 43 of LIPIcs, pages, 78-89, Patras, Greece, September **2015**.*

[C36] J. Baste, C. Paul, I. Sau, and C. Scornavacca.
Efficient FPT algorithms for (strict) compatibility of unrooted phylogenetic trees.
*In Proc. of the 11th International Conference on Algorithmic Aspects of Information and Management (**AAIM**), volume 9778 of LNCS, pages 53-64, Bergamo, Italy, July **2016**.*

[C37] D. Gözüpek, S. Özkan, C. Paul, I. Sau, and M. Shalom.
Parameterized complexity of the MINCCA problem on graphs of bounded decomposability.
*In Proc. of the 42nd International Workshop on Graph-Theoretic Concepts in Computer Science (**WG**), volume 9941 of LNCS, pages 195-206, Istanbul, Turkey, June **2016**.*

[C38] S. R. Alves, K. K. Dabrowski, L. Faria, S. Klein, I. Sau, and U. S. Souza.
On the (parameterized) complexity of recognizing well-covered (r, ℓ)-graphs.
*In Proc. of the 10th International Conference on Combinatorial Optimization and Applications (**COCOA**), volume 10043 of LNCS, pages 423-437, Hong Kong, China, December **2016**.*

[C39] J. Baste, M. Noy, and I. Sau.
On the number of labeled graphs of bounded treewidth.
*In Proc. of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science (**WG**), volume 10520 of LNCS, pages 88-99, Eindhoven, The Netherlands, June **2017**.*

[C40] J. Baste, D. Rautenbach, and I. Sau.
Uniquely restricted matchings and edge colorings.
*In Proc. of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science (**WG**), volume 10520 of LNCS, pages 100-112, Eindhoven, The Netherlands, June **2017**.*

[C41] N. Cohen, F. Havet, D. Mazauric, I. Sau, and R. Watrigant.
Complexity Dichotomies for the Minimum $\mathcal{F}$-Overlay Problem.
To appear in *Proc. of the 28th International Workshop on Combinatorial Algorithms (**IWOCA**), LNCS, Newcastle, Australia, July **2017**.*

[C42] J. Araújo, J. Baste, and I. Sau.
Ruling out FPT algorithms for Weighted Coloring on forests.
*In Proc. of the IX Latin and American Algorithms, Graphs and Optimization Symposium (**LAGOS**), volume 62 of ENDM, pages 195-200, Marseille, France, September **2017**.*

[C43] L. Faria, S. Klein, I. Sau, U. S. Souza, and R. Sucupira.
Maximum Cuts in Edge-colored Graphs.
*In Proc. of the IX Latin and American Algorithms, Graphs and Optimization Symposium (**LAGOS**), volume 62 of ENDM, pages 87-92, Marseille, France, September **2017**.*

[C44] M. Bougeret and I. Sau.
How much does a treedepth modulator help to obtain polynomial kernels beyond sparse graphs?
*In Proc. of the 12th International Symposium on Parameterized and Exact Computation (**IPEC**), volume 89 of LIPIcs, pages 10:1–10:13, Vienna, Austria, September **2017**.*

[C45] J. Baste, D. Gözüpek, C. Paul, I. Sau, M. Shalom, and D. M. Thilikos.
Parameterized complexity of finding a spanning tree with minimum reload cost diameter.
*In Proc. of the 12th International Symposium on Parameterized and Exact Computation (**IPEC**), volume 89 of LIPIcs, pages 3:1–3:12, Vienna, Austria, September **2017**.*

[C46] J. Baste, I. Sau and D. M. Thilikos.
Optimal algorithms for hitting (topological) minors on graphs of bounded treewidth.
*In Proc. of the 12th International Symposium on Parameterized and Exact Computation (**IPEC**), volume 89 of LIPIcs, pages 4:1–4:12, Vienna, Austria, September **2017**.*

[C47] J. Araújo, V. A. Campos, A. K. Maia, I. Sau, and A. Silva.
On the complexity of finding internally vertex-disjoint long directed paths.
To appear in *Proc. of the 13th Latin American Theoretical INformatics Symposium (**LATIN**), LNCS, Buenos Aires, Argentina, April **2018**.*

Book chapters

[B1] T. Cinkler, D. Coudert, M. Flammini, G. Monaco, L. Moscardelli, X. Muñoz, I. Sau, M. Shalom, and S. Zaks. Graphs and Algorithms in Communication Networks - Studies in Broadband, Optical, Wireless and Ad Hoc Networks. *Chapter*: Traffic Grooming: Combinatorial Results and Practical Resolutions. ***EATCS Texts in Theoretical Computer Science, Springer**. December 2009.*

[B2] I. Sau and J. Žerovnik. Graphs and Algorithms in Communication Networks - Studies in Broadband, Optical, Wireless and Ad Hoc Networks. *Chapter*: Permutation Routing and (ℓ, k)-routing on Plane Grids. ***EATCS Texts in Theoretical Computer Science, Springer**. December 2009.*

National conferences

[N1] O. Amini, S. Pérennes, and I. Sau. Hardness of Approximating the Traffic Grooming. *In Proc. of 9es Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (**AlgoTel**), Ile d'Oléron, France, May 2007.*

[N2] J. Araújo, C. Linhares-Sales, and I. Sau. Weighted Coloring on P_4-sparse Graphs. *In Proc. of 11es Journées Doctorales en Informatique et Réseaux (**JDIR**), Sophia Antipolis, France, March 2010.*

[N3] L. Faria, S. Klein, I. Sau, U. S. Souza, and R. Sucupira. Finding cuts in edge-colored graphs. *In XXXVI Congresso da Sociedade Brasileira de Computação - Encontro de Teoria da Computação (**CSBC-ETC**), Porto Alegre, Brazil, July 2016*, and *XLVIII Simpósio Brasileiro de Pesquisa Operacional (**SBPO**), Vitória, Brazil, September 2016.*

Submitted to journal

[S1] V. Garnero and I. Sau.
A linear kernel for planar total dominating set.
Submitted to *Discrete Mathematics & Theoretical Computer Science.*

[S2] V. Garnero, C. Paul, I. Sau, and D. M. Thilikos.
Explicit Linear Kernels for Packing Problems.
Submitted to *Algorithmica.*

[S3] J. Baste, D. Rautenbach, and I. Sau.
Upper Bounds on the Uniquely Restricted Chromatic Index.
Submitted to *Journal of Graph Theory.*

[S4] J. Baste, D. Rautenbach, and I. Sau.
Approximating Maximum Uniquely Restricted Matchings in Bipartite Graphs.
Submitted to *Discrete Applied Mathematics.*

[S5] S. R. Alves, K. K. Dabrowski, L. Faria, S. Klein, I. Sau, and U. S. Souza.
On the (parameterized) complexity of recognizing well-covered (r, ℓ)-graphs.
Submitted to *Theoretical Computer Science.*

[S6] J. Baste, D. Gözüpek, C. Paul, I. Sau, M. Shalom, and D. M. Thilikos.
Parameterized complexity of finding a spanning tree with minimum reload cost diameter.
Submitted to *Journal of Computer and System Sciences.*

[S7] M. Bougeret and I. Sau.
How much does a treedepth modulator help to obtain polynomial kernels beyond sparse graphs?
Submitted to *Algorithmica.*

[S8] J.-F. Raymond, P. Aboulker, S. Fiorini, T. Huynh, G. Joret, and Ignasi Sau.
A tight Erdős-Pósa function for wheel minors.
Submitted to *SIAM Journal on Discrete Mathematics.*

[S9] J. Araújo, V. A. Campos, A. K. Maia, I. Sau, and A. Silva.
On the complexity of finding internally vertex-disjoint long directed paths.
Submitted to *Journal of Computer and System Sciences*.

[S10] N. Cohen, F. Havet, D. Mazauric, I. Sau, and R. Watrigant.
Complexity Dichotomies for the Minimum $\mathcal{F}$-Overlay Problem.
Submitted to *Journal of Discrete Algorithms*.

2.3 Supervised students

03/2012-08/2012 Valentin Garnero.
Internship M2, ENS–Université de Nice Sophia-Antipolis.
Topic: Polynomial kernels for variants of domination problems on planar graphs.
Percentage of supervision: 100%.
Joint publications: [S1].

02/2013-07/2013 Julien Baste.
Internship M2, ENS Cachan.
Topic: The role of planarity in connectivity problems parameterized by treewidth.
Percentage of supervision: 100%.
Joint publications: [J24, C30].

02/2014-07/2014 Henri Perret du Cray.
Internship M2, Université de Montpellier.
Topic: FPT algorithms and kernels on graphs without induced subgraphs.
Percentage of supervision: 100%.
Joint publications: [J40, C29].
Current position: Ph.D student at LIMOS laboratory, Clermont-Ferrand, France.

10/2012-07/2016 Valentin Garnero.
Ph.D, co-supervised with Christophe Paul.
Topic: (Méta)-noyaux constructifs et linéaires dans les graphes peu denses.
Percentage of supervision: 60%.
Joint publications: [J25, J31, C24, C26, S2].
Current position: preparing the CAPES in Mathematics, to teach in a French high school.

09/2014-09/2017 Julien Baste.
Ph.D, co-supervised with Dimitrios M. Thilikos.
Topic: Treewidth: algorithmic, combinatorial and practical aspects.
Percentage of supervision: 70%.
Joint publications: [J30, J32, J36, C26, C29, C40, C42, C45, C46, S3, S4, S5, S6, S8].

Current position: postdoc (ATER) at Université Pierre et Marie Curie, Paris, France.

2.4 Awards, grants, scholarships, and projects

2000 Got accepted to a special plan for high performance students of Universitat Politècnica de Catalunya (10 students were selected all over Spain), to study in parallel the degrees of Mathematics and Telecommunications Engineering.

2000–2001 First year of university funded by the Spanish government for having the highest grades in my secondary school.

2-7/2005 `Research Support Scholarship` at MA4[13] of UPC (Barcelona, Catalunya). Grant of *Ministerio de Ciencia e Innovación* of Spain. Topic: *Graph Theory and Optical Networks*, under the direction of Xavier Muñoz.

2005–2006 Recipient of an `Erasmus` European grant to study in Nice, France.

2006–2009 Recipient of a French Ph.D grant `BDI` (joint of CNRS and région PACA).

2006–2009 Recipient of French grant `RECHERCHE 06` to promote student's mobility.

2006–2008 Recipient of 3 grants of European Project `COST 293` to perform Short Term Scientific Missions in Slovenia (September 2006) and Israel (June 2007 and May 2008).

6/2009 Recipient of the Best Student Paper Award of WG'09 for publication [C12].

11/2009 Recipient of a postdoctoral grant in the Computer Science Department of the Technion (Haifa, Israel).

7/2010 Recipient of the Best Paper Award of Track C of ICALP'10 for publication [C17].

2012-2015 Recipient of the Prime d'Excellence Scientifique (PES) of the CNRS.

2015-2016 Coordinator of an awarded bilateral research project funded by CNRS and TUBITAK between France and Turkey, entitled "*Reload Cost Concept in Graph Theory: A Combinatorial Analysis*". Grant 114E731. Funded 2.000€ per year to cover travel expenses.

2016 Recipient of 10.000€ from the Université de Montpellier (*Soutien à la Recherche 2016*) for the personal project "*Parameterized Algorithms for Structured grAphs (PASTA)*".

2016-2017 Awarded a Visiting Professor position at Department of Mathematics of Universidade Federal do Ceará (Fortaleza, Brazil) for the period 8/2016-8/2017.

[13] MA4: Departament de Matemàtica Aplicada 4 de la UPC

2017 Recipient of 1.250€ from the "*Exploration Japon 2017*" program to carry out a research visit in Tokyo, Japan.

Participation in French ANR projects

2009-2013 ANR AGAPE: Algorithmes de GrAphes Paramétrés et Exacts.
Project number: ANR-09-BLAN-0159.
Budget: 600.000€.

2016-2020 ANR GATO: Graphs, Algorithms and TOpology.
Project number: ANR-16-CE40-0009-01.
Budget: 355.000€.

2017-2021 ANR DE-MO-GRAPH: Décomposition de Modèles Graphiques.
Project number: ANR-16-CE40-0028.
Budget: 360.098€.

2018-2022 ANR ESIGMA: Efficiency and Structure In Graph Mining Applications.
Project number: To be assigned soon.
Budget: 408.191,4€.

2.5 Teaching activity

2011/2012 **Algorithmes élégants**.
Module de l'École Doctorale I2S, Université de Montpellier 2 - ***12h****.*

2012/2013 **Complexité et algorithmes paramétrés**.
Cours de M2 Informatique, Université de Montpellier 2 - ***10h****.*

2013/2014 **Algorithmes élégants**.
Module de l'École Doctorale I2S, Université de Montpellier 2 - ***11h****.*

2015/2016 **Graphes, algorithmique et complexité**.
Cours de M2 Informatique, Université de Montpellier - ***9h****.*

2016/2017 **Complexidade parametrizada**.
Curso de pós-graduação, Departamento de Matemática, Universidade Federal do Ceará, Fortaleza, Brazil - ***96h****.*

During my Ph.D (2006-2009) I have been *Moniteur* (corresponding to Junior Assistant Professor) at University of Nice-Sophia Antipolis.

2006/2007 **Programmation Répartie en Java (Distributed Programming in Java)**.
IUT UNSA, Bachelor in Computer Science, 2nd year - ***64h****.*

2007/2008 **Programmation concurrente en JAVA (Concurrent programming in JAVA)**.
IUT UNSA, Bachelor in Computer Science, 2nd year - ***21h****.*
Algorithmique et complexité (Algorithms and complexity).

*IUT UNSA, Bachelor in Computer Science, 3rd year - **49h**.*
Mathématiques discrètes et optimisation (Discrete maths and optimization).
*IUT UNSA, Bachelor in Computer Science, 3rd year - **21h**.*

2008/2009 **Bases algorithmiques (Algorithmic principles)**.
*IUT UNSA, Bachelor in Computer Science, 3rd year - **37h**.*

2.6 Committees and administrative duties

Committees of conferences and workshops

- Organizing Committee of AdHocNow'08, 7th International Conference on AD-HOC Networks, Sophia Antipolis, France, September 2008 (`www-sop.inria.fr/mascotte/adhocnow`).
- Organizing Committee of JGA'08, Journées Graphes et Algorithmes, Sophia Antipolis, France, November 2008 (`www-sop.inria.fr/mascotte/seminaires/JGA08`).
- General co-Chair of IMAGINE'09, 3rd International workshop on Mobility, Algorithms, Graph theory In dynamic NEtworks, Piran, Slovenia, May 2009.
- Organizing Committee of WG'10, 36th International Workshop on Graph Theoretic Concepts in Computer Science, Zarós, Greece, June 2010 (`wg2010.thilikos.info`).
- Program Committee of SIROCCO'12 (`sites.google.com/site/sirocco2012iceland`).
- Program Committee of AlgoTel'13 (`http://algotel2013.sciencesconf.org`).
- Program Committee of AlgoTel'14 (`http://algotel2014.sciencesconf.org`).
- Program Committee of JMDA'16 (`http://llati.upc.edu/JMDA16`).
- Program Committee of WG'18.

Organization of workshops and seminars

- Organizer of the 1st JCALM (Journées Combinatoire et Algorithmes du Littoral Méditerranéen) in Barcelona, June 2011 (`www.lirmm.fr/~sau/JCALM.html`).
- Co-organizer of Journées AGAPE on Parameterized Complexity and Exact Algorithms, Montpellier, France, February 2012 (`www.lirmm.fr/~sau/AGAPE12.html`).
- Organizer of the 2nd JCALM (Journées Combinatoire et Algorithmes du Littoral Méditerranéen) in Barcelona, October 2013 (`www.lirmm.fr/~sau/JCALM2013Bcn.html`).
- Co-organizer of 7th GROW, Aussois, France October 2015, (`http://grow2015.sciencesconf.org`).

- Co-organizer of 45th WG, Vall de Núria, Catalonia, June 2019.

Participation in Ph.D defenses

- Reviewer of the Ph.D of Miguel H. Camelo, UdG, Girona, Catalunya, October 2014.
- Jury member of the Ph.D defense of Valentin Garnero, Université de Montpellier, Montpellier, France, July 2016.
- Reviewer and jury member of the Ph.D of Rubens Sucupira, UFRJ, Rio de Janeiro, Brazil, January 2017.
- Reviewer and jury member of the Ph.D of Thiago Marcilon, UFC, Fortaleza, Brazil, February 2017.
- Jury member of the Ph.D defense of Julien Baste, Université de Montpellier, Montpellier, France, September 2017.

Refereeing of international research projets

- Funding scheme PRELUDIUM, National Science Centre of Poland, March 2016.
- Vidi grant, Netherlands Organisation for Scientific Research (NWO), December 2016.

Other administrative responsibilities

- Responsible of the weekly research seminar of our group `AlGCo` since April 2013 until July 2016 (`www2.lirmm.fr/algco/GT`).
- Member of the "Commission of Section 27 (MIPS)" of Université de Montpellier since June 2017.

2.7 Research visits

9/2006 **Department of Theoretical Computer Science of IMFM, Ljubljana, Slovenia.**
Visiting Janez Žerovnik (1 month).

6/2007 **Computer Science Department of Technion, Haifa, Israel.**
Visiting Shmuel Zaks and Mordechai Shalom (1 month).

12/2007 **Department of Theoretical Computer Science of IMFM, Ljubljana, Slovenia.**
Visiting Janez Žerovnik (2 weeks).

2-3/2008 **Departamento de Computaçao da Universidade Federal do Ceará (UFC), Fortaleza, Brazil.**
Visiting Cláudia Linhares-Sales (6 weeks).

6/2008 **Computer Science Department of Technion, Haifa, Israel.**
Visiting Shmuel Zaks and Mordechai Shalom (1 month).

11/2008 **Department of Mathematics of NKU, Athens, Greece.**
Visiting Dimitrios M. Thilikos (2 weeks).

12/2008 **Department of Theoretical Computer Science of IMFM, Ljubljana, Slovenia.**
Visiting Janez Žerovnik (2 weeks).

5/2009 **Algorithms Research Group of University of Bergen, Norway.**
Visiting Isolde Adler, Frederic Dorn, and Fedor Fomin (1 week).

5/2010 **School of Computer Science of McGill University, Montréal, Canada.**
Visiting Bruce Reed (1 month).

11/2010 **Department of Mathematics of NKU, Athens, Greece.**
Visiting Dimitrios M. Thilikos (1 week).

12/2010 **Department of Computer Science of Charles University, Prague, Czech Republic.**
Visiting Daniel Král' (2 weeks).

1/2011 **Faculty of Mathematics and Computer Science of Weizmann Institute of Science, Rehovot, Israel.**
Visiting David Peleg (2 weeks).

3/2012 **Département d'Informatique of Université Libre de Bruxelles, Brussels, Belgium.**
Visiting Gwenaël Joret and Samuel Fiorini (1 week).

4/2012 **Instituto de Ciencias Matemáticas, Madrid, Spain.**
Visiting Juanjo Rué (1 week).

9/2012 **Department of Mathematics of University of Pilsen, Czech Republic.**
Visiting Daniel Král' and Thomas Kaiser (1 week).

10/2012 **Algorithms Research Group of University of Bergen, Norway.**
Visiting Pinar Heggernes and Pim van 't Hof (1 week).

12/2014 **Departamento de Ciência da Computaçao da Universidade Federal do Rio de Janeiro (UFRJ), Brazil.**
Visiting Sulamita Klein and Luerbio Faria (4 weeks).

2/2015 **Departamento de Computaçao da Universidade Federal do Ceará (UFC), Fortaleza, Brazil.**
Visiting Júlio Araújo and Karol Maia (4 weeks).

5/2015 **Departament de Matemàtica Aplicada II of Universitat Politècnica de Catalunya (UPC), Barcelona, Catalunya.**
Visiting Marc Noy (1 week).

6/2015 **Computer Engineering Department of Gebze Technical University, Istanbul, Turkey.**
Visiting Didem Gözüpek, Sibel Ökzan, and Mordechai Shalom (1 week).

1/2016 **Departamento de Ciência da Computaçao da Universidade Federal do Rio de Janeiro (UFRJ), Brazil.**
Visiting Sulamita Klein, Luerbio Faria, and Uéverton Souza (3 weeks).

6/2016 **COATI team, INRIA Sophia Antipolis, France.**
Visiting Frédéric Havet and Dorian Mazauric (1 week).

6/2016 **Computer Engineering Department of Gebze Technical University, Istanbul, Turkey.**
Visiting Didem Gözüpek and Mordechai Shalom (1 week).

2016-2017 **Departamento de Matemática da Universidade Federal do Ceará (UFC), Fortaleza, Brazil.**
Position of Visiting Professor (1 year).

1/2017 **Departamento de Ciência da Computaçao da Universidade Federal do Rio de Janeiro (UFRJ), Brazil.**
Visiting Sulamita Klein, Luerbio Faria, and Uéverton Souza (1 week).

11/2017 **Université Pierre et Marie Curie, Paris, France.**
Visiting Julien Baste (1 week).

12/2017 **University of Electro-Communications, Tokyo, Japan.**
Visiting Rémy Belmonte (2 weeks).

2.8 Research talks

6/06 **Traffic grooming in optical networks.**
Seminar of Graph Theory, Combinatorics and Applications of UPC, Barcelona, Catalunya.

6/2006 **Traffic grooming in bidirectional WDM ring networks.**
Presentation of a paper in ICTON, Nottingham, UK.

4/2007 **Optimal permutation routing on mesh networks.**
Presentation of a paper in INOC, Spa, Belgium.

5/2007 **Hardness of approximating the traffic grooming problem.**
Presentation of a paper in AlgoTel, Ile d'Oleron, France.

4/08 **Maximum Degree-Bounded Connected Subgraph: Hardness and Approximation.**
Seminar in MASCOTTE project, Sophia-Antipolis, France.

6/2008 **Maximum degree-bounded connected subgraph: hardness and approximation.**
Seminar in the Computer Science Department of Technion, Haifa, Israel.

7/2008 **Traffic grooming in unidirectional WDM rings with bounded-degree request graph.**
Presentation of a paper in WG, Durham University, U.K.

9/2008 **Degree-constrained subgraph problems: hardness and approximation**.
Presentation of a paper in ALGO/WAOA, Universität Karlsruhe, Germany.

11/2008 **Degree-constrained subgraph problems: hardness and approximation**.
Seminar in the Department of Mathematics of NKU, Athens, Greece.

12/2008 **Degree-constrained subgraph problems: hardness and approximation**.
Seminar in the Department of Mathematics of IMFM, Ljubljana, Slovenia.

3/2009 **Subexponential parameterized algorithms for bounded-degree connected subgraph problems on planar graphs**.
Presentation of a paper in DIMAP AGT, University of Warwick, U.K.

5/2009 **Designing hypergraph layouts for GMPLS routing strategies**.
Presentation of a paper in SIROCCO, Piran, Slovenia.

6/2009 **On self-duality of branchwidth in graphs of bounded genus**.
Presentation of a paper in CTW, Paris, France.

6/2009 **Graph partitioning and traffic grooming with bounded degree request graph**.
Presentation of a paper in WG, Montpellier, France.

7/09 **Edge-partitioning regular graphs**.
Seminar in MASCOTTE project, Sophia-Antipolis, France.

10/2009 **Optimization in graphs under degree constraints. Application to telecommunication networks**.
Ph.D defense, Sophia Antipolis, France.

10/2009 **Optimization in graphs under degree constraints. Application to telecommunication networks**.
Seminar of Graph Theory, Combinatorics and Applications of UPC, Barcelona, Catalunya.

3/2010 **Edge-partitioning regular graphs, with applications to traffic grooming.**
Combinatorics Seminar of the Mathematics Department of the Technion, Haifa, Israel.

6/2010 **Dynamic programming for graphs on surfaces.**
In the Research Seminar in Discrete and Computational Geometry, Ben Gurion University of the Negev, Be'er Sheva, Israel.

10/2010 **Dynamic programming for graphs on surfaces.**
In the 2nd Workshop on Graph Decompositions, CIRM, Marseille, France.

11/2010 **Dynamic programming for graphs on surfaces.**
Seminar in the Department of Mathematics of NKU, Athens, Greece.

1/2011 **On approximating the d-girth of a graph.**
Presentation of a paper in SOFSEM, Novy Smokovec, Slovakia.

2/2011 **Single-exponential parameterized algorithms: good and bad news.**
Lecture in Journées AGAPE, Montpellier, France.

3/2011 **Dynamic programming in sparse graphs.**
Seminar of Graph Theory, Combinatorics and Applications of UPC, Barcelona, Catalunya.

3/2011 **Traffic grooming in bidirectional WDM ring networks.**
Talk in the celebration of Jean-Claude Bermond's EADS 2010 prize, Montpellier, France.

9/2011 **Hitting and harvesting pumpkins**.
Presentation of a paper in ESA, Saarbrücken, Germany.

9/2011 **Parameterized domination in circle graphs**.
Talk in GROW workshop, Daejeon, South Korea.

3/2012 **Dynamic programming in sparse graphs**.
Seminar in Université Libre de Bruxelles (ULB), Brussels, Belgium.

10/2012 **Parameterized domination in circle graphs**.
Seminar in the Algorithms Research Group, University of Bergen, Norway.

1/2013 **Optimal Erdös-Pósa property for pumpkins**.
Talk in Congreso de la Real Sociedad Matemática Española, Santiago de Compostela, Spain.

1/2013 **Linear kernels on graphs excluding topological minors**.
Talk in the ANR AGAPE meeting, Université d'Orléans, France.

2/2013 **Linear kernels and single-exponential algorithms via protrusion decompositions**.
Seminar of Graph Theory, Combinatorics and Applications of UPC, Barcelona, Catalunya.

6/2013 **Optimal Erdös-Pósa property for pumpkins**.
Talk in the Joint Mathematical Conference of Catalan, Slovenian, Austrian, Slovak and Czech Mathematical Societies (CSASC), Koper, Slovenia.

10/2013 **Bidimensionality theory**.
Tutorial in the JCALM workshop, Barcelona, Catalunya.

6/2014 **Programmation dynamique dans les graphes peu denses**.
Talk in the "Journée scientifique du LIRMM" , Montarnaud, France.

10/2014 **Linear kernels and single-exponential algorithms via protrusion decompositions.**
Seminar in Université Libre de Bruxelles (ULB), Brussels, Belgium.

11/2014 **On the Erdös-Pósa property for minors of graphs.**
Talk in Barcelona Mathematical Days, Institut d'Estudis Catalans, Barcelona, Catalunya.

12/2014 **Linear kernels and single-exponential algorithms via protrusion decompositions.**
Talk in Seminário de Grafos e Algoritmos, UFRJ, Rio de Janeiro, Brazil.

1/2015 **FPT algorithm for a generalized cut problem and some applications.**
Séminaire AlGCo, Montpellier, France.

1/2015 **FPT algorithm for a generalized cut problem and some applications.**
Talk in Rencontres Internationales sur les méthodes de décomposition de graphes, CIRM, Marseille, France.

1/2015 **FPT algorithm for a generalized cut problem and some applications.**
Seminário ParGO, Universidade Federal do Ceará, Fortaleza, Brazil.

5/2015 **The List Allocation problem and some of its applications in parameterized algorithms.**
Seminar of Graph Theory, Combinatorics and Applications of UPC, Barcelona, Catalunya.

6/2015 **On the complexity of computing the k-restricted edge-connectivity of a graph.**
Séminaire AlGCo, Montpellier, France.

6/2015 **On the complexity of computing the k-restricted edge-connectivity of a graph.**
Presentation of a paper in WG, Munich, Germany.

3/2016 **Noyaux (presque) linéaires dans les graphes d'expansion bornée et "nowhere dense"**.
Talk in the JCALM workshop, Montpellier, France.

5/2016 **The number of graphs of bounded treewidth**.
Séminaire COATI, Sophia Antipolis, France.

9/2016 **The number of graphs of bounded treewidth**.
Seminário ParGO, Fortaleza, Brazil.

6/2017 **Finding subdivisions of spindles on digraphs**.
Seminário ParGO, Fortaleza, Brazil.

9/2017 **Ruling out FPT algorithms for Weighted Coloring on forests**.
Presentation of a paper in LAGOS, Marseille, France.

10/2017 **Finding subdivisions of spindles on digraphs**.
Séminaire AlGCo, Montpellier, France.

2.9 Journal and conference refereeing

This is the number of articles that I have reviewed for the following journals and conferences (if no number is shown, it corresponds to just one article):

- **Journals**: Algorithmica (x3), Information and Computation, ACM Transactions on Algorithms (x2), Discrete Applied Mathematics (x13), Theoretical Computer Science (x7), Journal of Discrete Algorithms (x2), European Journal of Operational Research, Journal of Graph Algorithms and Applications, Networks, European Journal of Combinatorics (x2), Discrete Mathematics, SIAM Journal on Discrete Mathematics (x3), Journal of Combinatorial Optimization (x2), Discussiones Mathematicae Graph Theory, Journal of Combinatorial Theory B, Information Processing Letters, Journal of Computer and System Sciences.

- **Conferences**: AlgoTel'08, WG'09, STACS'10, CIAC'10, WG'10 (x3), IWOCA'10, ISAAC'10, IPEC'10, WG'11, IWOCA'11, FCT'11, ESA'11, DISC'11, SODA'12, SWAT'12, MFCS'12, IPEC'12, CIAC'13, ICALP'13 (x2), AlgoTel'13 (x6), WG'13, EuroComb'13, MFCS'13, ESA'13, FCT'13, IPEC'13 (x2), STACS'14, IPCO'14, CSR'14 (x2), ICALP'14, WG'14, MFCS'14 (x2), STACS'15, CIAC'15, WG'15, AlgoTel'15, ESA'15, IPEC'15, LATIN'16, SWAT'16, WG'16, JMDA'16 (x3), ESA'16, IWOCA'16, IPEC'16, SODA'17, ICALP'17, WG'17 (x2), ESA'17 (x2), IPEC'17, SODA'18, STACS'18.

Chapter 3

Preliminaries

We provide here some basic preliminaries and fix the notation to be used throughout this manuscript. We focus on the most important definitions here, and defer some of them to the corresponding chapter where they are locally used. This part is intended to be looked up when necessary, rather than to be read sequentially.

We assume that the reader is familiar with the classes P and NP, as well as with the asymptotic notation (like $\mathcal{O}$, o, Ω, or Θ). For additional background material concerning computational complexity, the reader is referred to the books of Garey and Johnson [154] and Varizani [239].

In Section 3.1 we state some preliminaries about graphs, in Section 3.2 we briefly introduce the field of parameterized complexity, and in Section 3.3 we define for completeness some classical graph problems to which we will refer in the sequel.

Contents

3.1 Graphs

In this section we state some basic preliminaries about graphs. Namely, we set in Section 3.1.1 the notation that we use for the basic notions, we define in Section 3.1.2 graph minors, contractions, and topological minors, and we introduce in Section 3.1.3 tree-decompositions and treewidth. Finally, even if it is a topic that goes beyond graphs, we briefly present in Section 3.1.4 (Counting) Monadic Second Order Logic.

3.1.1 Basic notation

We use standard graph-theoretical terminology, and we assume that the reader is familiar with the basic concepts of graph theory. For more details, see for instance the book of

Diestel [106]. All the graphs considered in Chapters 5-6-7-8 are undirected, and contain no loops nor multiple edges; such graphs are called *simple*.

Given a simple undirected graph $G = (V, E)$, and edge between the vertices u and v is denoted by $\{u, v\}$, and then u and v are said to be *adjacent*. An edge is *incident* to its two endpoints. The *degree* of a vertex v in G is the number of vertices incident to v in G. Namely, $d_G(v) = |\{u \in V(G) \mid \{u, v\} \in E(G)\}|$. The *maximum degree* (resp. *minimum degree*) of a graph G is the maximum (resp. minimum) degree over all its vertices, and it is denoted by $\Delta(G)$ (resp. $\delta(G)$).

For a graph $G = (V, E)$ and a subset $V' \subseteq V$, we denote the *induced subgraph* on V' by $G[V'] = (V', E')$, where $E' = \{\{u, v\} \in E \mid u, v \in V'\}$. For $X \subseteq V(G)$, we define $G - X := G[V(G) \setminus X]$. The *distance* between two vertices u and v in a graph G, denoted by $d_G(x, y)$, is the length of a shortest path in G between u and v, or ∞ if x, y lie in different connected components of G. The *diameter* of a graph is the maximum distance over all pairs of vertices.

By the *neighbors of a subgraph* $H \subseteq G$, denoted by $N^G(H)$, we mean the set of vertices in $V(G) \setminus V(H)$ that have at least one neighbor in H. We employ the same notation analogously to denote *neighbors of a subset of vertices* $N^G(S)$ for $S \subseteq V(G)$. If X is a subset of vertices disjoint from S, then $N^G_X(S)$ is the set $N^G(S) \cap X$. The same notation naturally extends to a subgraph $H \subseteq G$, that is, $N^G_X(H)$. If S_1, S_2 are subsets of vertices of a graph, we denote by $N^G_{S_1}(S_2)$ the set of vertices in S_1 that have a neighbor in S_2. The *distance* between a vertex v and a set of vertices S is defined as $d_G(v, S) = \min_{u \in S} d(v, u)$, where $d_G(v, u)$ denotes the usual distance.

For $v \in V$, we denote by $N^G(v)$ the *neighborhood* of v, namely $N^G(v) = \{u \in V : \{u, v\} \in E\}$. The *closed neighborhood* $N^G[v]$ of v is $N_G(v) \cup \{v\}$. In the same way we define $N^G[S]$ for $S \subseteq V$ as $N^G[S] = \bigcup_{v \in S} N_G[v]$, and $N(S) = N[S] \setminus S$. More generally, for an integer $r \geqslant 0$, the *rth neighborhood* of a vertex $N^G_r(v) := \{w \in G \mid d_G(v, w) \leqslant r\}$ is the set of vertices within distance at most r to v. in particular we have that $N^G_0(v) = \{v\}$ and $N^G_1(v) = N^G[v]$.

A graph on n vertices is called *complete* if it contains an edge between each pair of vertices, and is denoted by K_n. The complete graph on three vertices is known as the *triangle*. The *path* on n vertices $v_0, \ldots, v_{n-1}$ with the $n-1$ edges $\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{n-2}, v_{n-1}\}$ is denoted by P_n. The *length* is a path is its number of edges. The *cycle* on n vertices obtained from P_n by adding the edge $\{v_{n-1}, v_0\}$ is denoted by C_n. A graph G is k-partite if $V(G)$ can be partitioned into k classes $V_0, \ldots, V_{k-1}$ such that there are only edges between classes V_i and V_j with $i \neq j$. The 2-partite (resp. 3-partite) graphs are known as *bipartite* (resp. *tripartite*).

A graph is *planar* if it can be drawn in the plane without edge crossings. A graph is an *apex* graph if it contains a vertex whose removal results in a planar graph.

In all the notations defined above, we may drop the the subscripts or superscripts referring to the graph when it is clear from the context.

3.1.2 Graph minors

Let $G = (V, E)$ be a simple undirected graph and let $e = \{x, y\} \in E$. We define $E^G(v) = \{\{v, u\} \mid u \in N^G(v)\}$. We denote by $G \backslash e$ the graph G' where $G' = (V, E - \{e\})$ and we say that G' *occurs from* G *after an edge removal*. We also denote by G/e the graph G' where

$$G' = (V - \{x, y\} \cup \{v_{x,y}\},\ E - E^G(x) - E^G(y) \cup \{\{v_{xy}, z\} \mid z \in N^G(x, y)\}),$$

where $v_{xy} \notin V$ is a new vertex, not in G. In this case we say that G' *occurs from* G *after an edge contraction*.

If H occurs from a subgraph of G after a sequence of edge contractions, we say that H is a *minor* of G, and we denote it by $H \preceq_{\mathsf{m}} G$.

If H occurs from G itself after a sequence of edge contractions, we say that H is a *contraction* of G, and we denote it by $H \preceq_{\mathsf{c}} G$. Note that $H \preceq_{\mathsf{c}} G$ implies that $H \preceq_{\mathsf{m}} G$, but not vice-versa.

If H occurs from a subgraph of G after a sequence of edge contractions, such that each contracted edge has at least one endpoint with degree at most two, we say that H is a *topological minor* of G, and be denote it by $H \preceq_{\mathsf{tm}} G$. Note that $H \preceq_{\mathsf{tm}} G$ implies that $H \preceq_{\mathsf{m}} G$, but not vice-versa. Note also that $H \preceq_{\mathsf{tm}} G$ and $H \preceq_{\mathsf{c}} G$ do not imply each other in general. To see this, note that, for example, $P_3 \preceq_{\mathsf{tm}} K_3$ but $P_3 \not\preceq_{\mathsf{c}} K_3$ and that, if H and G are the graphs depicted in Figure 3.1, then $H \preceq_{\mathsf{c}} G$ but $H \not\preceq_{\mathsf{tm}} G$.

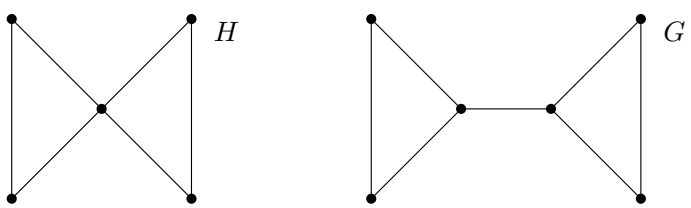

Figure 3.1: Graphs H and G such that $H \preceq_{\mathsf{c}} G$ but $H \not\preceq_{\mathsf{tm}} G$.

A graph G is *H-minor-free* (resp. *H-contraction-free*, *H-topological-minor-free*) if $H \not\preceq_{\mathsf{m}} G$ (resp. $H \not\preceq_{\mathsf{c}} G$, $H \not\preceq_{\mathsf{tm}} G$). If $\mathcal{F}$ is a collection of graphs, a graph G is *$\mathcal{F}$-minor-free* if $H \not\preceq_{\mathsf{m}} G$ for every $H \in \mathcal{F}$. The same notation applies to the contraction and topological minor relations. For instance, by Kuratowski's Theorem [106, 191], planar graphs are exactly $\{K_5, K_{3,3}$-minor-free graphs.

Graph minors have been the central topic of the so-called *graph minor theory*. This deep theory, mainly developed by Robertson and Seymour in a long series of papers, describes the structure of graphs with excluded minors. It culminates in the Graph Minors Theorem [227], which states that every class of graphs closed under taking minors can be characterized by a finite set of excluded minors; this fact was previously known as Wagner's conjecture [106, 240]. Equivalently, it says that graphs are well-quasi ordered (WQO) by the minor relation, meaning that in every infinite sequence of finite graphs there are two of them such that one is a minor of the other. The theory also has significant algorithmic consequences. Robertson and Seymour [226] proved that every class of graphs that is

closed under taking minors (this means that if a graph G is inside the class, so is every minor of G) can be recognized in cubic time. Kawarabayashi et al. [179] improved this algorithm to a quadratic one.

3.1.3 Treewidth

One of the most crucial ingredients in graph minor theory is, undoubtedly, the notion of *treewidth*. Loosely speaking, treewidth is a graph invariant that quantifies the resemblance of a graph to the structure of a forest, in the sense that the more a graph "differs" from a forest, the higher its treewidth.

The concept of treewidth was originally introduced by Bertelé and Brioschi [52], under a different name. It was later rediscovered by Halin [168], again under a different name. Some years later, it was rediscovered by Robertson and Seymour [225], who coined the current name. They used treewidth with combinatorial purposes, in order to prove Wagner's conjecture [106, 240], but it turned out to have a number of algorithmic and practical applications, as discussed later. Probably the most famous result certifying the algorithmic importance of treewidth is Courcelle's theorem [90], which states that every decision problem expressible in Monadic Second Order Logic (cf. Section 3.1.4 for the formal definition) can be solved in linear time on graphs of bounded treewidth.

In order to define treewidth, we first need to define tree-decompositions.

A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(T, \mathcal{X})$, where $T = (I, F)$ is a tree, and $\mathcal{X} = \{B_i\},\ i \in I$ is a family of subsets of $V(G)$, called *bags* and indexed by the nodes of T, such that

1. each vertex $v \in V$ appears in at least one bag, i.e., $\bigcup_{i \in I} B_i = V$;
2. for each edge $e = \{x, y\} \in E$, there is an $i \in I$ such that $x, y \in B_i$; and
3. for each $v \in V$ the set of nodes indexed by $\{i \mid i \in I,\ v \in B_i\}$ forms a subtree of T.

The *width* of a tree-decomposition is defined as $\max_{i \in I}\{|B_i| - 1\}$. The *treewidth* of G, denoted by $\mathsf{tw}(G)$, is the minimum width of a tree-decomposition of G. It is easy to see that the graphs of treewidth one are exectly the forests. We refer to the book of Kloks [184] and the survey of Bodlaender [57] for an introductory overview on treewidth and its use in algorithmic graph theory.

The vertices of the tree T are usually referred to as *nodes*. A *rooted tree-decomposition* is a tree-decomposition $(T, \mathcal{X} = \{B_x \mid x \in V(T)\})$ in which a distinguished node $r \in V(T)$ has been selected as the *root*. The bag B_r is called the *root-bag*. Note that the root defines a child/parent relation between every pair of adjacent nodes in T, and ancestors/descendants in the usual way. A node without children is called a *leaf*.

From an algorithmic point of view, tree-decompositions are particularly relevant for performing dynamic programming in a bottom-up manner from the leaves to the root of the tree. In this context, given a bag B of a tree-decomposition with a rooted tree T, we denote by T_B the subtree rooted at the node corresponding to bag B, and by $G_B := G[\bigcup_{x \in T_B} W_x]$ the subgraph of G induced by the vertices appearing in the bags corresponding to the nodes of T_B. If a bag B is associated with a node x of T, we may interchangeably use G_B or G_x.

A tree-decomposition $(T, \mathcal{X})$ rooted at a distinguished node t_r is *nice* if the following conditions are fulfilled:

- $B_{t_r} = \emptyset$ and this is the only empty bag,
- each node has at most two children,
- for each leaf $t \in V(T)$, $|B_t| = 1$,
- if $t \in V(T)$ has exactly one child t', then either
  - $B_t = B_{t'} \cup \{v\}$ for some $v \notin B_{t'}$ and t is called an *introduce-vertex* node, or
  - $B_t = B_{t'} \setminus \{v\}$ for some $v \in B_{t'}$ and t is called a *forget-vertex* node, or
  - $B_t = B_{t'}$, t is associated with an edge $\{x, y\} \in E(G)$ with $x, y \in B_t$, and t is called an *introduce-edge* node. We add the constraint that each edge of G labels exactly one node of T.
- and if $t \in V(T)$ has exactly two children t' and t'', then $B_t = B_{t'} = B_{t''}$. Then t is called a *join* node.

Note that we follow the definition of nice tree-decomposition given in [96], which slightly differs from the usual one [184]. Given a tree-decomposition, then we can build a nice tree-decomposition of G with the same width in polynomial time [96, 184]. As we will see in the next chapters, nice tree-decompositions are particularly useful for performing dynamic programming.

A tree-decomposition $(T, \mathcal{X})$ in which T is a path is called a *path-decomposition*. The width of a path decomposition $(T, \mathcal{X})$ is the width of $(T, \mathcal{X})$ as a tree-decomposition. The *pathwidth* is defined exactly as above by restricting to the path-decompositions, and is denoted by pw. A path decomposition $(T, \mathcal{X})$, with T a path of length n, is usually denoted by $(B_0, B_1, \ldots, B_n)$. By definition, it is clear that any graph G satisfies $\mathsf{tw}(G) \leqslant \mathsf{pw}(G)$.

3.1.4 (Counting) Monadic Second Order Logic

Monadic Second Order Logic (MSO) is an extension of First Order Logic that allows quantification over sets of objects. We identify graphs with relational structures over a vocabulary τ_{graph}, consisting of the unary relation symbols Vert and Edge and the binary relation symbol Inc. A graph $G = (V, E)$ is then represented by a τ_{graph}-structure $\mathcal{G}$ with universe $U(\mathcal{G}) = V \cup E$ such that:

- $\textsc{Vert}^{\mathcal{G}} = V$ and $\textsc{Edge}^{\mathcal{G}} = E$ represent the vertex set and the edge set, respectively, and
- $\textsc{Inc}^{\mathcal{G}} = \{\, (v, e) \mid v \in V, e \in E \text{ and } v \text{ is incident to } e \,\}$ represents the incidence relation.

A *Monadic Second Order* formula contains two types of variables: *individual variables* to be used for elements of the universe, usually denoted by lowercase letters $x, y, z, \ldots$ and *set variables* to be used for subsets of the universe, usually denoted by uppercase letters

$X, Y, Z, \ldots$. Atomic formulas on τ_{graph} are: $x = y$, $x \in X$, $x \in \textsc{Vert}$, $x \in \textsc{Edge}$, and $\textsc{Inc}(x, y)$ for all individual variables x, y and set variables X. MSO formulas on τ_{graph} are built from the atomic formulas using Boolean connectives $\neg, \wedge, \vee$, and quantification $\exists x, \forall x, \forall X, \forall Y$ for individual variables x and set variables X. MSO formulas are interpreted in τ_{graph}-structures in the natural way, e.g., $\textsc{Inc}(x, y)$ being true iff in G the vertex v represented by x is incident to the edge e represented by y.

In a *Counting* Monadic Second Order (CMSO) formula, we have additional atomic formulas $\mathbf{card}_{n,p}(X)$ on set variables X, which are true if the set U represented by the variable X has size $n \pmod p$. We refer to [91, 131] for a more detailed presentation on (C)MSO logic. In a p-MIN-CMSO graph problem (respectively, p-MAX-CMSO or p-EQ-CMSO) Π, one has to decide the existence of a set S of at most k vertices/edges (respectively, at least k or exactly k) in an input graph G such that the CMSO expressible predicate $P_\Pi(G, S)$ is satisfied.

3.2 Parameterized complexity

Parameterized complexity deals with algorithms for decision problems whose instances consist of a pair (x, k), where k is a secondary measurement of the input known as the *parameter*. A major goal in parameterized complexity is to investigate whether a problem with parameter k admits an algorithm with running time $f(k) \cdot |x|^{\mathcal{O}(1)}$, where f is a computable function depending only on the parameter and $|x|$ represents the input size. Parameterized problems that admit such algorithms are called *fixed-parameter tractable* and the class of all such problems is denoted by FPT.

This area originated in the 90's after the seminal work of Downey and Fellows (cf. for instance the foundational articles [113, 114]), and it has become nowadays one of the most active fields in theoretical computer science. This fact is testified by the fours books that have been already published on this subject [94, 115, 131, 214], or by the number of accepted articles on this topic in some of the most competitive conferences in the field such as STOC, FOCS, SODA, ICALP, ESA, or STACS.

In this section we do not pretend to provide a complete and formal introduction to the field of parameterized complexity, but to briefly introduce some of the most relevant notions, with particular emphasis on the concepts that will be used in the next chapters.

During the last decades, parameterized complexity theory has brought forth several algorithmic meta-theorems that imply that a wide range of problems are in FPT (see [188] for a survey). For instance, as mentioned in Section 3.1.3, Courcelle's theorem [90] states that every decision problem expressible in Monadic Second Order Logic can be solved in linear time when parameterized by the treewidth of the input graph. At the price of generality, such algorithmic meta-theorems may suffer from the fact that the function $f(k)$ is huge [144, 189] or non-explicit [90, 226]. Therefore, it has become a central task in parameterized complexity to provide FPT algorithms such that the behavior of the function $f(k)$ is *reasonable*; in other words, a function $f(k)$ that could lead to a practical algorithm.

Towards this goal, we say that an FPT parameterized problem is solvable in *single-exponential* time if there exists an algorithm solving it in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. For instance, recent results have shown that broad families of problems admit (deterministic or randomized) single-exponential algorithms parameterized by treewidth [96, 111, J21]. On the other hand, single-exponential algorithms are unlikely to exist for certain parameterized problems [96, 199]. Parameterizing by the size of the desired solution, in the case of Vertex Cover the existence of a single-exponential algorithm has been known for a long time, but it took a while to witness the first (deterministic) single-exponential algorithm for Feedback Vertex Set [100, 165].

In order to define parameters that are closed under taking minors, let us give a more formal definition of a parameter. Namely, a *parameter* p is any function mapping graphs to non-negative integers. Examples of parameters are the size of a minimum vertex cover or the size of a maximum clique. The *parameterized problem* associated with a parameter p asks, for some fixed k, whether $\mathsf{p}(G) \geqslant k$ for a given graph G. We say that a parameter p is *minor closed* if whenever H is a minor of G, $\mathsf{p}(H) \leqslant \mathsf{p}(G)$. Examples of minor closed parameters are the size of a longest path or the size of a minimum feedback vertex set. A powerful algorithmic consequence of the Graph Minors Theorem [227] is that every minor closed parameterized problem is in FPT, i.e., it admits an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$). The drawback of this result is that the function $f(k)$ and the constants hidden in the big-Oh notation can be huge, and therefore these general FPT algorithms can be of limited practical value.

A fundamental concept in parameterized complexity is that of *kernelization*. A kernelization algorithm, or just *kernel*, for a parameterized problem Π takes an instance (x, k) of the problem and, in time polynomial in $|x| + k$, outputs an instance (x', k') such that $|x'|, k' \leqslant g(k)$ for some function g, and $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$. The function g is called the *size* of the kernel and may be viewed as a measure of the "compressibility" of a problem using polynomial-time preprocessing rules. A kernel is called *polynomial* (resp. *linear*) if the function $g(k)$ is a polynomial (resp. linear) function in k. It is nowadays a well-known result in the area that a decidable problem is in FPT if and only if it has a kernelization algorithm. However, the kernel that one obtains in this way is typically of size at least exponential in the parameter. A natural problem in this context is to find polynomial or linear kernels for problems that are in FPT.

A breakthrough result of Bodlaender et al. [61] gave the first framework for proving that certain parameterized problems do not admit polynomial kernels, by establishing so-called *composition algorithms*. Together with a result of Fortnow and Santhanam [143] this allows to exclude polynomial kernels under the assumption that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, otherwise implying a collapse of the polynomial hierarchy to its third level [244]. A very successful notion for proving such results is that of cross-composition, introduced by Bodlaender et al. [64].

One of the main interests of parameterized complexity is that it can be proved that, under some reasonable complexity assumptions, not all NP-hard problems are FPT, and the distinction between those that are and those that are not provides a refined classification of "hard" problems, which is not possible in the classical polynomial/NP-hard dichotomy of computational problems.

More precisely, within parameterized problems, the class W[1] may be seen as the parameterized equivalent to the class NP of classical optimization problems. Without entering into details (see [94, 115, 131, 214] for the formal definitions), a parameterized problem being W[1]-*hard* can be seen as a strong evidence that this problem is *not* FPT. The canonical example of W[1]-hard problem is Independent Set parameterized by the size of the solution.

The class W[2] of parameterized problems is a class that contains W[1], and such that the problems that are W[2]-*hard* are even more unlikely to be FPT than those that are W[1]-hard (again, see [94, 115, 131, 214] for the formal definitions). The canonical example of W[2]-hard problem is Dominating Set parameterized by the size of the solution.

For $i \in [1, 2]$, to transfer W[i]-hardness from one problem to another, one uses a *parameterized reduction*, which given an input $I = (x, k)$ of the source problem, computes in time $f(k) \cdot |I|^c$, for some computable function f and a constant c, an equivalent instance $I' = (x', k')$ of the target problem, such that k' is bounded by a function depending only on k.

Hence, an equivalent definition of W[1]-hard (resp. W[2]-hard) problem is any problem that admits a parameterized reduction from Independent Set (resp. Dominating Set) parameterized by the size of the solution. The theory of parameterized complexity is based on the assumption that FPT $\neq$ W[1], which is stronger than P $\neq$ NP. Hence, assuming that FPT $\neq$ W[1], proving the W[1]-hardness of a parameterized problem rules out the existence of an FPT algorithm to solve it.

Even if a parameterized problem is W[1]-hard or W[2]-hard, it may still be solvable in polynomial time for *fixed* values of the parameter: such problems are said to belong to the complexity class XP. Formally, a parameterized problem whose instances consist of a pair (x, k) is in XP if it can be solved by an algorithm with running time $f(k) \cdot |x|^{g(k)}$, where f, g are computable functions depending only on the parameter and $|x|$ represents the input size. For example, Independent Set and Dominating Set parameterized by the solution size are easily seen to belong to XP.

There are even harder parameterized problems: they may be NP-hard for fixed values of the parameter, like Vertex Coloring parameterized by the number of colors. Such problems are called para-NP-hard. Note that, unless P = NP, a para-NP-hard problem cannot be in XP, hence it cannot be FPT either.

Logically, if stronger complexity assumptions are assumed, stronger hardness results might be proved. In this direction, it is worth mentioning the famous Exponential Time Hypothesis (ETH) of Impagliazzo et al. [170], stating that there exists $\delta > 0$ such that the 3-Sat problem with formulas on n variables does not allow an algorithm running in time $2^{\delta n} \cdot n^{\mathcal{O}(1)}$. It is well-known [94] that the ETH implies that FPT $\neq$ W[1], which in turn implies that P $\neq$ NP. For instance, a typical result that one can prove assuming the ETH (cf. [131, Chapter 16]) is that Vertex Cover parameterized by the size of the solution cannot be solved in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$.

3.3 Some classical problems

For the sake of completeness, we provide here the definition of some classical (non-parameterized) problems mentioned in this manuscript, all being NP-hard except Maximum Matching that is in P. For a complete list of classical NP-hard optimization problems, we refer the reader to [154].

Maximum Matching
Input: A graph $G = (V, E)$.
Output: A subset $E' \subseteq E$ of the maximum size such that no two edges in E' share a common endpoint.

Maximum Clique
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of the maximum size such that there is an edge in E between any two vertices in S.

Maximum Independent Set
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of the maximum size such that there is no edge in E between any two vertices in S.

Minimum Vertex Coloring
Input: A graph $G = (V, E)$.
Output: A function $f : V \to \{1, 2, \ldots, c\}$ such that $f(u) \neq f(v)$ for each edge $\{u, v\} \in E$ and such that c is minimized.

Minimum Edge Coloring
Input: A graph $G = (V, E)$.
Output: A function $f : E \to \{1, 2, \ldots, c\}$ such that $f(e) \neq f(e')$ whenever e and e' share an endpoint and such that c is minimized.

Minimum Vertex Cover
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that for every edge $e = \{u, v\} \in E$, either $u \in S$ or $v \in S$.

Minimum Feedback Vertex Set
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that $G[V \setminus S]$ has no cycles.

MINIMUM ODD CYCLE TRANSVERSAL
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that $G[V \setminus S]$ is bipartite.

MINIMUM TREEWIDTH-t VERTEX DELETION
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that $G[V \setminus S]$ has treewidth at most t.

MINIMUM DOMINATING SET
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that for every vertex $u \in V \setminus S$ there is a vertex $v \in S$ such that $\{u, v\} \in E$.

MINIMUM CONNECTED DOMINATING SET
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that for every vertex $u \in V \setminus S$ there is a $v \in S$ such that $\{u, v\} \in E$, and such that $G[S]$ is connected.

MINIMUM ACYCLIC DOMINATING SET
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that for every vertex $u \in V \setminus S$ there is a $v \in S$ such that $\{u, v\} \in E$, and such that $G[S]$ is a forest.

MINIMUM INDEPENDENT DOMINATING SET
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that for every vertex $u \in V \setminus S$ there is a $v \in S$ such that $\{u, v\} \in E$, and $G[S]$ is an independent set.

MINIMUM TOTAL DOMINATING SET
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that for every vertex $u \in V$ there is a vertex $v \in S$ such that $\{u, v\} \in E$.

MINIMUM EDGE DOMINATING SET
Input: A graph $G = (V, E)$.
Output: A subset $F \subseteq E$ of minimum size such that for every edge $\{u, v\} \in E \setminus F$ there is an edge $\{w, z\} \in F$ such that $\{u, v\} \cap \{w, z\} \neq \emptyset$.

LONGEST PATH
Input: A graph $G = (V, E)$.
Output: A path in G with the maximum number of edges.

HAMILTONIAN PATH
Input: A graph $G = (V, E)$.
Question: Does G contain a path of length $|V|$?

HAMILTONIAN CYCLE
Input: A graph $G = (V, E)$.
Question: Does G contain a cycle of length $|V|$?

CYCLE PACKING
Input: A graph $G = (V, E)$.
Output: A collection of vertex-disjoint cycles in G with maximum number of cycles.

MAXIMUM INDUCED MATCHING
Input: A graph $G = (V, E)$.
Output: A collection of edges in G of maximum cardinality such that the subgraph induced by their endvertices is 1-regular.

MINIMUM CHORDAL VERTEX DELETION
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that $G[V \setminus S]$ is chordal.

MINIMUM SPLIT VERTEX DELETION
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that $G[V \setminus S]$ is a split graph.

MINIMUM INTERVAL VERTEX DELETION
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that $G[V \setminus S]$ is an interval graph.

MINIMUM PROPER INTERVAL VERTEX DELETION
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of minimum size such that $G[V \setminus S]$ is a proper interval graph.

3-SAT
Input: A formula ϕ in conjunctive normal form, in which each clause has 3 variables.
Question: Is there a 0/1 assignment of the variables that satisfies every clause of ϕ?

Chapter 4

Summary of my contributions

This chapter is devoted to provide a summary of (most of) the research contributions that I have made after my Ph.D. We present these contributions divided into four categories: FPT algorithms in Section 4.1, kernelization in Section 4.2, combinatorial results in Section 4.3, and results for problems arising from applications in Section 4.4.

For each article, we provide a succinct description of the results (like an abstract with some extra explanations, figures, or contextualization), and in some cases links to other contributions. Within each category, the results are presented chronologically, from older to newer research, and not by publication date. Sometimes, several articles are grouped within the same item.

In order to keep the exposition as concise as possible, not all the definitions of the notions mentioned in this chapter are provided in the manuscript. We refer the reader to the corresponding articles for all the details.

Contents

4.1 FPT algorithms

These are some of the contributions that I have made to the area of FPT algorithms during the last years. A number of these contributions are devoted to the existence (or not) of *single-exponential algorithms* for several problems or families of problems, on both general and particular graph classes. Some of the articles summarized below also contain "classical" complexity results, such as NP-hardness results, and polynomial-time algorithms.

1. **Dynamic programming for topologically constrained graphs**.

   In [J21] we provide a framework for the design and analysis of dynamic programming algorithms for surface-embedded graphs on n vertices and branchwidth[1] at most k.

[1] Informally, *branchwidth* is a graph invariant (very similar to treewidth) that measures the topological complexity of a graph, in the sense that the higher the branchwidth of a graph is, the more it differs from a tree.

Our technique applies to general families of problems where standard dynamic programming runs in $2^{\mathcal{O}(k \cdot \log k)} \cdot n$ steps. Intuitively, such a running time appears when the natural approach is to enumerate, in the tables of the dynamic programming algorithm, all possible *packings* (that is, collections of pairwise disjoint sets) of the considered separator in the tree-like decomposition. For instance, this is the case when solving the Longest Path or the Connected Vertex Cover problems; such problems are commonly called *connectivity problems*.

Our approach combines tools from topological graph theory and analytic combinatorics. In particular, we introduce a new type of branch decomposition called *surface cut decomposition*, generalizing sphere cut decompositions of planar graphs introduced by Seymour and Thomas [229], which has nice combinatorial properties. Namely, the number of partial solutions that can be arranged on a surface cut decomposition can be upper-bounded by the number of non-crossing partitions on surfaces with boundary, which we enumerate in a separate article [J20]. It follows that partial solutions can be represented by a single-exponential (in the branchwidth k) number of configurations. This proves that, when applied on surface cut decompositions, dynamic programming runs in $2^{\mathcal{O}(k)} \cdot n$ steps. That way, we considerably extend the class of problems that can be solved in running times with a *single-exponential dependence* on branchwidth and unify/improve most previous results in this direction.

In a subsequent work [C33], we extend the above framework to the more general class of H-minor-free graphs. The main contribution here is to introduce a new type of branch decomposition for H-minor-free graphs, called an *H-minor-free cut decomposition*, and to show that they can be constructed in $O_h(n^3)$ steps, where the hidden constant depends exclusively on H. This decomposition generalizes the one for graphs of bounded genus discussed above [J21].

We show that the separators of such decompositions have connected packings whose behavior can be described in terms of a combinatorial object called *ℓ-triangulation*, defined as follows. Let $\mathbb{D}_k$ be a disc with k vertices on its border, labeled counterclockwise as $1, 2, \ldots, k$. By an *ℓ-triangulation* of $\mathbb{D}_k$ we mean a maximal set of diagonals with no pairwise crossing-set of size $\ell+1$. In other words, the graph whose vertices are the diagonals of the ℓ-triangulation and there is an edge between two diagonals if and only if they cross in an internal vertex, does not contain $K_{\ell+1}$ as a subgraph. This concept generalizes the classical notion of *triangulation* of a disc, as 1-triangulations correspond to triangulations.

Using the notion of ℓ-triangulation, our main result is that when applied on H-minor-free cut decompositions, dynamic programming runs in $2^{\mathcal{O}_h(k)} \cdot n^{\mathcal{O}(1)}$ steps. This broadens substantially the class of problems that can be solved deterministically in *single-exponential* time for H-minor-free graphs.

It should be mentioned that, some time after the above results appeared, several articles appeared providing single-exponential algorithms for many connectivity problems on *general graphs* [60, 96, 138], hence improving over the results we presented in [C33, J21].

2. **Fast minor testing on general and planar graphs**.

   The H-Minor containment problem asks whether a graph G contains some fixed graph H as a minor, that is, whether H can be obtained by some subgraph of G after contracting edges. The derivation of a polynomial-time algorithm for H-Minor containment is one of the most important and technical parts of the Graph Minors theory of Robertson and Seymour and it is a cornerstone for most of the algorithmic application of this theory. H-Minor containment for graphs of bounded branchwidth is a basic ingredient of this algorithm. A first solution of this problem, based on the ideas introduced by Robertson and Seymour, was given by Hicks [169], providing an algorithm that in time $\mathcal{O}(3^{k^2} \cdot (h+k-1)! \cdot m)$ decides if a graph G with m edges and branchwidth k, contains a fixed graph H on h vertices as a minor.

   In [J1] we improve the dependence on k of Hicks' result by showing that checking if H is a minor of G can be done in time $\mathcal{O}(2^{(2k+1)\cdot\log k} \cdot h^{2k} \cdot 2^{2h^2} \cdot m)$. We set up an approach based on a combinatorial object called *rooted packing*, which captures the properties of the potential models of subgraphs of H that we seek in our dynamic programming algorithm. This formulation with rooted packings allows us to speed up the algorithm when G is embedded in a fixed surface, obtaining the first single-exponential algorithm for minor containment testing. Namely, it runs in time $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$, with $n = |V(G)|$. Finally, we show that slight modifications of our algorithm permit to solve some related problems within the same time bounds, like induced minor or contraction containment.

   In another article [J2], we focus on the planar case to provide an improved algorithm. Namely, we give an algorithm that, given a planar n-vertex graph G and an h-vertex graph H, either finds in time $\mathcal{O}(2^{\mathcal{O}(h)} \cdot n + n^2 \cdot \log n)$ a model of H in G, or correctly concludes that G does not contain H as a minor. Our algorithm is the first *single-exponential* algorithm for this problem and improves all previous minor testing algorithms in planar graphs. In addition, we can *enumerate* and *count* the number of models within the same time bounds.

   Our technique is based on a novel approach called *partially embedded dynamic programming*, which differs substantially from the one we use in [J1]. More precisely, our technique is inspired by the technique of *embedded dynamic programming* introduced by Dorn [110] for solving Planar Subgraph Isomorphism for a pattern of size h and an input graph of size n in time $2^{\mathcal{O}(h)} \cdot n$. There, one controls the partial solutions by the ways the separators of G can be routed through the pattern. The difference (and difficulty) concerning Planar H-Minor Containment is that we look for a model M of size $\mathcal{O}(n)$ out of $2^{\mathcal{O}(n)}$ possible non-isomorphic models of H in G. In partially embedded dynamic programming, we look for potential models of H in G with a "magnifying glass" only at a given separator S of G. That is, we consider a collection $\mathcal{A}$ of graphs A arising from 'decontracting' a part of H, namely the part interacting with S. In this way, each graph A behaves like a subgraph of G inside the intersection with S, and outside that intersection A behaves like a minor of G; this is why we call our dynamic programming technique "partially embedded".

3. **Hitting and harvesting pumpkins**.

   The *c-pumpkin* is the graph with two vertices linked by $c \geqslant 1$ parallel edges. A 5-pumpkin is shown in Figure 4.1.

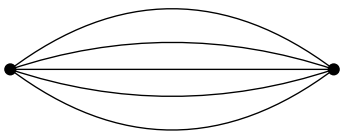

Figure 4.1: A 5-pumpkin.

   A *c-pumpkin-model* in a graph G is a pair $\{A, B\}$ of disjoint subsets of vertices of G, each inducing a connected subgraph of G, such that there are at least c edges in G between A and B. In [J14] we focus on hitting and packing c-pumpkin-models in a given graph in the realm of approximation algorithms and parameterized algorithms. We give an FPT algorithm running in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ deciding, for any fixed $c \geqslant 1$, whether all c-pumpkin-models can be hit by at most k vertices. This generalizes known single-exponential FPT algorithms for Vertex Cover and Feedback Vertex Set, which correspond to the cases $c = 1, 2$ respectively. Finally, we present an $\mathcal{O}(\log n)$-approximation algorithm for both the problems of hitting all c-pumpkin-models with a smallest number of vertices, and packing a maximum number of vertex-disjoint c-pumpkin-models.

   This article, beside being important by its own at the moment it was published, played an important role in forthcoming fundamental results in the area of single-exponential algorithms for the very general Planar-$\mathcal{F}$-Deletion problem [137, J15]; see Chapter 5.

4. **Single-exponential algorithms via protrusion decompositions**.

   In many cases, the key ingredient in order to solve a hard graph problem is to find an appropriate *decomposition* of the input graph, which allows to take advantage of the structure given by the graph class and/or the problem under study. In [J15] we follow this paradigm and present a novel linear-time algorithm to compute a decomposition for graphs G that have a set $X \subseteq V(G)$, called *t-treewidth-modulator*, such that the treewidth of $G - X$ is at most some constant $t - 1$. This decomposition is called a *protrusion decomposition*. We then exploit this decomposition in two different ways: to *analyze* the size of kernels (as discussed later) and to obtain *efficient* FPT algorithms. Let us focus on this second application here.

   Let $\mathcal{F}$ be a fixed finite family of graphs containing at least one planar graph. Given an n-vertex graph G and a non-negative integer k, Planar-$\mathcal{F}$-Deletion asks whether G has a set $X \subseteq V(G)$ such that $|X| \leqslant k$ and $G - X$ is H-minor-free for every $H \in \mathcal{F}$. This problem encompasses a number of well-studied parameterized problems such as Vertex Cover, Feedback Vertex Set, and Treewidth-t Vertex Deletion.

   As an application of our decomposition algorithm, we present the first *single-exponential* algorithm to solve Planar-$\mathcal{F}$-Deletion. Namely, our algorithm runs in time $2^{\mathcal{O}(k)} \cdot n^2$, which is asymptotically optimal with respect to k. So far, single-

exponential algorithms were only known for special cases of the family $\mathcal{F}$.

Full details of this contribution are provided in Chapter 5.

5. **Parameterized domination on circle graphs**.

   A *circle graph* is the intersection graph of a set of chords in a circle. See Figure. 4.2 for an example of a circle graph G together with a circle representation of it.

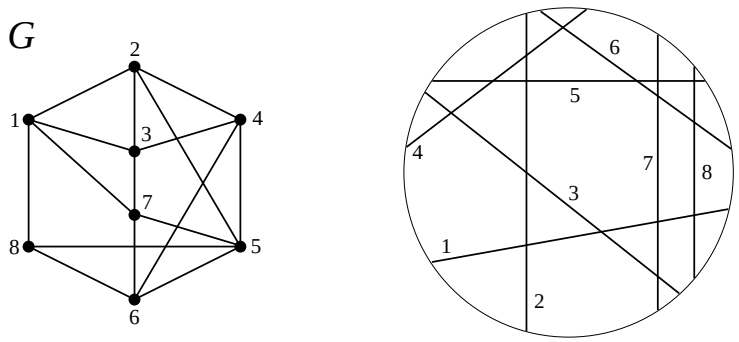

Figure 4.2: A circle graph G on 8 vertices together with a circle representation of it.

   Keil [181] proved that Dominating Set, Connected Dominating Set, and Total Dominating Set are NP-complete on circle graphs. To the best of our knowledge, nothing was known about the parameterized complexity of these problems on circle graphs. In [J7] we prove the following results, which contribute in this direction:

   - Dominating Set, Independent Dominating Set, Connected Dominating Set, Total Dominating Set, and Acyclic Dominating Set are W[1]-hard on circle graphs, parameterized by the size of the solution.
   - Whereas both Connected Dominating Set and Acyclic Dominating Set are W[1]-hard on circle graphs, it turns out that Connected Acyclic Dominating Set is polynomial-time solvable on circle graphs.
   - If T is a *given* tree, deciding whether a circle graph G has a dominating set inducing a graph isomorphic to T is NP-complete when T is in the input, and FPT when parameterized by $t = |V(T)|$. We prove that the FPT algorithm runs in subexponential time, namely $2^{\mathcal{O}(t \cdot \frac{\log \log t}{\log t})} \cdot n^{\mathcal{O}(1)}$, where $n = |V(G)|$.

6. **The role of planarity in connectivity problems parameterized by treewidth**.

   For some years it was believed that for "connectivity" problems such as Hamiltonian Cycle, algorithms running in time $2^{\mathcal{O}(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ –called *single-exponential*, as mentioned in the introduction– existed only on planar [112] and other topologically constrained graph classes [111, C33, J21], where tw stands for the treewidth of the n-vertex input graph. This was recently disproved by Cygan et al. [96], Bodlaender et al. [60], and Fomin et al. [138], who provided single-exponential algorithms on

general graphs for most connectivity problems that were known to be solvable in single-exponential time on topologically constrained graphs. In [J6] we further investigate the role of planarity in connectivity problems parameterized by treewidth, and convey that several problems can indeed be distinguished according to their behavior on planar graphs.

Known results from the literature [112, 199] imply that there exist problems, like Cycle Packing, that *cannot* be solved in time $2^{o(\mathsf{tw}\log\mathsf{tw})}\cdot n^{\mathcal{O}(1)}$ on general graphs but that can be solved in time $2^{\mathcal{O}(\mathsf{tw})}\cdot n^{\mathcal{O}(1)}$ when restricted to planar graphs. Our main contribution is to show that there exist natural problems that can be solved in time $2^{\mathcal{O}(\mathsf{tw}\log\mathsf{tw})}\cdot n^{\mathcal{O}(1)}$ on general graphs but that *cannot* be solved in time $2^{o(\mathsf{tw}\log\mathsf{tw})}\cdot n^{\mathcal{O}(1)}$ even when restricted to planar graphs. The existence of such problems was not known prior to our work.

Furthermore, we prove that Planar Cycle Packing and Planar Disjoint Paths *cannot* be solved in time $2^{o(\mathsf{tw})}\cdot n^{\mathcal{O}(1)}$. The mentioned negative results hold unless the Exponential Time Hypothesis (ETH) fails. We feel that our results constitute a first step in a subject that can be further exploited.

7. **Improved FPT algorithms for weighted independent set on bull-free graphs**.

Recently, Thomassé et al. [235] gave an FPT algorithm for Weighted Independent Set on bull-free graphs parameterized by the weight of the solution, running in time $2^{\mathcal{O}(k^5)}\cdot n^9$. The bull graph is depicted in Figure 4.3.

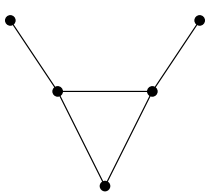

Figure 4.3: The bull.

In [C32] we improve this running time to $2^{\mathcal{O}(k^2)}\cdot n^7$. As a byproduct, we also improve the previous Turing-kernel for this problem from $\mathcal{O}(k^5)$ to $\mathcal{O}(k^2)$. Our approach consists in locally improving the algorithm provided by Thomassé et al. [235], by carefully analyzing some of the steps and using further structural properties of bull-free graphs given by Chudnovsky [86, 87].

Furthermore, for the subclass of bull-free graphs without holes[2] of length at most $2p-1$ for $p \geqslant 3$, we speed up the running time to $2^{\mathcal{O}(k\cdot k^{\frac{1}{p-1}})}\cdot n^7$. As p grows, this running time is asymptotically tight in terms of k, since we prove that for each integer $p \geqslant 3$, Weighted Independent Set cannot be solved in time $2^{o(k)}\cdot n^{\mathcal{O}(1)}$ in the class of $\{\text{bull}, C_4, \ldots, C_{2p-1}\}$-free graphs unless the ETH fails.

[2] A hole in a graph is an induced cycle of length at least 4.

8. **A polynomial-time algorithm for outerplanar diameter improvement.**

   The Outerplanar Diameter Improvement problem asks, given a graph G and an integer D, whether it is possible to add edges to G in a way that the resulting graph is outerplanar[3] and has diameter at most D. We provide in [J9] a dynamic programming algorithm that solves this problem in polynomial time. This algorithm heavily exploits the structure of outerplanar graphs, and runs in time $\mathcal{O}(n^3)$ for connected input graphs on n vertices, and in time $\mathcal{O}(n^7)$ or $\mathcal{O}(n^9)$ for disconnected input graphs, depending on whether D is odd or even, respectively. We believe that our approach might be interesting for generalizations or variations of Outerplanar Diameter Improvement, such as the one where we demand that the completed graph has fixed outerplanarity or is series-parallel.

   The Outerplanar Diameter Improvement problem is interesting because, in particular, it demonstrates several structural analogues to the celebrated and challenging Planar Diameter Improvement problem, where the resulting graph should, instead, be planar. The complexity status of this latter problem is open.

   It is worth mentioning that, while this result is not an FPT algorithm, but a polynomial-time one, it fits in this section since the complexity of Planar Diameter Improvement is a relevant open problem in parameterized complexity.

9. **An FPT 2-approximation for tree-cut decomposition.**

   The *tree-cutwidth* of a graph is a graph parameter defined by Wollan [242] with the help of tree-cut decompositions. In certain cases, tree-cutwidth appears to be more adequate than treewidth as an invariant that, when bounded, can accelerate the resolution of intractable problems. We skip the formal definition of tree-cutwidth, since it is somehow technical, but in Figure 4.4 we show the containment relations among graphs classes of bounded treewidth, of bounded tree-cutwidth, and of bounded treewidth and bounded degree.

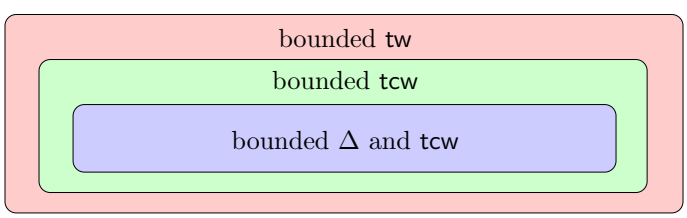

Figure 4.4: Relations among classes of bounded treewidth (tw) and tree-cut width (tcw).

   While designing algorithms for problems with bounded tree-cutwidth, it is important to have a parametrically tractable way to compute the exact value of this parameter or, at least, some constant approximation of it. In [C31] we give a parameterized 2-approximation algorithm for the computation of tree-cutwidth; for an input n-vertex graph G and an integer w, our algorithm either confirms that the tree-cutwidth of G is more than w or returns a tree-cut decomposition of G certifying that its tree-cutwidth is at most $2w$, in time $2^{\mathcal{O}(w^2 \log w)} \cdot n^2$. Prior to this work, no *constructive*

[3] A graph is *outerplanar* if it admits a planar embedding such that all vertices lie on the outerface.

parameterized algorithms, even approximated ones, existed for computing the tree-cutwidth of a graph. As a consequence of the Graph Minors series by Robertson and Seymour, only the *existence* of a decision algorithm was known.

10. **Parameterized algorithms for min-max multiway cut and list digraph homomorphism**.

    In [J16] we design FPT algorithms for two parameterized problems:

    - The first is List Digraph Homomorphism: given two digraphs G and H and a list of allowed vertices of H for every vertex of G, the question is whether there exists a homomorphism from G to H respecting the list constraints.
    - The second problem is a variant of Multiway Cut, namely Min-Max Multiway Cut: given a graph G, a non-negative integer ℓ, and a set T of r terminals, the question is whether we can partition the vertices of G into r parts such that (a) each part contains one terminal and (b) there are at most ℓ edges with only one endpoint in this part. We parameterize List Digraph Homomorphism by the number w of edges of G that are mapped to non-loop edges of H and we give a time $2^{\mathcal{O}(\ell \cdot \log h + \ell^2 \cdot \log \ell)} \cdot n^4 \cdot \log n$ algorithm, where h is the order of the host graph H. We also prove that Min-Max Multiway Cut can be solved in time $2^{\mathcal{O}((\ell r)^2 \log \ell r)} \cdot n^4 \cdot \log n$.

    Our approach introduces a general problem, called List Allocation, whose expressive power permits the design of parameterized reductions of both aforementioned problems to it. Then our results are based on an FPT algorithm for the List Allocation problem that is designed using a suitable adaptation of the *randomized contractions* technique introduced by Chitnis et al. [85].

11. **An $\mathcal{O}(\log \mathbf{OPT})$-approximation for covering/packing minor models of θ_r.**

    Given two graphs G and H, we define $\mathsf{v\text{-}cover}_H(G)$ (resp. $\mathsf{e\text{-}cover}_H(G)$) as the minimum number of vertices (resp. edges) whose removal from G produces a graph without any minor isomorphic to H. Also $\mathsf{v\text{-}pack}_H(G)$ (resp. $\mathsf{e\text{-}pack}_H(G)$) is the maximum number of vertex- (resp. edge-) disjoint subgraphs of G that contain a minor isomorphic to H. We denote by θ_r the r-pumpkin, that is, the graph with two vertices and r parallel edges between them; see Figure 4.1.

    When $H = \theta_r$, the parameters $\mathsf{v\text{-}cover}_H$, $\mathsf{e\text{-}cover}_H$, $\mathsf{v\text{-}pack}_H$, and $\mathsf{e\text{-}pack}_H$ are NP-hard to compute (for sufficiently large values of r). Drawing upon our combinatorial results in [J8], we give in [C29] an algorithmic proof that if $\mathsf{v\text{-}pack}_{\theta_r}(G) \leqslant k$, then $\mathsf{v\text{-}cover}_{\theta_r}(G) = \mathcal{O}(k \log k)$, and similarly for $\mathsf{e\text{-}pack}_{\theta_r}$ and $\mathsf{e\text{-}cover}_{\theta_r}$. In other words, the class of graphs containing θ_r as a minor has the vertex/edge Erdős-Pósa property, for every positive integer r.

    Using the algorithmic machinery of our proofs, we introduce a unified approach for the design of an $\mathcal{O}(\log \mathrm{OPT})$-approximation algorithm for $\mathsf{v\text{-}pack}_{\theta_r}$, $\mathsf{v\text{-}cover}_{\theta_r}$, $\mathsf{e\text{-}pack}_{\theta_r}$, and $\mathsf{e\text{-}cover}_{\theta_r}$ that runs in $\mathcal{O}(n \cdot \log n \cdot m)$ steps. Also, we derive several new Erdős-Pósa-type results from the techniques that we introduce.

12. **Parameterized complexity dichotomy for** (r,ℓ)-Vertex Deletion.

For two integers $r, \ell \geqslant 0$, a graph $G = (V, E)$ is an (r, ℓ)-*graph* if V can be partitioned into r independent sets and ℓ cliques. In the parameterized (r, ℓ)-Vertex Deletion problem, given a graph G and an integer k, one has to decide whether at most k vertices can be removed from G to obtain an (r, ℓ)-graph. This problem is NP-hard if $r + \ell \geqslant 1$ [73] and encompasses several relevant problems such as Vertex Cover and Odd Cycle Transversal.

The parameterized complexity of (r, ℓ)-Vertex Deletion was known for all values of (r, ℓ) except for $(2, 1)$, $(1, 2)$, and $(2, 2)$. In [J4] we prove, using the technique of *iterative compression*, that each of these three cases is FPT and, furthermore, solvable in single-exponential time, which is asymptotically optimal in terms of k.

A summary of the parameterized complexity of (r, ℓ)-Vertex Deletion is shown in Table 4.1, where for each value of (r, ℓ), the name of the problem (if any), the function $f(k)$ of the corresponding FPT algorithm, and the appropriate references are given. We denote by $\overline{\text{VC}}$ and $\overline{\text{OCT}}$ the complementary problems of Vertex Cover (VC) and Odd Cycle Transversal (OCT), respectively. We denote by Split D. the Split Vertex Deletion problem.

The results of [J4] correspond to the gray boxes, 'p-NP-c' stands for 'para-NP-complete', and 'P' means that the corresponding problem is polynomial-time solvable.

3	p-NP-c [73]	p-NP-c [73]	p-NP-c [73]	p-NP-c [73]
2	$\overline{\text{OCT}}$ 2.31^k [201]	3.31^k	3.31^k	p-NP-c [73]
1	$\overline{\text{VC}}$ 1.27^k [84]	Split D. 2^k [155]	3.31^k	p-NP-c [73]
0	P trivial	VC 1.27^k [84]	OCT 2.31^k [201]	p-NP-c [73]
ℓ \ r	0	1	2	3

Table 4.1: Summary of known results for the (r, ℓ)-Vertex Deletion problem. The results of [J4] correspond to the gray cells.

We also consider in [J4] the version of (r,ℓ)-Vertex Deletion where the set S of at most k vertices to be removed has to further satisfy that $G[S]$ is an independent set. We call this problem Independent (r,ℓ)-Vertex Deletion. Note that, in contrast to (r,ℓ)-Vertex Deletion, the cases (r,ℓ) and (ℓ,r) may not be symmetric anymore. This problem has received little attention in the literature and, excluding the most simple cases, to the best of our knowledge only the case $(2,0)$ has been studied by Marx et al. [204], who proved it to be FPT. Similarly to (r,ℓ)-Vertex Deletion, the problem is para-NP-complete if $\max\{r,\ell\} \geqslant 3$. As an additional motivation for studying this problem, note that solving Independent (r,ℓ)-Vertex Deletion on an input (G,k) corresponds exactly to deciding whether G is an $(r+1,\ell)$-graph where one of the independent sets has size at most k.

We manage to provide a complete characterization of the parameterized complexity of Independent (r,ℓ)-Vertex Deletion. The complexity landscape turns out to be richer than the one for (r,ℓ)-Vertex Deletion, and one should rather speak about a trichotomy: the problem is polynomial-time solvable if $r \leqslant 1$ and $\ell \leqslant 2$, NP-hard and FPT if $r = 2$ and $\ell \leqslant 2$, and para-NP-complete otherwise. In particular, as discussed at the end of the previous paragraph, it follows from our results that for $\ell \in \{0,1,2\}$, the recognition of the class of $(3,\ell)$-graphs such that one of the independent sets has size at most k is in FPT with parameter k. A summary of the complexity of Independent (r,ℓ)-Vertex Deletion is shown in Table 4.2, where our results correspond to the gray boxes. We would like to note that some of the polynomial cases, such as the case $(1,0)$, are not difficult to prove and may be already known, although we are not aware of it.

3	p-NP-c [73]	p-NP-c [73]	p-NP-c [73]	p-NP-c [73]
2	P	P	$2^{2^{\mathcal{O}(k^2)}}$	p-NP-c [73]
1	P	P	$2^{2^{\mathcal{O}(k^2)}}$	p-NP-c [73]
0	P trivial	IVC P	IOCT $2^{2^{\mathcal{O}(k^2)}}$ [204]	p-NP-c [73]
ℓ \ r	0	1	2	3

Table 4.2: Summary of known results for Independent (r,ℓ)-Vertex Deletion. The results of [J4] correspond to the gray cells.

13. **On the complexity of computing the k-restricted edge-connectivity of a graph**.

The *k-restricted edge-connectivity* of a graph G, denoted by $\lambda_k(G)$, is defined as the minimum size of an edge set whose removal leaves exactly two connected components each containing at least k vertices. This graph invariant, which can be seen as a generalization of a minimum edge-cut, has been extensively studied from a combinatorial point of view. However, very little was known about the complexity of computing $\lambda_k(G)$.

Recently, in the parameterized complexity community the notion of *good edge separation* of a graph has been defined and used extensively to design parameterized algorithms [85, 95, 180, 220, J16], which happens to be essentially the same as the k-restricted edge-connectivity.

Motivated by the relevance of this invariant from both combinatorial and algorithmic points of view, and motivated by our work in [J16] using good edge separations, we initiate in [J18] a systematic study of its computational complexity, with special emphasis on its parameterized complexity for several choices of the parameters. We provide a number of NP-hardness and W[1]-hardness results, as well as FPT algorithms.

The problems we consider and the results we obtain in [J18] are summarized in Table 4.3. A connected graph G is called *λ_k-connected* if $\lambda_k(G)$ exists.

Problem	Classical complexity	Parameterized complexity with parameter			
		$k+\ell$	k	ℓ	$k+\Delta$
Is G λ_k-connected ?	NPc, even if $\Delta \leqslant 5$	$\star$	FPT	$\star$	FPT
$\lambda_k(G) \leqslant \ell$?	NPh, even if G is λ_k-connected	FPT	W[1]-hard	FPT No poly kernels	FPT

Table 4.3: Summary of our results, where Δ denotes the maximum degree of the input graph G, and NPc (resp. NPh) stands for NP-complete (resp. NP-hard). The symbol '$\star$' denotes that the problem is not defined for that parameter.

14. **On the (parameterized) complexity of recognizing well-covered (r,ℓ)-graphs**.

An *(r,ℓ)-partition* of a graph G is a partition of its vertex set into r independent sets and ℓ cliques. As mentioned before, a graph is (r,ℓ) if it admits an (r,ℓ)-partition. A graph is *well-covered* if every maximal independent set is also maximum. A graph is *(r,ℓ)-well-covered* if it is both (r,ℓ) and well-covered.

In [C22] we consider two different decision problems. In the (r,ℓ)-Well-Covered Graph problem ((r,ℓ)WCG for short), we are given a graph G, and the question is whether G is an (r,ℓ)-well-covered graph. In the Well-Covered (r,ℓ)-Graph problem (WC(r,ℓ)G for short), we are given an (r,ℓ)-graph G together with an (r,ℓ)-partition of $V(G)$ into r independent sets and ℓ cliques, and the question is whether

G is well-covered. We classify most of these problems into P, coNP-complete, NP-complete, NP-hard, or coNP-hard.

In addition, we consider the parameterized complexity of these problems for several choices of parameters, such as the size α of a maximum independent set of the input graph, its neighborhood diversity, or the number ℓ of cliques in an (r, ℓ)-partition. In particular, we show that the parameterized problem of deciding whether a general graph is well-covered parameterized by α can be FPT-reduced to the WC$(0, \ell)$G problem parameterized by ℓ, and we prove that this latter problem is in XP but does not admit polynomial kernels unless NP $\subseteq$ coNP/poly.

15. **Maximum cuts in edge-colored graphs**.

The input of the Maximum Colored Cut problem consists of a graph $G = (V, E)$ with an edge-coloring $c : E \to \{1, 2, 3, \ldots, p\}$ and a positive integer $k > 0$, and the question is whether G has a nontrivial edge cut using at least k colors. The Colorful Cut problem has the same input but asks for a nontrivial edge cut using *all* colors.

Unlike what happens for the classical Maximum Cut problem, we prove in [C34] that both problems are NP-complete even on complete, planar, or bounded treewidth graphs. Furthermore, we prove that Colorful Cut is NP-complete even when each color class induces a clique of size at most 3, but is trivially solvable when each color induces a K_2. On the positive side, we prove that Maximum Colored Cut is FPT parameterized by either k or p, and that it admits a cubic kernel in both cases.

16. **Optimal algorithms for hitting (topological) minors on graphs of bounded treewidth**.

For a fixed collection of graphs $\mathcal{F}$, the $\mathcal{F}$-M-Deletion problem consists in, given a graph G and an integer k, decide whether there exists $S \subseteq V(G)$ with $|S| \leqslant k$ such that $G \setminus S$ does not contain any of the graphs in $\mathcal{F}$ as a minor. This problem has a huge expressive power, as it generalizes, for example, Vertex Cover, Feedback Vertex Set, or Vertex Planarization.

We are interested in the parameterized complexity of $\mathcal{F}$-M-Deletion when the parameter is the treewidth of G, denoted by tw. Our objective in [C27] is to determine, for a fixed $\mathcal{F}$, the smallest function $f_{\mathcal{F}}$ such that $\mathcal{F}$-M-Deletion can be solved in time $f_{\mathcal{F}}(\mathsf{tw}) \cdot n^{\mathcal{O}(1)}$ on n-vertex graphs. We also consider the version of the problem where the graphs in $\mathcal{F}$ are forbidden as *topological* minors, which we call $\mathcal{F}$-TM-Deletion. For the sake of readability, we use the notation $\mathcal{F}$-Deletion in statements that apply to *both* $\mathcal{F}$-M-Deletion and $\mathcal{F}$-TM-Deletion.

We present a number of upper and lower bounds for $\mathcal{F}$-Deletion parameterized by treewidth, several of them being tight. Namely, we prove the following results[4]:

(a) For every $\mathcal{F}$, $\mathcal{F}$-Deletion can be solved in time $\mathcal{O}^*\left(2^{2^{\mathcal{O}(\mathsf{tw} \cdot \log \mathsf{tw})}}\right)$.

(b) For every connected[5] $\mathcal{F}$ containing at least one planar graph (resp. subcubic

[4] We use the notation $\mathcal{O}^*(\cdot)$ that suppresses polynomial factors depending on the size of the input graph.
[5] A *connected* collection $\mathcal{F}$ is a collection containing only connected graphs.

planar graph), $\mathcal{F}$-M-Deletion (resp. $\mathcal{F}$-TM-Deletion) can be solved in time $\mathcal{O}^*\left(2^{\mathcal{O}(\mathsf{tw}\cdot\log\mathsf{tw})}\right)$. This is proved using and enhancing the machinery of boundaried graphs and small sets of representatives introduced by Bodlaender et al. [63].

(c) For any connected $\mathcal{F}$, $\mathcal{F}$-Deletion cannot be solved in time $\mathcal{O}^*(2^{o(\mathsf{tw})})$.

(d) When $\mathcal{F} = \{K_i\}$, the clique on i vertices, $\{K_i\}$-Deletion cannot be solved in time $\mathcal{O}^*(2^{o(\mathsf{tw}\cdot\log\mathsf{tw})})$ for $i \geqslant 4$. Note that $\{K_i\}$-Deletion can be solved in time $\mathcal{O}^*(2^{\mathcal{O}(\mathsf{tw})})$ for $i \leqslant 3$ [96], and that the case $i = 4$ is tight by item (b) above (as K_4 is planar).

(e) When $\mathcal{F} = \{C_i\}$, the cycle on i vertices, $\{C_i\}$-Deletion can be solved in time $\mathcal{O}^*(2^{\mathcal{O}(\mathsf{tw})})$ for $i \leqslant 4$, and cannot be solved in time $\mathcal{O}^*(2^{o(\mathsf{tw}\cdot\log\mathsf{tw})})$ for $i \geqslant 5$. Note that, by items (b) and (c) above, this settles completely the complexity of $\{C_i\}$-Deletion for every $i \geqslant 3$. The single-exponential algorithm for $\{C_4\}$-Deletion uses the rank-based approach introduced by Bodlaender et al. [60].

(f) When $\mathcal{F} = \{P_i\}$, the path on i vertices, $\{P_i\}$-Deletion can be solved in time $\mathcal{O}^*(2^{\mathcal{O}(\mathsf{tw})})$ for $i \leqslant 4$, and cannot be solved in time $\mathcal{O}^*(2^{o(\mathsf{tw}\cdot\log\mathsf{tw})})$ for $i \geqslant 6$. Note that, by items (b) and (c) above, this settles completely the complexity of $\{P_i\}$-Deletion for every $i \geqslant 2$, except for $i = 5$, where there is still a gap.

The lower bounds presented above hold unless the ETH fails, and the superexponential ones are inspired by a reduction of Marcin Pilipczuk [218].

17. **Ruling out FPT algorithms for Weighted Coloring on forests.**

Given a graph G, a *proper k-coloring* of G is a partition $c = (S_i)_{i\in[1,k]}$ of $V(G)$ into k stable sets $S_1, \ldots, S_k$. Given a weight function $w : V(G) \to \mathbb{R}^+$, the *weight of a color* S_i is defined as $w(i) = \max_{v\in S_i} w(v)$ and the *weight of a coloring* c as $w(c) = \sum_{i=1}^{k} w(i)$. Guan and Zhu [164] defined the *weighted chromatic number* of a pair (G, w), denoted by $\sigma(G, w)$, as the minimum weight of a proper coloring of G. For a positive integer r, they also defined $\sigma(G, w; r)$ as the minimum of $w(c)$ among all proper r-colorings c of G. The complexity of determining $\sigma(G, w)$ when G is a tree was open for almost 20 years, until Araújo et al. [44] recently proved that the problem cannot be solved in time $n^{o(\log n)}$ on n-vertex trees unless the ETH fails.

Our objective in [C23] is to provide hardness results for computing $\sigma(G, w)$ and $\sigma(G, w; r)$ when G is a tree or a forest, relying on complexity assumptions weaker than the ETH. Namely, we study the problem from the viewpoint of parameterized complexity, and we assume the weaker hypothesis FPT $\neq$ W[1]. Building on the techniques of Araújo et al. [44], we prove that when G is a forest, computing $\sigma(G, w)$ is W[1]-hard parameterized by the size of a largest connected component of G, and that computing $\sigma(G, w; r)$ is W[2]-hard parameterized by r. Our results rule out the existence of FPT algorithms for computing these invariants on trees or forests for many natural choices of the parameter.

18. **On the complexity of finding internally vertex-disjoint long directed paths**.

    A *subdivision* of a digraph F is a digraph obtained from F by replacing each arc (u, v) of F by a directed (u, v)-path. In [S36] we are interested in the (parameterized) complexity of several problems consisting in deciding whether a given digraph contains as a subdigraph a subdivision of a *spindle*, defined as follows. For k positive integers $\ell_1, \ldots, \ell_k$, an $(\ell_1, \ldots, \ell_k)$-*spindle* is the digraph containing k paths $P_1, \ldots, P_k$ from a vertex u to a vertex v, such that $|E(P_i)| = \ell_i$ for $1 \leqslant i \leqslant k$ and $V(P_i) \cap V(P_j) = \{u, v\}$ for $1 \leqslant i \neq j \leqslant k$. If $\ell_i = \ell$ for $1 \leqslant i \leqslant k$, an $(\ell_1, \ldots, \ell_k)$-spindle is also called a $(k \times \ell)$-*spindle*. See Figure 4.5 for an example.

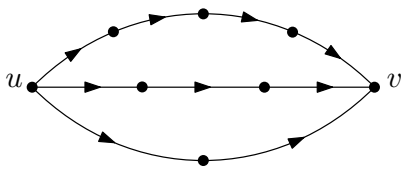

Figure 4.5: A $(4, 3, 2)$-spindle. This digraph contains a subdivision of a (3×2)-spindle, but not of a (3×3)-spindle.

    The problems we consider generalize both the Maximum Flow and Longest Path problems. In particular, we obtain the following complexity dichotomy: for a fixed $\ell \geqslant 1$, finding the largest k such that an input digraph G contains a subdivision of a $(k \times \ell)$-spindle is polynomial-time solvable if $\ell \leqslant 3$, and NP-hard otherwise.

    We place special emphasis on finding spindles with exactly two paths, which have recently attracted some interest in the literature [51, 88, 183]. We present FPT algorithms that are asymptotically optimal under the ETH. These algorithms are based on the technique of representative families in matroids introduced by Fomin et al. [138], and use also the color-coding technique of Alon et al. [42] as a subroutine. Finally, we study the case where the input graph is acyclic, and present several algorithmic and hardness results.

4.2 Kernelization

These are the some of the contributions that I have made in the last years to the field of kernelization. A significant part of my research has been devoted to find *linear* kernels for families of problems that are as general as possible, specially on sparse graph classes.

1. **Linear kernels via protrusion decompositions**.

    As mentioned above when dealing with FPT algorithms, we presented in [J15] an algorithm to compute a so-called *protrusion decomposition* of graphs satisfying certain conditions. We showed that any parameterized graph problem (with parameter k) that has *finite integer index* and such that positive instances have a treewidth-modulator of size $\mathcal{O}(k)$ admits a linear kernel on the class of H-topological-minor-

free graphs, for any fixed graph H. This result partially extended previous relevant meta-theorems on the existence of linear kernels on graphs of bounded genus [63] and H-minor-free graphs [140], and it is the currently most general result the existence of linear kernels on sparse graphs.

Full details of this contribution are provided in Chapter 5.

2. **Explicit linear kernels for domination problems on planar graphs**.

   By the celebrated meta-theorem of Bodlaender et al. [63], it follows that there *exists* a linear kernel for a vast family of problems on graphs of bounded genus. Nevertheless, it is not clear how such a kernel can be effectively *constructed*, and how to obtain explicit reduction rules with reasonably small constants. Thus, it makes sense to provide constructive linear kernels on planar graphs with explicit constants for particular problems, even if their *existence* is already known. In this direction, we have obtained the following results:

   - A *total dominating set* of a graph $G = (V, E)$ is a subset $D \subseteq V$ such that every vertex in V is adjacent to some vertex in D. Finding a total dominating set of minimum size is NP-hard on planar graphs and $W[2]$-complete on general graphs when parameterized by the solution size. Following the approach of Alber et al. [41] using region decompositions, we provided in [S38] an explicit linear kernel for Total Dominating Set on planar graphs, of size at most $410k$.
   - In the Red-Blue Dominating Set problem, we are given a bipartite graph $G = (V_B \cup V_R, E)$ and an integer k, and asked whether G has a subset $D \subseteq V_B$ of at most k "blue" vertices such that each "red" vertex from V_R is adjacent to a vertex in D. Following again the approach of Alber et al. [41], we provided in [J12] the first explicit linear kernel for this problem on planar graphs, of size at most $43k$.

   The two results presented above complement several known constructive linear kernels on planar graphs for other domination problems such as Dominating Set, Edge Dominating Set, Efficient Dominating Set, or Connected Dominating Set.

3. **Explicit linear kernels via dynamic programming**.

   Following the same research line discussed in the previous item, it has been already mentioned that in the last decade several algorithmic meta-theorems have appeared (Bodlaender et al. [63], Fomin et al. [140], Kim et al. [J15]) guaranteeing the *existence* of linear kernels on sparse graphs for problems satisfying some generic conditions. The drawback of such general results is that it is usually not clear how to derive from them *constructive* kernels with reasonably low *explicit* constants. To fill this gap, we present in [J11] a framework to obtain explicit linear kernels for some families of problems whose solutions can be certified by a subset of *vertices*.

   More precisely, we make a step toward a fully constructive meta-kernelization theory on sparse graphs. Our approach is based on a more explicit protrusion replacement

machinery that, instead of expressibility in CMSO logic, uses dynamic programming, which allows us to find an explicit upper bound on the size of the derived kernels.

Loosely speaking, the framework that we present can be summarized as follows. First of all, we propose a general definition of a problem encoding for the tables of dynamic programming when solving parameterized problems on graphs of bounded treewidth. Under this setting, we provide general conditions on whether such an encoding can yield a protrusion replacer. While our framework can also be seen as a possible formalization of dynamic programming, our purpose is to use it for constructing protrusion replacement algorithms and linear kernels whose size is explicitly determined.

In order to obtain an explicit linear kernel for a problem Π, the main ingredient is to prove that when solving Π on graphs of bounded treewidth via dynamic programming, we can use tables such that the maximum difference between all the values that need to be stored is bounded by a function of the treewidth. For this, we prove that when the input graph excludes a fixed graph H as a (topological) minor, this condition is sufficient for constructing an explicit protrusion replacer algorithm, i.e., a polynomial-time algorithm that replaces a large protrusion with an equivalent one whose size can be bounded by an *explicit* constant. Such a protrusion replacer can then be used, for instance, whenever it is possible to compute a linear protrusion decomposition of the input graph (that is, an algorithm that partitions the graph into a part of size linear in $\mathcal{O}(k)$ and a set of $\mathcal{O}(k)$ protrusions). As there is a wealth of results for constructing such decompositions [63, 137, 140, J15], we can use them as a starting point and, by applying dynamic programming, obtain an explicit linear kernel for Π.

We demonstrate the usefulness of our techniques by providing the first explicit linear kernels for r-Dominating Set and r-Scattered Set on apex-minor-free graphs, and for Planar-$\mathcal{F}$-Deletion on graphs excluding a fixed (topological) minor in the case where all the graphs in $\mathcal{F}$ are connected.

Full details of this contribution are provided in Chapter 6.

In a subsequent work [S37], we enhance our framework to deal with *packing* problems, that is, problems whose solutions can be certified by collections of *subgraphs* of the input graph satisfying certain properties. $\mathcal{F}$-Packing is a typical example: for a family $\mathcal{F}$ of connected graphs that we assume to contain at least one planar graph, the task is to decide whether a graph G contains k vertex-disjoint subgraphs such that each of them contains a graph in $\mathcal{F}$ as a minor. We provide in [S37] explicit linear kernels on sparse graphs for the following two orthogonal generalizations of $\mathcal{F}$-Packing: for an integer $\ell \geqslant 1$, one aims at finding either minor-models that are pairwise at distance at least ℓ in G (ℓ-$\mathcal{F}$-Packing), or such that each vertex in G belongs to at most ℓ minors-models ($\mathcal{F}$-Packing with ℓ-Membership). Finally, we also provide linear kernels for the versions of these problems where one wants to pack *subgraphs* instead of minors.

4. **Improved kernels for** Signed Max Cut **parameterized above lower bound on** (r,ℓ)**-graphs**.

   A graph G is *signed* if each edge is assigned "+" or "−". A signed graph is *balanced* if there is a bipartition of its vertex set such that an edge has sign "−" if and only if its endpoints are in different parts. The Edwards-Erdős bound states that every graph with n vertices and m edges has a balanced subgraph with at least $\frac{m}{2}+\frac{n-1}{4}$ edges [118,119]. In the Signed Max Cut Above Tight Lower Bound (Signed Max Cut ATLB) problem, given a signed graph G and a parameter k, the question is whether G has a balanced subgraph with at least $\frac{m}{2}+\frac{n-1}{4}+\frac{k}{4}$ edges. This problem generalizes Max Cut Above Tight Lower Bound, for which a kernel with $\mathcal{O}(k^5)$ vertices was given by Crowston et al. [93]. Crowston et al. [92] improved this result by providing a kernel with $\mathcal{O}(k^3)$ vertices for the more general Signed Max Cut ATLB problem.

   In [J10] we are interested in improving the size of the kernels for Signed Max Cut ATLB on restricted graph classes for which the problem remains hard. For two integers $r,\ell \geqslant 0$, a graph G is an (r,ℓ)*-graph* if $V(G)$ can be partitioned into r independent sets and ℓ cliques. Building on the techniques of Crowston et al. [92], for any $r,\ell \geqslant 0$ we provide a kernel with $\mathcal{O}((r+\ell)k^2)$ vertices on (r,ℓ)-graphs, and a simple linear kernel on subclasses of split graphs for which we prove that the problem is still NP-hard.

5. **How much does a treedepth modulator help to obtain polynomial kernels beyond sparse graphs?**

   In the last years, kernelization with structural parameters has been an active area of research within the field of parameterized complexity. As a relevant example, Gajarskỳ et al. [147] proved that every graph problem satisfying a property called finite integer index (cf. Chapter 5 for the formal definition) admits a linear kernel on graphs of bounded expansion and an almost linear kernel on nowhere dense graphs, parameterized by the size of a c-treedepth modulator, which is a vertex set whose removal results in a graph of treedepth[6] at most c, where $c \geqslant 1$ is a fixed integer. The authors left as further research to investigate this parameter on general graphs, and in particular to find problems that, while admitting polynomial kernels on sparse graphs, behave differently on general graphs.

   In [C28] we answer this question by finding two natural such problems: we prove that Vertex Cover admits a polynomial kernel on general graphs for any integer $c \geqslant 1$, and that Dominating Set does *not* for any integer $c \geqslant 2$ even on degenerate graphs, unless NP $\subseteq$ coNP/poly. For the positive result, we build on the techniques of Jansen and Bodlaender [171], and for the negative result we use a polynomial parameter transformation for $c \geqslant 3$ and an or-cross-composition for $c = 2$. As existing results imply that Dominating Set admits a polynomial kernel on degenerate graphs for $c = 1$, our result provides a dichotomy about the existence of polynomial kernels for Dominating Set on degenerate graphs with this parameter.

[6] Treedepth is a graph invariant that plays a crucial structural role on graphs of bounded expansion and nowhere dense graphs; see [212] for the definition.

4.3 Combinatorial results

We will now discuss some of the combinatorial results that I have obtained during the last years. I am particularly interested in the so-called *Erdős-Pósa property*. Before stating the results, we first say a few words about this property.

Typically, an Erdős-Pósa property reveals relations between covering and packing invariants in combinatorial structures. The origin of the study of such properties comes from the Erdős-Pósa theorem [120], stating that there is a function $f : \mathbb{N} \to \mathbb{N}$ such that for every $k \in \mathbb{N}$ and for every graph G, either G contains k vertex-disjoint cycles, or there is a set X of $f(k)$ vertices in G meeting all cycles of G. In particular, Erdős-Pósa proved this result for $f(k) = \mathcal{O}(k \cdot \log k)$.

An interesting line of research aims at extending Erdős-Pósa theorem for packings and coverings of more general graph structures. In this direction, we say that a graph class $\mathcal{G}$ *satisfies the Erdős-Pósa property* if there exists a function $f_{\mathcal{G}} : \mathbb{N} \to \mathbb{N}$ such that, for every graph G and every positive integer k, either G contains k mutually vertex-disjoint subgraphs, each isomorphic to a graph in $\mathcal{G}$, or it contains a set S of $f_{\mathcal{G}}(k)$ vertices meeting every subgraph of G that is isomorphic to a graph in $\mathcal{G}$. When this property holds for a class $\mathcal{G}$, we call the function $f_{\mathcal{G}}$ *the gap of the Erdős-Pósa property for the class* $\mathcal{G}$. In this sense, the classic Erdős-Pósa Theorem says that the class containing all cycles satisfies the Erdős-Pósa property with gap $\mathcal{O}(k \cdot \log k)$.

1. **Asymptotic enumeration of non-crossing partitions on surfaces**.

   As a key combinatorial ingredient for the algorithmic framework that we presented in [J21], we generalize in [J20] the notion of non-crossing partition on a disc to general surfaces with boundary. For this, we consider a surface Σ and introduce the number $C_\Sigma(n)$ of non-crossing partitions of a set of n points laying on the boundary of Σ. Our main result is an asymptotic estimate for $C_\Sigma(n)$. The proofs use bijective techniques arising from map enumeration, joint with the symbolic method and singularity analysis on generating functions (see [129] for an introduction to analytic combinatorics). An outcome of our results is that the exponential growth of $C_\Sigma(n)$ is the same as the one of the n-th Catalan number, i.e., does not change when we move from the case where Σ is a disc to general surfaces with boundary.

2. **An edge variant of the Erdős-Pósa property**.

   Recall that for every $r \in \mathbb{N}$, we denote by θ_r the r-pumpkin, that is, the multigraph with two vertices and r parallel edges; see Figure 4.1 for an illustration. Given a graph G, we say that a subgraph H of G is *a model of* θ_r *in* G if H contains θ_r as a contraction. We prove in [J19] that the following *edge variant* of the Erdős-Pósa property holds for every $r \geqslant 2$: if G is a graph and k is a positive integer, then either G contains a packing of k mutually edge-disjoint models of θ_r, or it contains a set S of $f_r(k)$ edges such that $G \setminus S$ has no θ_r-model, for both $f_r(k) = \mathcal{O}(k^2 r^3 \text{polylog} kr)$ and $f_r(k) = \mathcal{O}(k^4 r^2 \text{polylog} kr)$.

3. **Minors in graphs of large Θ_r-girth.**

   Still dealing with the r-pumpkin, let the *θ_r-girth* of a graph G be the minimum number of edges of a subgraph of G that can be contracted to θ_r. This notion generalizes the usual concept of girth which corresponds to the case $r = 2$. Kühn and Osthus [190] showed that graphs of sufficiently large minimum degree contain clique-minors whose order is an exponential function of their girth.

   In [J8] we extend this result for the case of θ_r-girth and we show that the minimum degree can be replaced by some connectivity measurement. As an application of our results, we prove that, for every fixed r, graphs excluding as a minor the disjoint union of k θ_r's have treewidth $\mathcal{O}(k \cdot \log k)$. This results has algorithmic applications to the Erdős-Pósa property for r-pumpkins, which we exploit in [C29].

4. **On the number of labeled graphs of bounded treewidth.**

   Given an integer $k > 0$, a *k-tree* is a graph that can be constructed starting from a $(k+1)$-clique and iteratively adding a vertex connected to k vertices that form a clique. They are natural extensions of trees, which correspond to 1-trees. A formula for the number of labeled k-trees on n vertices was first found by Beineke and Pippert [50], and alternative proofs were given by Moon [209] and Foata [132]. Namely, the number of n-vertex labeled k-trees is equal to

   $$\binom{n}{k}(kn - k^2 + 1)^{n-k-2}.$$

   Surprisingly, almost nothing was known about the number of *labeled* partial k-trees.

   Towards this end, let $T_{n,k}$ be the number of labeled graphs on n vertices and treewidth at most k (equivalently, the number of labeled partial k-trees). We show in [C25] that

   $$\left(c \cdot \frac{k\ 2^k n}{\log k}\right)^n 2^{-\frac{k(k+3)}{2}} k^{-2k-2} \leqslant T_{n,k} \leqslant \left(k\ 2^k n\right)^n 2^{-\frac{k(k+1)}{2}} k^{-k},$$

   for $k > 1$ and some explicit absolute constant $c > 0$. Disregarding lower-order terms, the gap between the lower and upper bound is of order $(\log k)^n$. The upper bound is a direct consequence of the well-known formula for the number of labeled k-trees, while the lower bound is obtained from an explicit construction. It follows from this construction that both bounds also apply to graphs of pathwidth and proper-pathwidth at most k.

   Full details of this contribution are provided in Chapter 7.

5. **Uniquely restricted matchings and edge colorings**.

   A matching in a graph is *uniquely restricted* if no other matching covers exactly the same set of vertices. This notion was defined by Golumbic et al. [157] and studied in a number of articles. Our contribution in [C26] is twofold.

   On the one hand, we provide approximation algorithms for computing a uniquely restricted matching of maximum size in some bipartite graphs. In particular, we achieve a ratio of $5/9$ for subcubic bipartite graphs, improving over a $1/2$-approximation algorithm proposed by Mishra [206].

   On the other hand, we study the *uniquely restricted chromatic index* of a graph, defined as the minimum number of uniquely restricted matchings into which its edge set can be partitioned. We provide tight upper bounds in terms of the maximum degree and characterize all extremal graphs. Our constructive proofs yield efficient algorithms to determine the corresponding edge colorings.

6. **A tight Erdős-Pósa function for wheel minors**.

   Let H be a fixed graph. An *H-model* $\mathcal{M}$ in a graph G is a collection $\{S_x \subseteq G : x \in V(H)\}$ of vertex-disjoint connected subgraphs of G such that S_x and S_y are linked by an edge in G for every edge $\{x, y\} \in E(H)$.

   Robertson and Seymour [223] proved that the Erdős-Pósa property holds for H-models if and only if H is planar. Their original bounding function was exponential. However, this has been significantly improved by recent breakthrough results of Chekuri and Chuzhoy [80], implying that there exist integers $a, b, c \geqslant 0$ such that for every planar graph H on h vertices, the Erdős-Pósa property holds for H-models with bounding function $f(k) = ah^b \cdot k \log^c(k+1)$. It follows from their proof that $c \leqslant 36$.

   This upper bound is remarkably close to being best possible: If H is planar with at least one cycle, then there is an $\Omega(k \log k)$ lower bound on bounding functions [120]. Closing this gap for different instantiations of H is an area of research that has attracted some interest in the last years.

   Let W_t denote the wheel on $t+1$ vertices, that is, the graph consisting of a cycle of length t and a vertex adjacent to all the vertices in the cycle. We prove in [S35] that for every integer $t \geqslant 3$ there is a constant $c = c(t)$ such that for every integer $k \geqslant 1$ and every graph G, either G has k vertex-disjoint subgraphs each containing W_t as minor, or there is a subset X of at most $ck \log k$ vertices such that $G - X$ has no W_t minor. By the above discussion, this is best possible, up to the value of c.

   Since the existence of a $\mathcal{O}(k \log k)$ bounding function for H-models is preserved under taking minors of H (see [S35] for a proof), our result generalizes several other results in the literature [120, 127, C29].

   Finally, we conjecture that the result remains true more generally if we replace W_t with any fixed planar graph H.

4.4 Problems arising from applications

During the last years I have also been interested in several problems motivated by practical applications, such as communications networks or bioinformatics. It is worth saying that, while the problems discussed below are indeed motivated by real practical applications, I do not claim that my algorithmic results are "easily" implementable in a computer. These results should be rather thought of as a theoretical analysis of these problems, in order to better understand its bottlenecks, and which aspects or their instances make them easier or harder.

1. **Placing regenerators in optical networks to satisfy multiple sets of requests**.

   The placement of regenerators in optical networks has become an active area of research during the last years. Given a set of lightpaths in a network G and a positive integer d, regenerators must be placed in such a way that in any lightpath there are no more than d hops without meeting a regenerator. We consider in [J17] a cost function given by the *total* number of regenerators placed at the nodes, which we believe to be a more accurate estimation of the real cost of the network than the number of *locations* considered by Flammini et al. [130].

   Furthermore, in our model we assume that we are given a finite set of p possible traffic patterns (each given by a set of lightpaths), and our objective is to place the minimum number of regenerators at the nodes so that each of the traffic patterns is satisfied. While this problem can be easily solved when $d = 1$ or $p = 1$, we prove that for any fixed $d, p \geqslant 2$ it does not admit a PTAS, even if G has maximum degree at most 3 and the lightpaths have length $\mathcal{O}(d)$.

   We complement this hardness result with a constant-factor approximation algorithm with ratio $\log(d \cdot p)$. We then study the case where G is a path, proving that the problem is polynomial-time solvable for two particular families of instances.

   Finally, we generalize our model in two natural directions, which allows us to capture the model of Flammini et al. [130] as a particular case, and we settle some questions that were left open therein.

2. **On the parameterized complexity of the** Edge Monitoring **problem**.

   Another application concerns the edge monitoring concept in sensor networks. Formally, in a graph $G = (V, E)$, a vertex $v \in V$ *monitors* an edge $\{u, u'\} \in E$ if $\{v, u\} \in E$ and $\{v, u'\} \in E$. Given an n-vertex graph $G = (V, E)$, in which each edge is contained in at least one triangle, and an integer k, the Edge Monitoring problem consists in finding a set $S \subseteq V$ of size at most k such that each edge of the graph is monitored by at least one element of S. This problem is known to be NP-hard, even on unit disk graphs [109].

   We prove in [J3] that Edge Monitoring is also W[2]-hard when parameterized by k. Using bidimensionality theory [103], we provide an FPT algorithm running in time $2^{\mathcal{O}(\sqrt{k} \cdot \log(\max_{e \in E} \omega(e)))} \cdot n$ for the weighted version of Edge Monitoring when the

input graph is restricted to be apex-minor-free. In particular, this algorithm applies to planar graphs, and where we additionally impose each edge e to be monitored at least $\omega(e)$ times, and the solution to be contained in a set of selected vertices.

3. **Efficient FPT algorithms for (strict) compatibility of unrooted phylogenetic trees**.

   In phylogenetics, a central problem is to infer the evolutionary relationships between a set of species X; these relationships are often depicted via a *phylogenetic tree* (that is, a tree having its leaves labeled bijectively by elements of X and without degree-2 nodes) called the "species tree". One common approach for reconstructing a species tree consists in first constructing several phylogenetic trees from primary data (e.g. DNA sequences originating from some species in X), and then constructing a single phylogenetic tree maximizing the "concordance" with the input trees. The obtained tree is our estimation of the species tree and, when the input trees are defined on overlapping, but not identical, sets of labels, is called "supertree".

   In [J5], we focus on two problems that are central when combining phylogenetic trees into a supertree: the compatibility and the strict compatibility problems for unrooted phylogenetic trees. These problems are strongly related, respectively, to the notions of "containing as a minor" and "containing as a topological minor" in the graph community. Both problems are known to be fixed-parameter tractable in the number of input trees k, by using their expressibility in Monadic Second Order Logic and a reduction to graphs of bounded treewidth [90].

   Motivated by the fact that the dependency on k of these algorithms is prohibitively large, we give the first explicit dynamic programming algorithms for solving these problems, both running in time $2^{\mathcal{O}(k^2)} \cdot n$, where n is the total size of the input.

4. **Parameterized complexity of reload cost problems**.

   Numerous network optimization problems can be modeled by edge-colored graphs. Wirth and Steffan introduced in [241] the concept of *reload cost*, which refers to the cost that arises in an edge-colored graph while traversing a vertex via two consecutive edges of different colors. The value of the reload cost depends on the colors of the traversed edges. Although the reload cost concept has many important applications in telecommunication networks, transportation networks, and energy distribution networks, it has surprisingly received attention only recently. Namely, recent works in the literature focused on numerous problems related to the reload cost concept: the minimum reload cost cycle cover problem [150], the problems of finding a path, trail or walk with minimum total reload cost between two given vertices [158], the problem of finding a spanning tree that minimizes the sum of reload costs of all paths between all pairs of vertices [151], various path, tour, and flow problems related to reload costs [43], the minimum changeover cost arborescence problem [149,160,162], and problems related to finding a proper edge coloring of the graph so that the total reload cost is minimized [161].

We have recently studied the following two reload cost problems:

- The Minimum Changeover Cost Arborescence (MinCCA) problem consists in finding an arborescence with a given root vertex such that the total changeover cost of the internal vertices is minimized. It has been recently proved by Gözüpek et al. [160] that the MinCCA problem when parameterized by the treewidth and the maximum degree of the input graph is FPT. In [J13] we present the following hardness results for MinCCA:

  - the problem is W[1]-hard when parameterized by the vertex cover number of the input graph, even on graphs of degeneracy at most 3. In particular, it is W[1]-hard parameterized by the treewidth of the input graph, which answers the main open problem in the work of Gözüpek et al. [160];
  - it is W[1]-hard on multigraphs parameterized by the tree-cutwidth of the input multigraph; and
  - it remains NP-hard on planar graphs even when restricted to instances with at most 6 colors and 0/1 symmetric costs, or when restricted to instances with at most 8 colors, maximum degree bounded by 4, and 0/1 symmetric costs.

- In [C24], we study the minimum diameter spanning tree problem under the reload cost model (Diameter-Tree for short), which is the problem introduced by Wirth and Steffan in their foundational article [241]. In this problem, given an undirected edge-colored graph G, reload costs on a path arise at a node where the path uses consecutive edges of different colors. The objective is to find a spanning tree of G of minimum diameter with respect to the reload costs. We initiate a systematic study of the parameterized complexity of the Diameter-Tree problem by considering the following parameters: the cost of a solution, and the treewidth and the maximum degree Δ of the input graph. We prove that Diameter-Tree is para-NP-hard for any combination of two of these three parameters, and that it is FPT parameterized by the three of them. We also prove that the problem can be solved in polynomial time on cactus graphs. This result is somehow surprising since we prove Diameter-Tree to be NP-hard on graphs of treewidth two, which is best possible as the problem can be trivially solved on forests.

  When the reload costs satisfy the triangle inequality, Wirth and Steffan [241] proved that the problem can be solved in polynomial time on graphs with $\Delta = 3$, and Galbiati [148] proved that it is NP-hard if $\Delta = 4$. Our results show, in particular, that without the requirement of the triangle inequality, the problem is NP-hard if $\Delta = 3$, which is also best possible. Finally, in the case where the reload costs are polynomially bounded by the size of the input graph, we prove that Diameter-Tree is in XP and W[1]-hard parameterized by the treewidth plus Δ.

  Full details of this contribution are provided in Chapter 8.

5. **Complexity dichotomies for the** Minimum $\mathcal{F}$-Overlay **problem**.

   For a (possibly infinite) fixed family of graphs $\mathcal{F}$, we say that a graph G *overlays* $\mathcal{F}$ on a hypergraph H if $V(H)$ is equal to $V(G)$ and the subgraph of G induced by every hyperedge of H contains some member of $\mathcal{F}$ as a spanning subgraph. While it is easy to see that the complete graph on $|V(H)|$ vertices overlays $\mathcal{F}$ on a hypergraph H whenever the problem admits a solution, the Minimum $\mathcal{F}$-Overlay problem asks for such a graph with the minimum number of edges. This problem allows to generalize some natural problems which may arise in practice. For instance, if the family $\mathcal{F}$ contains all connected graphs, then Minimum $\mathcal{F}$-Overlay corresponds to the Minimum Connectivity Inference problem (also known as Subset Interconnection Design problem) introduced for the low-resolution reconstruction of macro-molecular assembly in structural biology, or for the design of networks.

   Our main contribution in [C30] is a strong dichotomy result regarding the polynomial vs. NP-hard status with respect to the considered family $\mathcal{F}$. Roughly speaking, we show that the easy cases one can think of (e.g. when edgeless graphs of the right sizes are in $\mathcal{F}$, or if $\mathcal{F}$ contains only cliques) are the only families giving rise to a polynomial problem: all others are NP-complete.

   We then investigate the parameterized complexity of the problem and give similar sufficient conditions on $\mathcal{F}$ that give rise to W[1]-hard, W[2]-hard or FPT problems when the parameter is the size of the solution. This yields an FPT/W[1]-hard dichotomy for a relaxed problem, where every hyperedge of H must contain some member of $\mathcal{F}$ as a (non necessarily spanning) subgraph.

Chapter 5

Linear kernels and single-exponential algorithms via protrusion decompositions

In this chapter we present a linear-time algorithm to compute a decomposition scheme for graphs G that have a set $X \subseteq V(G)$, called a *treewidth-modulator*, such that the treewidth of $G - X$ is bounded by a constant. Our decomposition, called a *protrusion decomposition*, is the cornerstone in obtaining the following two main results.

Our first result is that any parameterized graph problem (with parameter k) that has *finite integer index* and such that Yes-instances have a treewidth-modulator of size $\mathcal{O}(k)$ admits a linear kernel on the class of H-topological-minor-free graphs, for any fixed graph H. This result partially extends previous meta-theorems on the existence of linear kernels on graphs of bounded genus and H-minor-free graphs.

Let $\mathcal{F}$ be a fixed finite family of graphs containing at least one planar graph. Given an n-vertex graph G and a non-negative integer k, Planar-$\mathcal{F}$-Deletion asks whether G has a set $X \subseteq V(G)$ such that $|X| \leqslant k$ and $G - X$ is H-minor-free for every $H \in \mathcal{F}$. As our second application, we present the first *single-exponential* algorithm to solve Planar-$\mathcal{F}$-Deletion. Namely, our algorithm runs in time $2^{\mathcal{O}(k)} \cdot n^2$, which is asymptotically optimal with respect to k. So far, single-exponential algorithms were only known for special cases of the family $\mathcal{F}$.

Keywords: parameterized complexity; algorithmic meta-theorems; sparse graphs; graph minors; hitting minors.

Contents

5.1 Introduction

This chapter contributes to the two main areas of parameterized complexity mentioned in Section 3.2, namely, kernels and fixed-parameter tractable (FPT) algorithms. In many cases, the key ingredient in order to solve a hard graph problem is to find an appropriate *decomposition* of the input graph, which allows to take advantage of the structure given by the graph class and/or the problem under study. We follow this paradigm and present a novel linear-time algorithm to compute a decomposition with nice properties for graphs G that have a set $X \subseteq V(G)$, called *t-treewidth-modulator*, such that the treewidth of $G - X$ is at most some constant $t - 1$. We then exploit this decomposition in two different ways: to *analyze* the size of kernels and to obtain *efficient* FPT algorithms. We would like to note that similar decompositions have already been (explicitly or implicitly) used for obtaining polynomial kernels [41, 63, 135, 140, 166].

Linear kernels. During the last decade, a plethora of results emerged on linear kernels for graph problems restricted to *sparse* graph classes. A celebrated result is the linear kernel for Dominating Set on planar graphs by Alber et al. [41]. This paper prompted an explosion of research papers on linear kernels on planar graphs, including Dominating Set [41, 82], Feedback Vertex Set [67], Cycle Packing [68], Induced Matching [176, 210], Full-Degree Spanning Tree [167], and Connected Dominating Set [200]. Guo and Niedermeier [166] designed a general framework and showed that problems that satisfy a certain "distance property" have linear kernels on planar graphs. This result was subsumed by that of Bodlaender et al. [63] who provided a meta-theorem for problems to have a linear kernel on graphs of bounded genus, a strictly larger class than planar graphs. Later Fomin et al. [140] extended these results for bidimensional problems to an even larger graph class, namely, H-minor-free and apex-minor-free graphs. (In all these works, the problems are parameterized by the *solution size*. See also [147, 152] for some recent meta-kernelization results considering structural parameters.) A common feature of these meta-theorems on sparse graphs is a *decomposition scheme* of the input graph that, loosely speaking, allows to deal with each part of the decomposition independently. For instance, the approach of [166], which is much inspired from [41], is to consider a so-called *region decomposition* of the input planar graph. The key point is that in an appropriately reduced Yes-instance, there are $\mathcal{O}(k)$ regions and each one has constant size, yielding the desired linear kernel. This idea was generalized in [63] to graphs on surfaces, where the role of regions is played by *protrusions*, which are graphs with small

treewidth and small boundary (see Section 5.2 for details). The resulting decomposition is called *protrusion decomposition*. A crucial point is that while the reduction rules of [41] are *problem-dependent*, those of [63] are *automated*, relying on a property called *finite integer index* (FII), which was introduced by Bodlaender and de Fluiter [70]. Loosely speaking (see Section 5.2), having FII guarantees that "large" protrusions of a graph can be replaced by "small" gadget graphs preserving equivalence of instances. This operation is usually called the *protrusion replacement rule*. FII is also of central importance to the approach of [140] on H-minor-free graphs. In fact, the idea of protrusion replacement (using a different terminology) can be traced back to the early 90's in the work of Arnborg et al. [46], and afterwards Bodlaender and de Fluiter [70] generalized the results in [46] to optimization problems. See also [39, 125, 126] for related work on this area.

Following the spirit of the aforementioned results, we present a novel algorithm to compute protrusion decompositions that allows us to obtain linear kernels on a larger class of sparse graphs, namely H-topological-minor-free graphs. Our algorithm takes as input a graph G and a t-treewidth-modulator $X \subseteq V(G)$, and outputs a set of vertices Y_0 containing X such that every connected component of $G - Y_0$ is a protrusion (see Section 5.3 for details).

When G is the input graph of a parameterized graph problem Π with parameter k, we call a protrusion decomposition of G *linear* if both $|Y_0|$ and the number of protrusions of $G - Y_0$ are $\mathcal{O}(k)$. If Π is such that YES-instances have a t-treewidth-modulator of size $\mathcal{O}(k)$ for some constant t (such problems are called *treewidth-bounding*, see Section 5.4), and G excludes some fixed graph H as a topological minor, we prove that the protrusion decomposition given by our algorithm is linear. If in addition Π has FII, then each protrusion can be replaced with a gadget of constant size, obtaining an equivalent instance of size $\mathcal{O}(k)$. Our first main result summarizes the above discussion.

Theorem 5.1 *Fix a graph H. Let Π be a parameterized graph problem on the class of H-topological-minor-free graphs that is treewidth-bounding and has finite integer index. Then Π admits a linear kernel.*

Consequences of Theorem 5.1. It turns out that a host of problems including TREEWIDTH-t VERTEX DELETION, CHORDAL VERTEX DELETION, INTERVAL VERTEX DELETION, EDGE DOMINATING SET, to name a few, satisfy the conditions of our theorem. Since for any fixed graph H, the class of H-topological-minor-free graphs strictly contains the class of H-minor-free graphs, our result is in fact an extension of the results of Fomin et al. [140]. As we discuss in Section 5.7, there is evidence that our result may reach the limit of sparse graph classes for which there exist meta-theorems about the existence of linear (or even uniform polynomial) kernels.

We also exemplify how our algorithm to obtain a linear protrusion decomposition can be applied to obtain *explicit* linear kernels, that is, kernels without using a generic protrusion replacement. This is shown by exhibiting a simple explicit linear kernel for the EDGE DOMINATING SET problem on H-topological-minor-free graphs. So far, all known linear kernels for EDGE DOMINATING SET on H-minor-free graphs [140] and H-topological-minor-free graphs (given by Theorem 5.1) relied on generic protrusion replacement.

Single-exponential algorithms. In order to prove Theorem 5.1, similarly to [63, 140, 166] our protrusion decomposition algorithm is only used to *analyze* the size of the resulting instance after having applied the protrusion reduction rule. In the second part of the chapter we show that our decomposition scheme can also be used to obtain *efficient* FPT algorithms. Before stating our second main result, let us motivate the problem that we study.

During the last decades, parameterized complexity theory has brought forth several algorithmic meta-theorems that imply that a wide range of problems are in FPT (see [188] for a survey). For instance, as mentioned in Chapter 3, Courcelle's theorem [90] states that every decision problem expressible in Monadic Second Order Logic can be solved in linear time when parameterized by the treewidth of the input graph. At the price of generality, such algorithmic meta-theorems may suffer from the fact that the function $f(k)$ is huge [144, 189] or non-explicit [90, 226]. Therefore, it has become a central task in parameterized complexity to provide FPT algorithms such that the behavior of the function $f(k)$ is *reasonable*; in other words, a function $f(k)$ that could lead to a practical algorithm.

Towards this goal, recall that we say that an FPT parameterized problem is solvable in *single-exponential* time if there exists an algorithm solving it in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. For instance, recent results have shown that broad families of problems admit (deterministic or randomized) single-exponential algorithms parameterized by treewidth [96, 111, J21]. On the other hand, single-exponential algorithms are unlikely to exist for certain parameterized problems [96, 199]. Parameterizing by the size of the desired solution, in the case of Vertex Cover the existence of a single-exponential algorithm has been known for a long time, but it took a while to witness the first (deterministic) single-exponential algorithm for Feedback Vertex Set, or equivalently Treewidth-One Vertex Deletion [100, 165].

Both Vertex Cover and Feedback Vertex Set can be seen as graph modification problems in order to attain a hereditary property, that is, a property closed under taking induced subgraphs. It is well-known that deciding whether at most k vertices can be deleted from a given graph in order to attain any non-trivial hereditary property is NP-complete [194]. The particular case where the property can be characterized by a finite set of forbidden induced subgraphs can be solved in single-exponential time when parameterizing by the number of modifications, even in the more general case where also edge deletions or additions are allowed [75]. If the family of forbidden induced subgraphs is infinite, no meta-theorem is known and not every problem is even FPT [195]. A natural question arises: can we carve out a larger class of hereditary properties for which the corresponding graph modification problem can be solved in single-exponential time?

A line of research emerged pursuing this question, which is much inspired by the Feedback Vertex Set problem. Interestingly, when the *infinite* family of forbidden *induced subgraphs* can also be captured by a *finite* set $\mathcal{F}$ of forbidden *minors* (namely, when the problem is closed under taking minors [227]), the $\mathcal{F}$-Deletion problem (namely, the problem of removing at most k vertices from an input graph to obtain a graph which is H-minor-free for every $H \in \mathcal{F}$) is in non-uniform[1] FPT by the seminal meta-theorem of

[1] A non-uniform FPT algorithm for a parameterized problem is a collection of algorithms, one for each value of the parameter k.

Robertson and Seymour [226][2].

Let $\mathcal{F}$ be a finite family of graphs containing at least one planar graph. The parameterized problem that we consider in the second part of this chapter is Planar-$\mathcal{F}$-Deletion, which is defined as follows:

Planar-$\mathcal{F}$-Deletion
Input: A graph G and a non-negative integer k.
Parameter: The integer k.
Question: Does G have a set $X \subseteq V(G)$ such that $\|X\| \leqslant k$ and $G - X$ is H-minor-free for every $H \in \mathcal{F}$?

Note that Vertex Cover and Feedback Vertex Set correspond to the special cases of $\mathcal{F} = \{K_2\}$ and $\mathcal{F} = \{K_3\}$, respectively. A recent work by Joret et al. [J14] handled the case $\mathcal{F} = \{\theta_c\}$ and achieved a single-exponential algorithm for Planar-θ_c-Deletion for any value of $c \geqslant 1$, where θ_c is the (multi)graph consisting of two vertices and c parallel edges between them. (Note that the cases $c = 1$ and $c = 2$ correspond to Vertex Cover and Feedback Vertex Set, respectively.) Kim et al. [182] obtained a single-exponential algorithm for $\mathcal{F} = \{K_4\}$, also known as Treewidth-Two Vertex Deletion. Related works of Philip et al. [217] and Cygan et al. [97] resolve the case $\mathcal{F} = \{K_3, T_2\}$, or equivalently Pathwidth-One Vertex Deletion, in single-exponential time.

The Planar-$\mathcal{F}$-Deletion problem was first stated by Fellows and Langston [124], who proposed a non-uniform (and non-constructive) $f(k) \cdot n^2$-time algorithm for some function $f(k)$, as well as a $f(k) \cdot n^3$-time algorithm for the general $\mathcal{F}$-Deletion problem, both relying on the meta-theorem of Robertson and Seymour [226]. Explicit bounds on the function $f(k)$ for Planar-$\mathcal{F}$-Deletion can be obtained via dynamic programming. Indeed, as the Yes-instances of Planar-$\mathcal{F}$-Deletion have treewidth $\mathcal{O}(k)$, using standard dynamic programming techniques on graphs of bounded treewidth (see for instance [56, J1]), it can be seen that Planar-$\mathcal{F}$-Deletion can be solved in time $f(k) \cdot n^2$ with $f(k) = 2^{2^{\mathcal{O}(k \log k)}}$. In an unpublished paper [136], Fomin et al. proposed a $2^{\mathcal{O}(k \log k)} \cdot n^2$-time algorithm for Planar-$\mathcal{F}$-Deletion, which is, up to our knowledge, the best known result. More recently, Fomin et al. [137] provided a $2^{\mathcal{O}(k)} \cdot n \log^2 n$-time algorithm for the Planar-Connected-$\mathcal{F}$-Deletion problem, which is the special case of Planar-$\mathcal{F}$-Deletion when *every* graph in the family $\mathcal{F}$ is *connected*. In this chapter, we get rid of the connectivity assumption, and we prove that the general Planar-$\mathcal{F}$-Deletion problem can be solved in single-exponential time. Namely, our second main result is the following.

Theorem 5.2 *The parameterized* Planar-$\mathcal{F}$-Deletion *problem can be solved in time* $2^{\mathcal{O}(k)} \cdot n^2$.

This result unifies, generalizes, and simplifies a number of results given in [83, 100, 137, 165, 182, J14]. Let us make a few considerations about the fact that the family $\mathcal{F}$ may contain

[2]It is worth noting that, in contrast to the removal of vertices, the problems corresponding to the operations of removing or contracting edges are not minor-closed (we provide a proof of this fact in Section 5.6.1), and therefore the result of Robertson and Seymour [226] cannot be applied to these modification problems.

disconnected graphs or not. Besides the fact that removing the connectivity constraint is an important theoretical step towards the general $\mathcal{F}$-Deletion problem, it turns out that many natural such families $\mathcal{F}$ do contain disconnected graphs. For instance, the disjoint union of g copies of K_5 (or $K_{3,3}$) is a minimal forbidden minor, or an *obstruction*, for the graphs of genus $g-1$ [47] (see also [208]). In particular, the (disconnected) graph made of two copies of K_5 is in the obstruction set of the graphs that can be embedded in the torus. In Section 5.6.2 we show that many natural obstruction sets also contain disconnected *planar* graphs.

It should also be noted that the function $2^{\mathcal{O}(k)}$ in Theorem 5.2 is best possible, assuming the Exponential Time Hypothesis (ETH); see Section 3.2 for the definition. Namely, it is known that unless the ETH fails, Vertex Cover cannot be solved in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ [131, Chapter 16]. It is noteworthy that when $\mathcal{F}$ does not contain any planar graph, up to our knowledge no single case is known to admit a single-exponential algorithm. For instance, we point out that Planar Vertex Deletion, which amounts to $\mathcal{F} = \{K_5, K_{3,3}\}$, is not known to admit a single-exponential parameterized algorithm after having been the focus of several articles [172,177,205]. The currently best known algorithm is by Jansen et al. [172] and runs in time $2^{\mathcal{O}(k \log k)} \cdot n$. Moreover, Pilipczuk [218] recently proved that the existence of an algorithm that follows the approach of [172] and solves Planar Vertex Deletion in time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$ would contradict the ETH.

Technical ingredients in the proof of Theorem 5.2. As mentioned above, when employing protrusion replacement, often the problem needs to have FII. Many problems enjoy this property, for example Treewidth-t Vertex Deletion or (Connected) Dominating Set, among others. Having FII makes the problem amenable to this powerful reduction rule, and essentially this was the basic ingredient of previous works such as [137, 182, J14]. In particular, when every graph in $\mathcal{F}$ is connected, the Planar-$\mathcal{F}$-Deletion problem has FII [63], and the single-exponential time algorithm of [137] heavily depends on this feature. However, if one aims at Planar-$\mathcal{F}$-Deletion without any connectivity restriction on the family $\mathcal{F}$, the requirement for FII seems to be a fundamental hurdle, as if $\mathcal{F}$ may contain disconnected graphs, then Planar-$\mathcal{F}$-Deletion does not have FII for some choices of $\mathcal{F}$.[3] We observe that the unpublished $2^{\mathcal{O}(k \log k)} \cdot n^2$-time algorithm of [136] applies to the general Planar-$\mathcal{F}$-Deletion problem (that is, $\mathcal{F}$ may contain some disconnected graph). The reason is that instead of relying on FII, they rather use tools from *annotated kernelization* [63].

To circumvent the situation of not having FII, our algorithm *does not use any reduction rule*, but instead relies on a series of branching steps. First of all, we apply the iterative compression technique (introduced by Reed et al. [221]) in order to reduce the Planar-$\mathcal{F}$-Deletion problem to its *disjoint* version. In the Disjoint Planar-$\mathcal{F}$-Deletion problem, given a graph G and an initial solution X of size k, the task is to decide whether G contains an alternative solution $\tilde{X}$ disjoint from X of size at most $k-1$. In our case, the assumption that $\mathcal{F}$ contains some planar graph is fundamental, as then $G - X$ has bounded treewidth [223]. Central to our single-exponential algorithm is our linear-time

[3] As we were not able to find a reference with a proof of this fact, for completeness we provide it in Section 5.6.3.

algorithm to compute a protrusion decomposition, in this case with the initial solution X as treewidth-modulator. A first step to this end is to use the aforementioned algorithm (the one used for the analysis of linear kernel) and compute a superset Y_0 of X such that each component of $G - Y_0$, together with its neighborhood in Y_0, forms a protrusion. But for the resulting protrusion decomposition to be linear, it turns out that we first need to guess the intersection of the alternative solution with the set Y_0. Once we have the desired linear protrusion decomposition, instead of applying protrusion replacement, we simply identify a set of $\mathcal{O}(k)$ vertices among which the alternative solution has to live, if it exists. In the whole process described above, there are three branching steps: the first one is inherent to the iterative compression paradigm, the second one is required to compute a linear protrusion-decomposition, and finally the last one enables us to guess the set of vertices containing the solution. It can be proved that each branching step is compatible with single-exponential time, which yields the desired result. Finally, it is worth mentioning that our algorithm is *fully constructive* (cf. Section 5.5.3 for details).

Organization of the chapter. In Section 5.2 we state the main definitions of the protrusion machinery developed in [63, 140] that we need for our purposes. We then exhibit our protrusion decomposition algorithm in Section 5.3. As our first application of our decomposition result, we prove Theorem 5.1 in Section 5.4. In Section 5.5 we prove Theorem 5.2. Finally, in Section 5.7 we conclude with some closing remarks.

5.2 Protrusions, t-boundaried graphs, and finite integer index

In this section we provide some definitions that will be used in this chapter; some of them will also be used in Chapter 6. For convenience, we assume in this chapter that $V(G)$ is a totally ordered set. Since we will mainly be concerned with sparse graphs we let $|G|$ denote the number of vertices in the graph G. We denote by $\omega(G)$ the size of the largest complete subgraph of G and by $\#\omega(G)$ the *number* of complete subgraphs (not necessarily maximal ones).

We proceed to restate the main definitions of the protrusion machinery developed in [63, 140]. Given a graph $G = (V, E)$ and a set $W \subseteq V$, we define $\partial_G(W)$ as the set of vertices in W that have a neighbor in $V \setminus W$. For a set $W \subseteq V$ the neighborhood of W is $N^G(W) = \partial_G(V \setminus W)$. Superscripts and subscripts are omitted when it is clear which graph is being referred to.

Definition 5.1 (t-protrusion [63]) *Given a graph G, a set $W \subseteq V(G)$ is a* t-protrusion *of G if $|\partial_G(W)| \leqslant t$ and $\mathsf{tw}(G[W]) \leqslant t - 1$.[4] If W is a t-protrusion, the vertex set $W' = W \setminus \partial_G(W)$ is the* restricted protrusion of W. *We call $\partial_G(W)$ the* boundary *and $|W|$ the* size *of the t-protrusion W of G. Given a restricted t-protrusion W', we denote its* extended protrusion *by $W'^{+} = W' \cup N(W')$.*

Note that if W' is the restricted protrusion of W, then $W'^{+} = W$. A rough outline of a protrusion is depicted in Figure 5.1.

[4] In [63], $\mathsf{tw}(G[W]) \leqslant t$, but we want the size of the bags to be at most t.

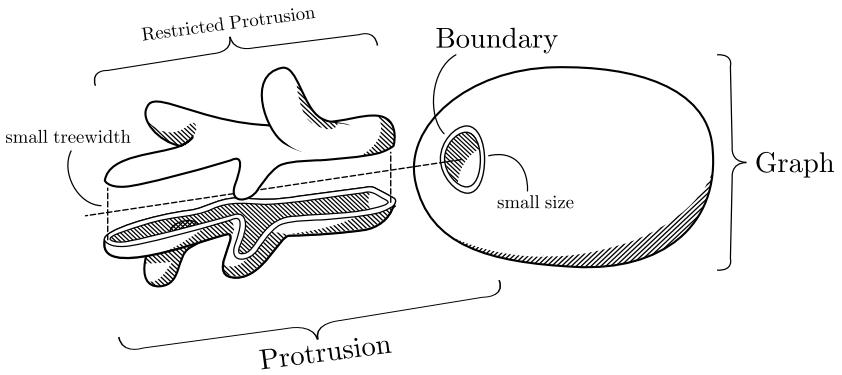

Figure 5.1: Basic anatomy of a protrusion.

A *t-boundaried graph* is a graph $G = (V, E)$ with a set $\mathbf{bd}(G)$ (called the *boundary*[5] or the *terminals* of G) of t distinguished vertices labeled 1 through t. Let $\mathcal{G}_t$ denote the class of t-boundaried graphs, with graphs from $\mathcal{G}$. If $W \subseteq V$ is an r-protrusion in G, then we let G_W be the r-boundaried graph $G[W]$ with boundary $\partial_G(W)$, where the vertices of $\partial_G(W)$ are assigned labels 1 through r according to their order in G.

Definition 5.2 (Gluing and ungluing) *For t-boundaried graphs G_1 and G_2, we let $G_1 \oplus G_2$ denote the graph obtained by taking the disjoint union of G_1 and G_2 and identifying each vertex in $\mathbf{bd}(G_1)$ with the vertex in $\mathbf{bd}(G_2)$ with the same label. This operation is called* gluing*.*

Let $G_1 \subseteq G$ with a boundary B of size t. The operation of ungluing *G_1 from G creates the t-boundaried graph $G \ominus_B G_1 := G - (V(G_1) \setminus B)$ with boundary B. The vertices of $\mathbf{bd}(G \ominus_B G_1)$ are assigned labels 1 through t according to their order in the graph G.*

Note that the gluing operation entails taking the union of edges both of whose endpoints are in the boundary with the deletion of multiple edges to keep the graph simple. The ungluing operation preserves the boundary (both the vertices and the edges).

Definition 5.3 (Replacement) *Let $G = (V, E)$ be a graph with a t-protrusion W; let G_W denote the graph $G[W]$ with boundary $\mathbf{bd}(G_W) = \partial_G(W)$; and finally, let G_1 be a t-boundaried graph. Then* replacing *G_W by G_1 corresponds to the operation $(G \ominus G_W) \oplus G_1$.*

Definition 5.4 (Protrusion decomposition) *An* (α, t)-protrusion decomposition *of a graph G is a partition $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $V(G)$ such that:*

1. *for every $1 \leqslant i \leqslant \ell$, $N(Y_i) \subseteq Y_0$;*
2. *$\max\{\ell, |Y_0|\} \leqslant \alpha$;*
3. *for every $1 \leqslant i \leqslant \ell$, $Y_i \cup N_{Y_0}(Y_i)$ is a t-protrusion of G.*

The set Y_0 is called the separating part *of $\mathcal{P}$.*

[5] Usually denoted by $\partial(G)$, but this collides with our usage of ∂.

Hereafter, the value of t will be fixed to some constant. When G is the input of a parameterized graph problem with parameter k, we say that an (α, t)-protrusion decomposition of G is *linear* (resp. *quadratic*) whenever $\alpha = \mathcal{O}(k)$ (resp. $\alpha = \mathcal{O}(k^2)$).

We now restate the definition of one of the most important notions used in this chapter.

Definition 5.5 (Finite integer index (FII) [70]) *Let $\Pi_{\mathcal{G}}$ be a parameterized graph problem restricted to a class $\mathcal{G}$ and let G_1, G_2 be two t-boundaried graphs in $\mathcal{G}_t$. We say that $G_1 \equiv_{\Pi,t} G_2$ if there exists a constant $\Delta_{\Pi,t}(G_1, G_2)$ (that depends on Π, t, and the ordered pair (G_1, G_2)) such that for all t-boundaried graphs G_3 and for all k:*

1. *$G_1 \oplus G_3 \in \mathcal{G}$ if and only if $G_2 \oplus G_3 \in \mathcal{G}$;*
2. *$(G_1 \oplus G_3, k) \in \Pi$ if and only if $(G_2 \oplus G_3, k + \Delta_{\Pi,t}(G_1, G_2)) \in \Pi$.*

We say that the problem $\Pi_{\mathcal{G}}$ has finite integer index *in the class $\mathcal{G}$ if for every integer t, the equivalence relation $\equiv_{\Pi,t}$ has finite index (that is, it has a finite number of equivalence classes). In the case that $(G_1 \oplus G, k) \notin \Pi$ or $G_1 \oplus G \notin \mathcal{G}$ for* all *$G \in \mathcal{G}_t$, we set $\Delta_{\Pi,t}(G_1, G_2) = 0$. Note that $\Delta_{\Pi,t}(G_1, G_2) = -\Delta_{\Pi,t}(G_2, G_1)$.*

We would like to note that the definition of finite integer index given in Definition 5.5 differs from the definition given in [63,70], where the first condition in the definition of the equivalence relation $\equiv_{\Pi,t}$ is not required. As in [140], we adopt the above definition for notational simplicity, due to the following reason. Assume that the equivalence relation defined only by the second condition in Definition 5.5 has finite index. Assume furthermore that the membership in the graph class $\mathcal{G}$ can be expressed in MSO logic. Then by Myhill-Nerode's theorem[6], it follows that the equivalence relation $\equiv_{\Pi,t}$ has finite index as well (see for instance [115,131,214]). As the membership in the class of graphs that can exclude a fixed graph H as a topological minor can be expressed in MSO logic (see Section 5.6.4 for the precise formula), it will be simpler to already incorporate the first condition in Definition 5.5.

If a parameterized problem has finite integer index then its instances can be reduced by "replacing protrusions". The technique of replacing protrusions hinges on the fact that each protrusion of "large" size can be replaced by a "small" gadget from the same equivalence class as the protrusion, which consequently behaves similarly with respect to the problem at hand. If G_1 is replaced by a gadget G_2, then the parameter k in the problem changes by $\Delta_{\Pi,t}(G_1, G_2)$. What is not immediately clear is that given that a problem Π has finite integer index, how does one show that there always exists a set of representatives for which the parameter is guaranteed not to *increase*. The next lemma shows that this is indeed the case. The ideas of the proof are implicit in [70].

Lemma 5.1 *Let Π be a parameterized graph problem that has finite integer index in a graph class $\mathcal{G}$. Then for every fixed t, there exists a finite set $\mathcal{R}_t$ of t-boundaried graphs such that for each t-boundaried graph $G \in \mathcal{G}_t$ there exists a t-boundaried graph $G' \in \mathcal{R}_t$ such that $G \equiv_{\Pi,t} G'$ and $\Delta_{\Pi,t}(G, G') \geqslant 0$.*

[6] A thorough historical study of the family of Myhill-Nerode's theorems can be found in [238].

Proof: The set $\mathcal{R}_t$ consists of one element from each equivalence class of $\equiv_{\Pi,t}$. Since Π has finite integer index, the set $\mathcal{R}_t$ is finite. Therefore we only have to show that there exist representatives that satisfy the requirement in the statement of the lemma.

To this end, fix any equivalence class $\mathcal{G}'_t \in \mathcal{G}_t/\equiv_{\Pi,t}$. First consider the case where there exists $G_1 \in \mathcal{G}'_t$ such that for all $G \in \mathcal{G}_t$, either $G_1 \oplus G \notin \mathcal{G}$ or for all $k \in \mathbb{N}_0$, $(G_1 \oplus G, k) \notin \Pi$. Since $\mathcal{G}'_t$ is an equivalence class, this means that at least one of these two conditions holds for *every* graph $G \in \mathcal{G}'_t$. Thus $\Delta_{\Pi,t}(G_1, G_2) = 0$ for all t-boundaried graphs $G_1, G_2 \in \mathcal{G}'_t$ and we can simply take a graph of smallest size from $\mathcal{G}'_t$ as representative.

We can now assume that for the chosen $\mathcal{G}'_t$ it holds that there exists a t-boundaried graph $G \in \mathcal{G}_t$ such that for *all* $G_1 \in \mathcal{G}'_t$ we have that $G_1 \oplus G \in \mathcal{G}$ and, for some $k \in \mathbb{N}$, $(G_1 \oplus G, k) \in \Pi_{\mathcal{G}}$. Consider the following binary relation $\preceq$ over $\mathcal{G}'_t$: for all $G_1, G_2 \in \mathcal{G}'_t$,

$$G_1 \preceq G_2 \Leftrightarrow \Delta_{\Pi,t}(G_1, G_2) \geqslant 0.$$

As $\Delta_{\Pi,t}(G, G) = 0$ for all $G \in \mathcal{G}_t$, it immediately follows that the relation $\preceq$ is reflexive. Furthermore, the relation is *total* as every graph is comparable to every other graph from the same equivalence class.

We next show that the relation $\preceq$ is also transitive, making it a total quasi-order. Let $G_1, G_2, G_3 \in \mathcal{G}'_t$ be such that $G_1 \preceq G_2$ and $G_2 \preceq G_3$. This is equivalent to saying that $c_{12} = \Delta_{\Pi,t}(G_1, G_2) \geqslant 0$ and $c_{23} = \Delta_{\Pi,t}(G_2, G_3) \geqslant 0$. For every $G \in \mathcal{G}_t$ such that $G_1 \oplus G \in \mathcal{G}$ and $(G_1 \oplus G, k) \in \Pi$ for some $k \in \mathbb{N}$, we have

$$\begin{aligned}(G_1 \oplus G, k) \in \Pi &\Leftrightarrow (G_2 \oplus G, k + c_{12}) \in \Pi \\ &\Leftrightarrow (G_3 \oplus G, k + c_{12} + c_{23}) \in \Pi.\end{aligned}$$

By definition, $\Delta_{\Pi,t}(G_1, G_3) = c_{12} + c_{23} \geqslant 0$ and hence $G_1 \preceq G_3$. We conclude that $\preceq$ is transitive and therefore a total quasi-order.

We now show that the class $\mathcal{G}'_t$ can be partitioned into layers that can be linearly ordered. We will pick our representative for the class $\mathcal{G}'_t$ from the first layer in this ordering. To do this, we define the following equivalence relation over $\mathcal{G}'_t$. For all $G_1, G_2 \in \mathcal{G}'_t$, define

$$\begin{aligned}G_1 \equiv G_2 &\Leftrightarrow G_1 \preceq G_2 \text{ and } G_2 \preceq G_1 \\ &\Leftrightarrow \Delta_{\Pi,t}(G_1, G_2) = 0.\end{aligned}$$

Now, the equivalence classes $\mathcal{G}'_t/\equiv$ can be linearly ordered as follows. Fix a graph $G \in \mathcal{G}_t$ such that for any $G_1 \in \mathcal{G}'_t$ we have that $G_1 \oplus G \in \mathcal{G}$ and $(G_1 \oplus G, k) \in \Pi$ for some $k \in \mathbb{N}$, this graph must exist since we handled equivalence classes of $\mathcal{G}_t/\equiv_{\Pi,t}$ which do not have such a graph in the first part of the proof. Consider the function $\Phi_G \colon \mathcal{G}'_t/\equiv \to \mathbb{N}_0$ defined via

$$\Phi_G([G']) = \min\left\{k \in \mathbb{N} \mid (G' \oplus G, k) \in \Pi\right\}.$$

Observe that $\Phi_G([G_2]) = \Phi_G([G_1]) + \Delta_{\Pi,t}(G_1, G_2)$ for all $G_1, G_2 \in \mathcal{G}'_t$ and, in particular, that

$$\Phi_G([G_1]) = \Phi_G([G_2]) \Leftrightarrow G_1 \equiv G_2.$$

Thus Φ_G induces a linear order on $\mathcal{G}'_t/\equiv$. Moreover, since $\Phi_G(\cdot) \geqslant 0$, there exists a class $[G^*]$ in $\mathcal{G}'_t/\equiv$ that is a minimum element in the order induced by Φ_G. For any t-boundaried

graph $G \in [G^*]$, it then follows that for all $G_1 \in \mathcal{G}'_t$, $\Delta_{\Pi,t}(G, G_1) \geqslant 0$. The representative of $\mathcal{G}'_t$ in $\mathcal{R}_t$ is an arbitrary t-boundaried graph $G' \in [G^*]$ of smallest size. This proves the lemma. □

Definition 5.6 (Protrusion limit) *For a parameterized graph problem* Π *that has finite integer index in the class* $\mathcal{G}$*, let* $\mathcal{R}_t$ *denote the set of representatives of the equivalence classes of* $\equiv_{\Pi,t}$ *satisfying the requirement stated in Lemma 5.1. The* protrusion limit *of* $\Pi_{\mathcal{G}}$ *is defined as* $\rho_{\Pi_{\mathcal{G}}}(t) = \max_{G \in \mathcal{R}_t} |V(G)|$*. We drop the subscript when it is clear which graph problem is being referred to. We also define* $\rho'(t) := \rho(2t)$*.*

The next two lemmas deal with finding protrusions in graphs. The first of these guarantees that whenever there exists a "large enough" protrusion, there exists a protrusion that is large but of size bounded by a *constant* (that depends on the problem and the boundary size). As we shall see later, the fact that we deal with protrusions of constant size enables us to efficiently test which representative to replace them by, assuming that we have the set of representatives. For completeness, we provide the proof of the following lemma.

Lemma 5.2 (Bodlaender et al. [63]) *Let* Π *be a parameterized graph problem with finite integer index in* $\mathcal{G}$ *and let* $t \in \mathbb{N}$ *be a constant. For a graph* $G \in \mathcal{G}$*, if one is given a* t*-protrusion* $X \subseteq V(G)$ *such that* $\rho'_{\Pi_{\mathcal{G}}}(t) < |X|$*, then one can, in time* $\mathcal{O}(|X|)$*, find a* $2t$*-protrusion* W *such that* $\rho'_{\Pi_{\mathcal{G}}}(t) < |W| \leqslant 2 \cdot \rho'_{\Pi_{\mathcal{G}}}(t)$*.*

Proof: Let $(T, \mathcal{X})$ be a nice tree-decomposition for $G[X]$ of width $t-1$. Root T at an arbitrary node. Let u be the *lowest* node of T such that if W is the set of vertices in the bags associated with the nodes in the subtree T_u rooted at u, then $|W| > \rho'_{\Pi_{\mathcal{G}}}(t)$. Clearly W is a $2t$-protrusion with boundary $X_u \cup \partial_G(X)$, where $X_u \subseteq V(G)$ is the bag associated with the node u of T. By the choice of u, it is clear that u cannot be a forget node. If u is an introduce node with child v, then the number of vertices in the bags associated with the nodes of T_v must be exactly $\rho'_{\Pi_{\mathcal{G}}}(t)$. Since u introduces exactly one additional vertex of G, we have $|W| = \rho'_{\Pi_{\mathcal{G}}}(t) + 1$. Finally consider the case when u is a join node with children y, z. Then the bags associated with these nodes X_u, X_y, X_z are identical and since

$$|\bigcup_{j \in V(T_y)} X_j| < \rho'_{\Pi_{\mathcal{G}}}(t) \qquad \text{and} \qquad |\bigcup_{j \in V(T_z)} X_j| < \rho'_{\Pi_{\mathcal{G}}}(t),$$

we have that $W = \bigcup_{j \in V(T_y)} X_j \cup \bigcup_{j \in V(T_z)} X_j$ has size at most $2 \cdot \rho'_{\Pi_{\mathcal{G}}}(t)$.

Computing a nice tree-decomposition $(T, \mathcal{X})$ of $G[X]$ takes time $2^{\mathcal{O}(t^3)} \cdot |X|$ [58] and the time required to compute a $2t$-protrusion from T is $\mathcal{O}(|X|)$. Since t is a constant, the total time taken is $\mathcal{O}(|X|)$. □

For a fixed t, the protrusion W is of *constant* size but, in the reduction rule to be described, would be replaced by a representative of smaller size, namely at most $\rho'_{\Pi_{\mathcal{G}}}(t) = \rho_{\Pi_{\mathcal{G}}}(2t)$. This means that each time the reduction rule is applied, the size of the graph strictly decreases and, by Lemma 5.1, the parameter does not increase. The reduction rule can therefore be applied at most n times, where n is the number of vertices in the input graph.

As we shall see later, each application of the reduction rule takes time polynomial in n, assuming that we are given the set of representatives. Therefore, in polynomial time, we would obtain an instance in which every t-protrusion has size at most $\rho_{\Pi_{\mathcal{G}}}(2t)$. This trick is described in [63] but is stated here for the sake of completeness.

The next lemma describes how to find a t-protrusion of maximum size.

Lemma 5.3 (Finding maximum sized protrusions) *Let t be a constant. Given an n-vertex graph G, a t-protrusion of G with the maximum number of vertices can be found in time $\mathcal{O}(n^{t+1})$.*

Proof: For a vertex set $B \subseteq V(G)$ of size at most t, let $C_{B,1}, \ldots, C_{B,p}$ be the connected components of $G - B$ such that, for $1 \leqslant i \leqslant p$, $\mathsf{tw}(G[V(C_{B,i}) \cup B]) \leqslant t$. The connected components of $G - B$ can be determined in $\mathcal{O}(n)$ time and one can test whether the graph induced by $V(C_{B,i}) \cup B$ has treewidth at most $t-1$ in time $2^{\mathcal{O}(t^3)} \cdot n$ [58]. Since we have assumed that t is a fixed constant, deciding whether the treewidth is within $t-1$ can be done in linear time. By definition, $\bigcup_{i=1}^{p} V(C_{B,i}) \cup B$ is a t-protrusion with boundary B. Conversely every t-protrusion W consists of a boundary $\partial(W)$ of size at most t such that the restricted protrusion $W' = W \setminus \partial(W)$ is a collection of connected components C of $G - \partial(W)$ satisfying the condition $\mathsf{tw}(G[V(C) \cup \partial(W)]) \leqslant t-1$. Therefore to find a t-protrusion of maximum size, one simply runs through all vertex sets B of size at most t and for each set determines the maximum t-protrusion with boundary B. The largest t-protrusion over all choices of the boundary B is a largest t-protrusion in the graph. All of this takes time $\mathcal{O}(n^{t+1})$. □

Finally, given a $2t$-protrusion W with the desired size constraints, we show how to determine which representative of our equivalence class is equivalent to $G[W]$.

Lemma 5.4 *Let Π be a parameterized graph problem that has finite integer index on $\mathcal{G}$. For a constant $t \in \mathbb{N}$, suppose that the set $\mathcal{R}_t$ of representatives of the equivalence relation $\equiv_{\Pi,t}$ is given. If W is a t-protrusion of size at most a fixed constant c, then one can decide in constant time which $G' \in \mathcal{R}_t$ satisfies $G' \equiv_{\Pi,t} G[W]$.*

Proof: Fix $G' \in \mathcal{R}_t$. We wish to test whether $G' \equiv_{\Pi,t} G[W]$. For each $\tilde{G} \in \mathcal{R}_t$, solve the problem Π on the constant-sized instances $G[W] \oplus \tilde{G}$ and $G' \oplus \tilde{G}$ and let $s(G[W], \tilde{G})$ and $s(G', \tilde{G})$ denote the value of the parameter associated with the problem. Then by the definition of finite integer index, we have $G' \equiv_{\Pi,t} G[W]$ if and only if $s(G[W], \tilde{G}) - s(G', \tilde{G})$ is the *same* for all $\tilde{G} \in \mathcal{R}_t$. To find out which graph in $\mathcal{R}_t$ is the correct representative of $G[W]$, we run this test for each graph in $\mathcal{R}_t$, of which there are a constant number. The total time taken is, therefore, a constant. □

5.3 Constructing protrusion decompositions

In this section we present our algorithm to compute protrusion decompositions. Our approach is based on an algorithm which marks the bags of a tree-decomposition of an input graph G that comes equipped with a subset $X \subseteq V(G)$ such that the graph $G - X$

has bounded treewidth. Let henceforth t be an integer such that $\mathsf{tw}(G - X) \leqslant t - 1$ and let r be an integer that is also given to the algorithm. This parameter r will depend on the particular graph class to which G belongs and the precise problem one might want to solve (see Sections 5.4 and 5.5 for more details). More precisely, given optimal tree-decompositions of the connected components of $G - X$ with at least r neighbors in X, the bag marking algorithm greedily identifies a set $\mathcal{M}$ of bags in a bottom-up manner. The set $V(\mathcal{M})$ of vertices contained in marked bags together with X will form the separating part Y_0 of the protrusion decomposition. Bags will be marked in two different steps, called "Large-subgraph" and "LCA" marking steps. The bags marked in the Large-subgraph marking step will be mapped bijectively into a collection of pairwise vertex-disjoint connected subgraphs of $G - X$, each of which has a large neighborhood in X (namely, of size greater than r), implying in several particular cases a limited number of marked bags (see Sections 5.4 and 5.5). In order to guarantee that the connected components of $G-(X\cup V(\mathcal{M}))$ form protrusions with small boundary, in the LCA marking step the set $\mathcal{M}$ is closed under taking LCA's (least common ancestors; see Lemma 5.6. We would like to note that this technique was also used in [63, 135]). The precise description of the procedure can be found in Algorithm 1 below and a sketch of the decomposition is depicted in Figure 5.2.

Input: A graph G, a subset $X \subseteq V(G)$ such that $\mathsf{tw}(G - X) \leqslant t - 1$, and an integer $r > 0$.

Set $\mathcal{M} \leftarrow \emptyset$ as the set of marked bags;
Compute an optimal rooted tree-decomposition $\mathcal{T}_C = (T_C, \mathcal{B}_C)$ of every connected component C of $G - X$ such that $|N_X(C)| \geqslant r$;

Repeat the following loop for every rooted tree-decomposition $\mathcal{T}_C$;
while $\mathcal{T}_C$ *contains an unprocessed bag* **do**
  Let B be an unprocessed bag at the farthest distance from the root of $\mathcal{T}_C$;
  [LCA marking step]
  if B *is the LCA of two bags of* $\mathcal{M}$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{B\}$ and remove the vertices of B from every bag of $\mathcal{T}_C$;
  end
  [Large-subgraph marking step]
  else if G_B *contains a connected component* C_B *such that* $|N_X(C_B)| \geqslant r$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{B\}$ and remove the vertices of B from every bag of $\mathcal{T}_C$;
  end
  Bag B is now processed;
end
return $Y_0 = X \cup V(\mathcal{M})$;

Algorithm 1: Bag marking algorithm.

Before we discuss properties of the set $\mathcal{M}$ of marked bags and the set $Y_0 = X \cup V(\mathcal{M})$, let us establish the time complexity of the bag marking algorithm and describe how the Large-subgraph marking step can be implemented using dynamic programming techniques. Since

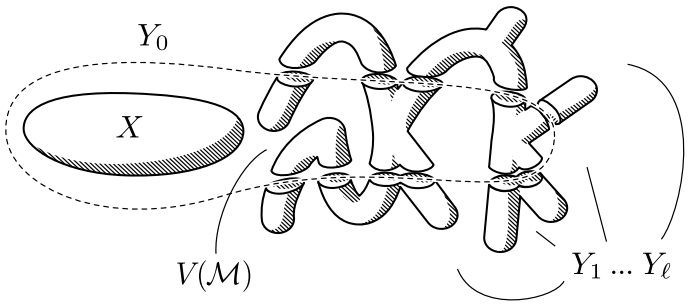

Figure 5.2: A sketch of how the marking algorithm obtains a protrusion decomposition. X denotes a treewidth-modulator. Edges among the individual vertex sets are not depicted.

the dynamic programming procedure is quite standard, we just sketch the main ideas.

Implementation and time complexity of Algorithm 1. First, an optimal tree-decomposition of every connected component C of $G - X$ such that $|N_X(C)| \geqslant r$ can be computed in time linear in $n = |V(G)|$ using the algorithm of Bodlaender for graphs of bounded treewidth [58]. We root such tree-decomposition at an arbitrary bag. For the sake of simplicity of the analysis, we can assume that the tree-decompositions are nice (cf. Section 3.1.3), but it is not necessary for the algorithm.

Note that the LCA marking step can clearly be performed in linear time. Let us now briefly discuss how we can detect, in the Large-subgraph marking step, if a graph G_B contains a connected component C_B such that $|N_X(C_B)| \geqslant r$ using dynamic programming. For each bag B of the tree-decomposition, we have to keep track of which vertices of B belong to the same connected component of G_B.

Note that we only need to remember the connected components of the graph G_B which intersect B, as the other ones will never be connected to the rest of the graph. For each such connected component C_B intersecting B, we also store $N_X(C_B)$, and note that by definition of the algorithm, it follows that for non-marked bags B, $|N_X(C_B)| < r$. At a "join" bag J with children B_1 and B_2, we merge the connected components of G_{B_1} and G_{B_2} sharing at least one vertex (which is necessarily in J), and update their neighborhood in X accordingly. If for some of these newly created connected components C_J of G_J, it holds that $|N_X(C_J)| \geqslant r$, then the bag J needs to be marked. At a "forget" bag F corresponding to a forgotten vertex v, we only have to forget the connected component C of G_F containing v if $V(C) \cap F = \emptyset$. Finally, at an "introduce" bag I corresponding to a new vertex v, we have to merge connected components of G_I after the addition of vertex v, and update the neighbors in X according to the neighbors of v in X.

Note that for each bag B, the time needed to update the information about the connected components of G_B depends polynomially on t and r. In order for the whole algorithm to run in linear time, we can deal with the removal of marked vertices in the following way. Instead of removing them from every bag of the tree-decomposition, we can just label them as "marked" when marking a bag B, and just not take them into account when processing further bags.

The next lemma follows from the above discussion.

Lemma 5.5 *Algorithm 1 can be implemented to run in time $\mathcal{O}(n)$, where the hidden constant depends only on t and r.*

Basic properties of Algorithm 1. Denote by $\mathcal{T}$ the union of the set of optimal tree-decompositions $\mathcal{T}_C$ of every connected component C of $G - X$ with at least r neighbors in X.

Lemma 5.6 *If T is a maximal connected subtree of $\mathcal{T}$ not containing any marked bag of $\mathcal{M}$, then T is adjacent to at most two marked bags of $\mathcal{T}$.*

Proof: As every tree-decomposition in $\mathcal{T}$ is rooted, so is any maximal subtree T of $\mathcal{T}$ not containing any marked bag of $\mathcal{M}$. Assume that $\mathcal{M}$ contains two distinct marked bags, say B_1 and B_2, each adjacent to a leaf of T. As T is connected, observe that the LCA B of B_1 and B_2 belongs to T. Since $\mathcal{M}$ is closed under taking LCA, T contains a marked bag B, a contradiction. It follows that T is adjacent to at most two marked bags: a unique one adjacent to a leaf, and possibly another one adjacent to its root. □

As a consequence of the previous lemma we can now argue that every connected component of $G - Y_0$ has a small neighborhood in X and thus forms a restricted protrusion.

Lemma 5.7 *Let Y_0 be the set of vertices computed by Algorithm 1. Every connected component C of $G - Y_0$ satisfies $|N_X(C)| < r$ and $|N_{Y_0}(C)| < r + 2t$.*

Proof: Let C be a connected component of $G - Y_0$. Observe that C is contained in a connected component C_X of $G - X$ such that either $|N_X(C_X)| < r$ or $|N_X(C_X)| \geqslant r$. In the former case, as Algorithm 1 does not mark any vertex of C_X, $C = C_X$ and so $|N_{Y_0}(C)| < r + 2t$ trivially holds. So assume that $|N_X(C_X)| \geqslant r$. Then C_X has been chopped by Algorithm 1 and clearly $C \subseteq C_X \setminus V(\mathcal{M})$. More precisely, if $\mathcal{T}_{C_X}$ is the rooted tree-decomposition of C_X, there exists a maximal connected subtree T of $\mathcal{T}_{C_X}$ not containing any marked bag such that $C \subseteq V(T) \setminus V(\mathcal{M})$. By construction of $\mathcal{M}$, every connected component of the subgraph induced by $V(T) \setminus V(\mathcal{M})$ has strictly less than r neighbors in X (otherwise the root of T or one of its descendants would have been marked at the Large-subgraph marking step). It follows that $|N_X(C)| < r$. To conclude, observe that Lemma 5.6 implies that the neighbors of C in $V(\mathcal{M})$ are contained in at most two marked bags of $\mathcal{T}$. It follows that $|N_{Y_0}(C)| < r + 2t$. □

Given a graph G and a subset $S \subseteq V(G)$, we define a *cluster of $G - S$* as a maximal collection of connected components of $G - S$ with the same neighborhood in S. Note that the set of all clusters of $G - S$ induces a partition of the set of connected components of $G - S$, which can be easily found in linear time if G and S are given.

By Lemma 5.7 and using the fact that $\mathsf{tw}(G - X) \leqslant t - 1$, the following proposition follows.

Proposition 5.1 *Let r, t be two positive integers, let G be a graph and $X \subseteq V(G)$ such that $\mathsf{tw}(G - X) \leqslant t - 1$, let $Y_0 \subseteq V(G)$ be the output of Algorithm 1 with input (G, X, r), and let $Y_1, \ldots, Y_\ell$ be the set of all clusters of $G - Y_0$. Then $\mathcal{P} := Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ is a $(\max\{\ell, |Y_0|\}, 2t + r)$-protrusion decomposition of G.*

In other words, each cluster of $G - Y_0$ is a restricted $(2t + r)$-protrusion. Note that Proposition 5.1 neither bounds ℓ nor $|Y_0|$. In the sequel, we will use Algorithm 1 and Proposition 5.1 to give explicit bounds on ℓ and $|Y_0|$, in order to achieve two different results. In Section 5.4 we use Algorithm 1 and Proposition 5.1 to obtain *linear kernels* for a large class of problems on *sparse* graphs. In Section 5.5 we use Algorithm 1 and Proposition 5.1 to obtain a *single-exponential algorithm* for the parameterized Planar-$\mathcal{F}$-Deletion problem.

5.4 Linear kernels on graphs excluding a topological minor

We start this section by proving Theorem 5.1 in Section 5.4.1. We then state a number of concrete problems that satisfy the structural constraints imposed by this theorem (Section 5.4.2), discuss these constraints in the context of previous work in this area (Section 5.4.3), and trace graph classes to which our approach can be lifted (Section 5.4.4). Finally (Section 5.4.5), we discuss how to use the machinery developed in proving Theorem 5.1 to obtain a concrete kernel for the Edge Dominating Set problem.

5.4.1 Proof of Theorem 5.1

With the protrusion machinery outlined in Section 5.2 at hand, we can now describe the protrusion reduction rule. Informally, we find a sufficiently large t-protrusion (for some yet to be fixed constant t), replace it with a small representative, and change the parameter accordingly. In the following, we will drop the subscript from the protrusion limit functions ρ_Π and ρ'_Π.

Reduction Rule 1 (Protrusion reduction rule) *Let $\Pi_\mathcal{G}$ denote a parameterized graph problem restricted to some graph class $\mathcal{G}$, let $(G, k) \in \Pi_\mathcal{G}$ be a Yes-instance of $\Pi_\mathcal{G}$, and let $t \in \mathbb{N}$ be a constant. Suppose that $W' \subseteq V(G)$ is a t-protrusion of G such that $|W'| > \rho'(t)$, obtained as described in Lemma 5.3. Let $W \subseteq V(G)$ be a $2t$-protrusion of G such that $\rho'(t) < |W| \leqslant 2 \cdot \rho'(t)$, obtained as described in Lemma 5.2. We let G_W denote the $2t$-boundaried graph $G[W]$ with boundary $\mathbf{bd}(G_W) = \partial_G(W)$. Let further $G_1 \in \mathcal{R}_{2t}$ be the representative of G_W for the equivalence relation $\equiv_{\Pi, |\partial_G(W)|}$ as defined in Lemma 5.1.*

The protrusion reduction rule (for boundary size t) is the following:

$$\text{Reduce } (G, k) \text{ to } (G', k') = (G \ominus G_W \oplus G_1, k - \Delta_{\Pi, 2t}(G_1, G_W)).$$

By Lemma 5.1, the parameter in the new instance does not increase. We now show that the protrusion reduction rule is safe.

Lemma 5.8 (Safety) *Let $\mathcal{G}$ be a graph class and let $\Pi_{\mathcal{G}}$ be a parameterized graph problem with finite integer index with respect to $\mathcal{G}$. If (G', k') is the instance obtained from one application of the protrusion reduction rule to the instance (G, k) of $\Pi_{\mathcal{G}}$, then*

1. *$G' \in \mathcal{G}$;*
2. *(G', k') is a* YES*-instance if and only if (G, k) is a* YES*-instance; and*
3. *$k' \leqslant k$.*

Proof: Suppose that (G', k') is obtained from (G, k) by replacing a $2t$-boundaried subgraph G_W (induced by a $2t$-protrusion W) by a representative $G_1 \in \mathcal{R}_{2t}$. Let $\tilde{G}$ be the $2t$-boundaried graph $G - W'$, where W' is the restricted protrusion of W and $\mathbf{bd}(\tilde{G}) = \partial_G(W)$. Since $G_W \equiv_{\Pi,2t} G_1$, we have by Definition 5.5,

1. $G = \tilde{G} \oplus G_W \in \mathcal{G}$ iff $\tilde{G} \oplus G_1 \in \mathcal{G}$.
2. $(\tilde{G} \oplus G_W, k) \in \Pi_{\mathcal{G}}$ iff $(\tilde{G} \oplus G_1, k - \Delta_{\Pi,2t}(G_1, G_W)) \in \Pi_{\mathcal{G}}$.

Hence $G' = \tilde{G} \oplus G_1 \in \mathcal{G}$. Lemma 5.1 ensures that $\Delta_{\Pi,2t}(G_1, G_W) \geqslant 0$, and hence $k' = k - \Delta_{\Pi,2t}(G_1, G_W)) \leqslant k$. □

Remark 5.1 *If (G, k) is reduced with respect to the protrusion reduction rule with boundary size β, then for all $t \leqslant \beta$, every t-protrusion W of G has size at most $\rho'(t)$.*

In order to obtain linear kernels, we require the problem instances to have more structure. In particular, we adapt the notion of *quasi-compactness* introduced in [63] to define what we call *treewidth-bounding*.

Definition 5.7 (Treewidth-bounding) *A parameterized graph problem $\Pi_{\mathcal{G}}$ is called* (s, t)-treewidth-bounding *if there exists a function $s\colon \mathbb{N} \to \mathbb{N}$ and a constant t such that for every $(G, k) \in \Pi_{\mathcal{G}}$ there exists $X \subseteq V(G)$ such that:*

1. *$|X| \leqslant s(k)$; and*
2. *$\mathsf{tw}(G - X) \leqslant t - 1$.*

We call a problem treewidth-bounding on a graph class $\mathcal{G}$ *if the above property holds under the restriction that $G \in \mathcal{G}$. We call X a* t-treewidth-modulator *of G, s the* treewidth-modulator size *and t the* treewidth bound *of the problem Π.*

We assume in the following that the problem $\Pi_{\mathcal{G}}$ at hand is (s, t)-treewidth-bounding with bound t and modulator size $s(\cdot)$, that is, a YES-instance $(G, k) \in \Pi_{\mathcal{G}}$ has a modulator set $X \subseteq V(G)$ with $|X| \leqslant s(k)$ and $\mathsf{tw}(G - X) \leqslant t - 1$. Note that in general s, t depend on $\Pi_{\mathcal{G}}$ and $\mathcal{G}$. For many problems that are treewidth-bounding, such as VERTEX COVER, FEEDBACK VERTEX SET, TREEWIDTH-t VERTEX DELETION, the set X is actually the solution set. However, in general, X could be *any* vertex set and does not have to be given nor efficiently computable to obtain a kernel. The fact that it exists is all we need for our proof to go through.

The rough idea of the proof of Theorem 5.1 is as follows. We assume that the given instance (G, k) is reduced w.r.t. the protrusion reduction rule for some yet to be fixed

constant boundary size β. Consequently, every β-protrusion of G has size at most $\rho'(\beta)$. For a protrusion decomposition $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ obtained from Algorithm 1 with a carefully chosen threshold r (see Proposition 5.1), we can then show that $|Y_0| = \mathcal{O}(k)$ using properties of H-topological-minor-free graphs. The bound on the total size of the clusters of $G - Y_0$ follows from these properties and from the protrusion reduction rule.

We first prove a result (Theorem 5.3) that is slightly more general than Theorem 5.1 and identifies all the key ingredients needed for our result. To do this, we use a sequence of lemmas (5.9, 5.10, 5.11) which bounds the total size of the clusters of the protrusion decomposition. To this end, we define the *constriction* operation, which essentially shrinks paths into edges.

Definition 5.8 (Constriction) *Let G be a graph and let $\mathcal{P}$ be a set of paths in G such that for each $P \in \mathcal{P}$ it holds that:*

1. *the endpoints of P are not connected by an edge in G; and*
2. *for all $P' \in \mathcal{P}$, with $P' \neq P$, P and P' share at most a single vertex which must also be an endpoint of both paths.*

We define the constriction *of G under $\mathcal{P}$, written $G|_{\mathcal{P}}$, as the graph H obtained by connecting the endpoints of each $P \in \mathcal{P}$ by an edge and then removing all inner vertices of P.*

We say that H is a *d-constriction* of G if there exists $G' \subseteq G$ and a set of paths $\mathcal{P}$ in G' such that $d = \max_{P \in \mathcal{P}} |P|$ and $H = G'|_{\mathcal{P}}$. Given graph classes $\mathcal{G}, \mathcal{H}$ and some integer $d \geqslant 2$, we say that $\mathcal{G}$ *d-constricts into* $\mathcal{H}$ if for every $G \in \mathcal{G}$, every possible d-constriction H of G is contained in the class $\mathcal{H}$. For the case that $\mathcal{G} = \mathcal{H}$ we say that $\mathcal{G}$ is *closed under d-constrictions*. We will call $\mathcal{H}$ the *witness* class, as the proof of Theorem 5.3 works by taking an input graph G and constricting it into some witness graph H whose properties will yield the desired bound on $|G|$. We let $\omega(G)$ denote the size of a largest clique in G and $\#\omega(G)$ the total number of cliques in G (not necessarily maximal ones).

Theorem 5.3 *Let $\mathcal{G}, \mathcal{H}$ be graph classes closed under taking subgraphs such that $\mathcal{G}$ d-constricts into $\mathcal{H}$ for a fixed constant $d \in \mathbb{N}$. Assume that $\mathcal{H}$ has the property that there exist functions $f_E, f_{\#\omega} \colon \mathbb{N} \to \mathbb{N}$ and a constant $\omega_{\mathcal{H}}$ (depending only on $\mathcal{H}$) such that for each graph $H \in \mathcal{H}$ the following conditions hold:*

$$|E(H)| \leqslant f_E(|H|), \;\; \#\omega(H) \leqslant f_{\#\omega}(|H|), \;\; \text{and } \omega(H) < \omega_{\mathcal{H}}.$$

Let Π be a parameterized graph problem that has finite integer index and is (s,t)-treewidth-bounding, both on the graph class $\mathcal{G}$. Define $x_k := s(k) + 2t \cdot f_E(s(k))$. Then any reduced instance $(G,k) \in \Pi$ has a protrusion decomposition $V(G) = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ such that:

1. *$|Y_0| \leqslant x_k$;*
2. *$|Y_i| \leqslant \rho'(2t + \omega_{\mathcal{H}})$ for $1 \leqslant i \leqslant \ell$; and*
3. *$\ell \leqslant f_{\#\omega}(x_k) + f_E(x_k) + x_k + 1$.*

Hence Π *restricted to* $\mathcal{G}$ *admits kernels of size at most*

$$x_k + (f_{\#\omega}(x_k) + f_E(x_k) + x_k + 1)\rho'(2t + \omega_{\mathcal{H}}).$$

Even if it is not our main objective to optimize the running time of the kernelization algorithm given by Theorem 5.3, we just note that it is dominated by the algorithm of Lemma 5.3 for finding protrusions. We split the proof of Theorem 5.3 into several lemmas. First, let us fix the way in which the decomposition $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ is obtained: given a reduced YES-instance $(G, k) \in \Pi$, let $X \subseteq V(G)$ be a treewidth-modulator of size at most $|X| \leqslant s(k)$ such that $\mathsf{tw}(G - X) \leqslant t - 1$. For the analysis of the kernel size, we run Algorithm 1 on the input $(G, X, \omega_{\mathcal{H}})$.

Lemma 5.9 *The protrusion decomposition* $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ *obtained by running Algorithm 1 on* $(G, X, \omega_{\mathcal{H}})$ *has the following properties:*

1. *For each* $1 \leqslant i \leqslant \ell$, *we have* $|Y_i| \leqslant \rho'(2t + \omega_{\mathcal{H}})$;
2. *For each connected subgraph* C_B *found by Algorithm 1 in the "Large-subgraph marking step",* $|C_B| \leqslant \rho'(2t + \omega_{\mathcal{H}}) + t$.

Proof: The first claim follows directly from Lemma 5.7: for each $1 \leqslant i \leqslant \ell$, we have $|N_{Y_0}(Y_i)| \leqslant 2t + \omega_{\mathcal{H}}$. As $Y_i \subseteq G - X$, it follows that $\mathsf{tw}(G[Y_i]) \leqslant t - 1$ and therefore Y_i forms a restricted $(2t + \omega_{\mathcal{H}})$-protrusion in G. Since our instance is reduced, we have $|Y_i| \leqslant \rho'(2t + \omega_{\mathcal{H}})$.

Note that during a run of the algorithm, if a bag B currently being considered is not marked, then each connected component C_B of G_B satisfies $|N_X(C_B)| < \omega_{\mathcal{H}}$. Hence C_B along with its neighbors in X is a $(2t + \omega_{\mathcal{H}})$-protrusion and since the instance is reduced we have $|C_B| \leqslant \rho'(2t + \omega_{\mathcal{H}})$. Moreover the algorithm ensures that $|N_R(C_B)| \leqslant 2t$, where $R = V(G) \setminus X$, and thus a component with a neighborhood larger than $2t + \omega_{\mathcal{H}}$ must have at least $\omega_{\mathcal{H}}$ neighbors in X. Now as every step of the algorithm adds at most t more vertices to the components of G_B, it follows that once a component with at least $\omega_{\mathcal{H}}$ neighbors in X *is* found, it can contain at most $\rho'(2t + \omega_{\mathcal{H}}) + t$ vertices. □

Now, let us prove the claimed bound on $|Y_0|$ by making use of the assumed bounds $\omega_{\mathcal{H}}$ and $f_E(\cdot)$ imposed on graphs of the witness class $\mathcal{H}$.

Lemma 5.10 *The number of bags marked by Algorithm 1 to obtain* $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ *is at most* $2f_E(s(k))$, *and therefore* $|Y_0| \leqslant x_k = s(k) + 2f_E(s(k)) \cdot t$.

Proof: For each bag marked in the "Large-subgraph marking step" of the algorithm, a connected subgraph C of $G - X$ with $|N_X(C)| \geqslant \omega_{\mathcal{H}}$ is found. Suppose that the algorithm finds p such connected subgraphs $C_1, \ldots, C_p$. Then the number of marked bags is at most $2p$, since the LCA marking step can at most double the number of marked bags.

By the design of Algorithm 1, the connected subgraphs C_i are pairwise vertex-disjoint and $|C_i| \leqslant \rho'(2t + \omega_{\mathcal{H}}) + t$, for all $1 \leqslant i \leqslant p$, cf. Lemma 5.9. Define $\mathcal{P}$ to be a largest collection of paths such that the following conditions hold. For each path $P \in \mathcal{P}$:

- the endpoints of P are both in X;
- the inner vertices of P are all in a single subgraph C_i, for some $1 \leqslant i \leqslant p$; and
- for all $P' \in \mathcal{P}$ with $P' \neq P$, the endpoints of P and P' are not identical and their inner vertices are in different subgraphs C_i and C_j.

First, we show that any largest collection $\mathcal{P}$ of paths satisfying the above conditions is such that $|\mathcal{P}| = p$, that is, such a collection has one path per subgraph in $\{C_1, \ldots, C_p\}$. Assume that $\mathcal{P}$ is a largest collection of paths satisfying the conditions stated above and consider the graph $H = G|_{\mathcal{P}}[X]$ induced by the vertex set X in the graph $G|_{\mathcal{P}}$ obtained by constricting the paths in $\mathcal{P}$. By assumption, $H \in \mathcal{H}$ as $\mathcal{G}$ d-constricts into $\mathcal{H}$ and $\mathcal{H}$ is closed under taking subgraphs. The constant d is given by

$$d = \max_{P \in \mathcal{P}} |P| \leqslant \max_{1 \leqslant i \leqslant p} |C_i| \leqslant \rho'(2t + \omega_{\mathcal{H}}) + t.$$

Suppose that $|\mathcal{P}| < p$, i.e., there exists some C_i with $1 \leqslant i \leqslant p$ such that no path of $\mathcal{P}$ uses vertices of C_i. Consider the neighborhood $Z = N_X^G(C_i)$ of C_i in X. As we chose the threshold of the marking algorithm to ensure that $|Z| \geqslant \omega_{\mathcal{H}}$, it follows that Z cannot induce a clique in H. But then there exist vertices $u, v \in Z$ with $uv \notin E(H)$ and we could add a uv-path whose inner vertices are in C_i to $\mathcal{P}$ without conflicting with any of the above constraints (including the bound on d), which contradicts our assumption that $\mathcal{P}$ is of largest size. We therefore conclude that $|\mathcal{P}| = p$.

Since there is a bijection from the collection of subgraphs $\{C_1, \ldots, C_p\}$ and the paths of $\mathcal{P}$, we may bound p by the number of edges in H, which is at most $f_E(|H|)$. But $|H| = |X| = s(k)$ and we thus obtain the bound $p \leqslant f_E(s(k))$ on the *number* of large-degree subgraphs found by Algorithm 1. Therefore the number of marked bags is $|\mathcal{M}| \leqslant 2f_E(s(k))$. As every marked bag adds at most t vertices to Y_0, we obtain the claimed bound

$$|Y_0| = |X| + t \cdot |\mathcal{M}| \leqslant s(k) + 2t \cdot f_E(s(k)) = x_k. \qquad \square$$

We will now use this bound on the size of Y_0 to bound the sizes of the clusters $Y_1 \uplus \cdots \uplus Y_\ell$ of $G - Y_0$. The important properties used are that the instance (G, k) is reduced, that each Y_i has a small neighborhood in Y_0 and hence has small size, and that the witness graph obtained from G via constrictions has a bounded number of cliques, given by the function $f_{\#\omega}(\cdot)$.

Lemma 5.11 *The number of vertices in* $\bigcup_{1 \leqslant i \leqslant \ell} Y_i$ *is bounded by* $(f_{\#\omega}(|Y_0|) + f_E(|Y_0|) + |Y_0| + 1) \cdot \rho'(2t + \omega_{\mathcal{H}})$.

Proof: The clusters $Y_1, \ldots, Y_\ell$ contain connected components of $G - Y_0$ and have the property that for each $1 \leqslant i \leqslant \ell$, $|N_{Y_0}^G(Y_i)| \leqslant 2t + \omega_{\mathcal{H}}$. We proceed analogously to the proof of Lemma 5.10. Let $\mathcal{P}$ be a maximum collection of paths P such that the endvertices of P are in Y_0 and all its inner vertices are in some cluster Y_i. Moreover for all paths $P_1, P_2 \in \mathcal{P}$, with $P_1 \neq P_2$, each path has a distinct set of endvertices and a distinct component for their inner vertices. Note that some clusters might have no or only one neighbor in Y_0, those cannot be used by any of the paths in $\mathcal{P}$.

Consider the graph $H = G|_{\mathcal{P}}[Y_0]$ induced by Y_0 in the graph obtained from G by constricting the paths in $\mathcal{P}$. Note that for each cluster Y_i whose vertices do not participate in any path of $\mathcal{P}$ it holds that $Z_i = N^G_{Y_0}(Y_i)$ induces a clique in H as otherwise we could augment $\mathcal{P}$ by another path. The neighborhoods of clusters not participating in any path of $\mathcal{P}$ therefore can be upperbounded by the number of cliques in H. The neighborhoods of clusters that *do* participate in paths of $\mathcal{P}$ in turn are upperbounded by $|\mathcal{P}|$, which in turns is at most $|E(H)|$.

Using the bounds $f_{\#\omega}, f_E$ for $\mathcal{H}$, we deduce that the collection of neighborhoods $\{Z_1, \ldots, Z_\ell\}$, where $Z_i = N^G_{Y_0}(Y_i)$ for $1 \leqslant i \leqslant \ell$, contains at most $f_{\#\omega}(|H|) + f_E(|H|) + |H| + 1$ distinct sets where the sum $|H| + 1$ takes care of clusters that have zero to one neighbor in Y_0. Thus

$$\ell \leqslant f_{\#\omega}(|H|) + f_E(|H|) + |H| + 1 = f_{\#\omega}(|Y_0|) + f_E(|Y_0|) + |Y_0| + 1,$$

where we used the fact that $|H| = |Y_0|$. Since $Y_1, \ldots, Y_\ell$ are clusters w.r.t. Y_0, we obtain ℓ restricted $(2t + \omega_{\mathcal{H}})$-protrusions in G (adding the respective neighborhood in Y_0 to each cluster yields the corresponding $(2t+\omega_{\mathcal{H}})$-protrusion). Thus the sets $Y_1, \ldots, Y_\ell$ contain in total at most

$$\Big|\bigcup_{1 \leqslant i \leqslant \ell} Y_i\Big| \leqslant (f_{\#\omega}(|Y_0|) + f_E(|Y_0|) + |Y_0| + 1) \cdot \rho'(2t + \omega_{\mathcal{H}})$$

vertices. □

We can now easily prove Theorem 5.3.

Proof of Theorem 5.3: By Lemma 5.10 we know that $|Y_0| = x_k$. Together with Lemma 5.11 we can bound the total number of vertices in a reduced instance by

$$\begin{aligned} |V(G)| &= |Y_0| + |Y_1| + \cdots + |Y_\ell| \\ &\leqslant x_k + (f_{\#\omega}(x_k) + f_E(x_k) + x_k + 1)\rho'(2t + \omega_{\mathcal{H}}), \end{aligned}$$

again using the shorthand $x_k = s(k) + 2f_E(s(k)) \cdot t$. □

We now show how to apply Theorem 5.3 to obtain kernels. Let $\mathcal{G}_H$ be the class of graphs that exclude some fixed graph H as a topological minor. Observe that $\mathcal{G}_H$ is closed under taking topological minors, and is therefore closed under taking d-constrictions for any $d \geqslant 2$.

In order to obtain $f_E, f_{\#\omega}$, and $\omega_{\mathcal{G}_H}$ we use the fact that H-topological-minor-free graphs are *ε-degenerate*. That is, there exists a constant ε (that depends only on H) such that every subgraph of $G \in \mathcal{G}_H$ contains a vertex of degree at most ε. The following are well-known properties of degenerate graphs.

Proposition 5.2 (Bollobás and Thomason [71], Komlós and Szemerédi [186]) *There is a constant $\beta \leqslant 10$ such that, for $r > 2$, every graph with no K_r-topological-minor has average degree at most βr^2.*

As an immediate consequence, any graph with average degree larger than βr^2 contains *every* r-vertex graph as a topological minor. If a graph G excludes H as a topological minor, then G clearly excludes $K_{|H|}$ as a topological minor. What is also true is that the *total* number of cliques (not necessarily maximal) in G is $\mathcal{O}(|G|)$.

Proposition 5.3 (Fomin, Oum, and Thilikos [142]) *There is a constant $\tau < 4.51$ such that, for $r > 2$, every n-vertex graph with no K_r-topological-minor has at most $2^{\tau r \log r} n$ cliques.*

Henceforth, let $r := |H|$ denote the size of the forbidden topological minor. The following is a slightly generalized version of our first main theorem.

Theorem 5.4 *Fix a graph H and let $\mathcal{G}_H$ be the class of H-topological-minor-free graphs. Let Π be a parameterized graph problem that has finite integer index and is $(s_{\Pi,\mathcal{G}_H}, t_{\Pi,\mathcal{G}_H})$-treewidth-bounding on the class $\mathcal{G}_H$. Then Π admits a kernel of size $\mathcal{O}(s_{\Pi,\mathcal{G}_H}(k))$.*

Proof: We use Theorem 5.3 with the functions $f_E(n) = \frac{1}{2}\beta r^2 n, f_{\#\omega}(n) = 2^{\tau r \log r} n$ obtained from Propositions 5.2 and 5.3. Observe that an H-topological-minor-free graph cannot contain a clique of size r, thus $\omega_{\mathcal{G}_H} \leqslant r$. The kernel size is then bounded by

$$s_{\Pi,\mathcal{G}_H}(k) \cdot \left(1 + \beta r^2 t\right) \left(1 + (2^{\tau r \log r} + \frac{1}{2}\beta r^2 + 1)\rho'(2t + r)\right) + \rho'(2t + r)$$

$$s_{\Pi,\mathcal{G}_H}(k) \cdot (1 + \beta r^2 t + (2^{\tau r \log r}(1 + \beta r^2 t) + \beta r^2 t) \cdot \rho'(2t + r)) + \rho'(2t + r),$$

where we omitted the subscript of $t_{\Pi,\mathcal{G}_H}$ for the sake of readability. □

Theorem 5.1 is now just a consequence of the special case for which the treewidth-bound is linear. Note that the class of graphs with bounded degree is a subset of those that exclude a fixed topological minor, thus the above result translates directly to this class.

5.4.2 Problems affected by our result

We present concrete problems that satisfy the prerequisites of Theorem 5.1.

Corollary 5.1 *Fix a graph H. The following problems are linearly treewidth-bounding and have finite integer index and linear treewidth-bound on the class of H-topological-minor-free graphs and hence possess a linear kernel on this graph class:* Vertex Cover[7]*;* Cluster Vertex Deletion[7]*;* Feedback Vertex Set*;* Chordal Vertex Deletion[8]*;* Interval *and* Proper Interval Vertex Deletion*;* Cograph Vertex Deletion*;* Edge Dominating Set*.*

[7] Listed for completeness; these problems have a kernel with a linear number of vertices on general graphs.

[8] Note that Chordal Vertex Deletion is indeed linearly treewidth-bounding on H-topological-minor-free graphs, since an H-topological-minor-free chordal graph has bounded clique size, and hence bounded treewidth as well.

In particular, Corollary 5.1 also implies that Chordal Vertex Deletion and Interval Vertex Deletion can be decided on H-topological-minor-free graphs in time $\mathcal{O}(c^k \cdot \text{poly}(n))$ for some constant c. (This follows because one can first obtain a linear kernel and then use brute-force to solve the kernelized instance.) On general graphs (complicated) single-exponential algorithms for Interval Vertex Deletion have appeared only very recently [77,78,219] (until now, this problem was not even known to be FPT), whereas only an $\mathcal{O}(f(k) \cdot \text{poly}(n))$ algorithm is known for Chordal Vertex Deletion, where $f(k)$ is not even specified [203].

Corollary 5.2 Chordal Vertex Deletion *and* Interval Vertex Deletion *are solvable in single-exponential time on H-topological-minor-free graphs.*

A natural extension of the (vertex deletion) problems in Corollary 5.1 is to seek a solution that induces a *connected* graph. The connected versions of problems are typically more difficult both in terms of proving fixed-parameter tractability and establishing polynomial kernels. For instance, Vertex Cover admits a $2k$-vertex kernel but Connected Vertex Cover has no polynomial kernel unless NP $\subseteq$ coNP/poly [108]. However on H-topological-minor-free graphs, Connected Vertex Cover (and a couple of others) admit a linear kernel.

Corollary 5.3 Connected Vertex Cover, Connected Cograph Vertex Deletion, *and* Connected Cluster Vertex Deletion *have linear kernels in graphs excluding a fixed topological minor.*

Another property owell-known graph width measures treewidth (tw), rankwidth (rw), and cliquewidth (cw), are all within a constant multiplicative factor of one another.

Proposition 5.4 (Fomin, Oum, and Thilikos [142]) *There is a constant τ such that for every $r > 2$, if G excludes K_r as a topological minor, then*

$$\mathsf{rw}(G) \leqslant \mathsf{cw}(G) < 2 \cdot 2^{\tau r \log r} \mathsf{rw}(G)$$
$$\mathsf{rw}(G) \leqslant \mathsf{tw}(G) + 1 < \frac{3}{4}(r^2 + 4r - 5) 2^{\tau r \log r} \mathsf{rw}(G).$$

An interesting vertex-deletion problem related to graph width measures is Width-b Vertex Deletion [182]: given a graph G and an integer k, do there exist at most k vertices whose deletion results in a graph with width at most b? From Definition 5.7 (see Section 5.2), it follows that if the width measure is treewidth, then this problem is treewidth-bounding. By Proposition 5.4, this also holds if the width measure is either rankwidth or cliquewidth. The fact that this problem has finite integer index follows from the sufficiency condition known as *strong monotonicity* in [63]. Since branchwidth differs only by a constant factor from treewidth in general graphs [224], this gives us the following.

Corollary 5.4 *The* Width-b Vertex Deletion *problem has a linear kernel on H-topological-minor-free graphs, where the width measure is either treewidth, cliquewidth, branchwidth, or rankwidth.*

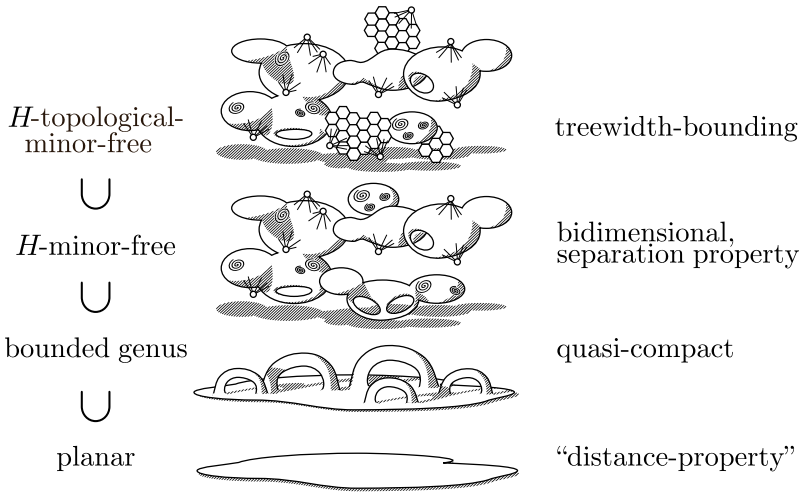

Figure 5.3: Kernelization results for problems with finite integer index on sparse graph classes with their corresponding additional condition.

5.4.3 A comparison with earlier results

We briefly compare the structural constraints imposed in Theorem 5.1 with those imposed in the results on linear kernels on graphs of bounded genus [63] and H-minor-free graphs [140]. In particular, we discuss how restrictive is the condition of being treewidth-bounding. A graphical summary of the various notions of sparseness and the associated structural constraints used to obtain results on linear kernels is depicted in Figure 5.3.

The theorem that guarantees linear kernels on graphs of bounded genus in [63] imposes a condition called *quasi-compactness*. The notion of quasi-compactness is similar to that of treewidth-bounding: YES-instances (G, k) satisfy the condition that there exists a vertex set $X \subseteq V(G)$ of "small" size whose deletion yields a graph of bounded treewidth. Formally, a problem Π is called *quasi-compact* if there exists an integer r such that for every $(G, k) \in \Pi$, there is an embedding of G onto a surface of Euler-genus at most g and a set $X \subseteq V(G)$ such that $|X| \leqslant r \cdot k$ and $\mathsf{tw}(G - R^r_G(X)) \leqslant r$. Here $R^r_G(X)$ denotes the set of vertices of G at radial distance at most r from X. It is easy to see that the property of being treewidth-bounding is stronger than quasi-compactness in the sense that if a problem is treewidth-bounding and the graphs are embeddable on a surface of genus g, then the problem is also quasi-compact, but not the other way around. The fact that we use a stronger structural condition is expected, since our result proves a linear kernel on a much larger graph class.

More interesting are the conditions imposed for linear kernels on H-minor-free graphs [140]. The problems here are required to be *bidimensional* and satisfy a so-called *separation property*. Roughly speaking, a problem is bidimensional if the solution size on a $k \times k$-grid is $\Omega(k^2)$ and the solution size does not decrease by deleting/contracting edges. The notion of the separation property is essentially the following. A problem has the separation property, if for any graph G and any vertex subset $X \subseteq V(G)$, the optimum solution of G projected on any subgraph G' of $G - X$ differs from the optimum for G' by at most $|X|$

(cf. [140] for details.) At first glance, these conditions seem to have nothing to do with the property of being treewidth-bounding. However in the same paper [140, Lemma 3.2], the authors show that if a problem on H-minor-graphs is bidimensional and has the separation property then it is also (ck, t)-treewidth-bounding for some constants c, t that depend on the graph H excluded as a minor. Using this fact, the main result of [140] (namely, that bidimensional problems with FII and the separation property have linear kernels on H-minor-free graphs) can be reproved as an easy corollary of Theorem 5.3.

This discussion shows that in the results on linear kernels on sparse graph classes that we know so far, the treewidth-bounding condition has appeared in some form or the other. In the light of this we feel that this is the key condition for proving linear kernels on sparse graph classes.

5.4.4 The limits of our approach

It is interesting to know for which notions of sparseness (beyond H-topological-minor-free graphs) we can use our technique to obtain polynomial kernels. We show that our technique *fails* for the following notion of sparseness: graph classes that locally exclude a minor [98]. The notion of locally excluding a minor was introduced by Dawar et al. [98] and graphs that locally exclude a minor include bounded-genus graphs but are incomparable with H-minor-free graphs [213]. However we also show that there exist (restricted) graph classes that locally exclude a minor where it is still possible to obtain a polynomial kernel using our technique.

Definition 5.9 (Locally excluding a minor [98]) *A class $\mathcal{G}$ of graphs* locally excludes a minor *if for every $r \in \mathbb{N}$ there is a graph H_r such that the r-neighborhood of a vertex of any graph of $\mathcal{G}$ excludes H_r as a minor.*

Therefore if $\mathcal{G}$ locally excludes a minor then the 1-neighborhood of a vertex in any graph of $\mathcal{G}$ does not contain H_1 as a minor, and hence as a subgraph. In particular, the neighborhood of no vertex contains a clique on $h_1 := |H_1|$ vertices as a subgraph, meaning that the clique number (that is, the maximum size of a clique) of such graphs is bounded from above by h_1. The total number of cliques in any graph of $\mathcal{G}$ is then bounded by $h_1 n^{h_1}$, and the number of edges can be trivially bounded by n^2. We now have almost all the prerequisites for applying Theorem 5.3. However the class $\mathcal{G}$ is not closed under taking d-constrictions. Taking a d-constriction in a graph $G \in \mathcal{G}$ can increase the clique number of the constricted graph. This seems to be a bottleneck in applying Theorem 5.3. However if we assume that the size of the locally forbidden minors $\{H_r\}_{r\in\mathbb{N}}$ grows very slowly, then we can still obtain a polynomial kernel.

Definition 5.10 *Given $g\colon \mathbb{N} \to \mathbb{N}$, we say that a graph class $\mathcal{G}$* locally excludes minors according to g *if there exists a constant $n_0 \in \mathbb{N}$, such that for all $r \geqslant n_0$, the $g(r)$-neighborhood of a vertex in any graph of $\mathcal{G}$ does not contain K_r as a minor.*

Lemma 5.12 *Let $\mathcal{G}$ be a graph class that locally excludes a minor according to $g\colon \mathbb{N} \to \mathbb{N}$ and let n_0 be the constant as in the above definition. Then for any $r \geqslant n_0$, the class $\mathcal{G}$ $g(r)$-constricts into a graph class $\mathcal{H}$ that excludes K_r as a subgraph.*

Proof: Assume the contrary. Let $G \in \mathcal{G}$ and suppose that for some $r \geqslant 2$ the graph H obtained by a $g(r)$-constriction of G contains K_r as a subgraph. Pick any vertex v in this subgraph of H. The $g(r)$-neighborhood of v in G must contain K_r as a minor, a contradiction. □

Note that in the following, we assume that the problem is treewidth-bounding on general graphs.

Corollary 5.5 *Let Π be a parameterized graph problem with finite integer index that is $(s(k), t_\Pi)$-treewidth-bounding. Let $\mathcal{G}$ be a graph class locally excluding a minor according to a function $g\colon \mathbb{N} \to \mathbb{N}$ such that for all $r \geqslant n_0$, $g(r) \geqslant \rho'(2t_\Pi + r) + 1$. Then there exists a constant r_0 such that Π admits kernels of size $\mathcal{O}(s(k)^{r_0})$ on $\mathcal{G}$.*

Proof: By Lemma 5.12, taking a $g(r)$-constriction results in a graph class $\mathcal{H}$ that excludes K_r as a subgraph, for large enough r. Fixing $r = n_0$, where n_0 is the constant in Definition 5.10, we apply Theorem 5.3 with the trivial functions $f_E(n) = n^2$, $f_{\#\omega}(n) = r{\cdot}n^r$ and $\omega_{\mathcal{H}} = r$. By Lemma 5.12, we have that $\omega_{\mathcal{G}_H} \leqslant r$. The kernel size is then bounded by

$$s(k)+2ts(k)^2+\big((s(k)+2t\cdot s(k)^2)^r+(s(k)+2ts(k)^2)^2+s(k)+2ts(k)^2+1\big)\,\rho'(2t+r) \in \mathcal{O}(s(k)^{2r}),$$

where we omitted the subscript of t_Π for the sake of readability. With $r_0 = 2r = 2n_0$, the bound in the statement of the corollary follows. □

We do not know how quickly the function $\rho'(\cdot)$ grows but intuition from automata theory seems to suggest that this has at least superexponential growth. As such, the graph class for which the polynomial kernel result holds (Corollary 5.5) is pretty restricted. However this does suggest a limit to which our approach can be pushed as well as some intuition as to why our result is not easily extendable to graph classes locally excluding a minor. We note that graph classes of bounded expansion present the same problem.

5.4.5 An illustrative example: Edge Dominating Set

In this section we show how Theorem 5.1 can actually be used to obtain a simple explicit kernel for the Edge Dominating Set problem on H-topological-minor-free graphs. This is made possible by the fact that we can find in polynomial time a small enough treewidth-modulator *and* replace the generic protrusion reduction rule by a handcrafted specific reduction rule.

Let us first recall the problem at hand. We say that an edge e is *dominated* by a set of edges D if either $e \in D$ or e is incident with at least one edge in D. The problem Edge Dominating Set asks, given a graph G and an integer k, whether there is an edge dominating set $D \subseteq E(G)$ of size at most k, i.e., an edge set which dominates every edge of G. The canonical parameterization of this problem is by the integer k, i.e., the size of solution set.

There is a simple 2-approximation algorithm for Edge Dominating Set [243]. Given an instance (G, k), where G is H-topological-minor-free, let D be an edge dominating

set of G, given by the 2-approximation. We can assume that $|D| \leqslant 2k$ since otherwise we can correctly declare (G, k) as a No-instance. Take $X := \{v \in V(G) \mid v \text{ is incident to some edge in } D\}$ as the treewidth-modulator: note that $|X| \leqslant 4k$ and that $G - X$ is of treewidth at most 0, i.e., an independent set. One can easily verify that that the bag marking Algorithm 1 of Section 5.3 would mark exactly those vertices of $G - X$ whose neighborhood in X has size at least $r := |H|$. By applying the edge-bound of Proposition 5.2 to Lemma 5.10 we get that $|V(\mathcal{M})| \leqslant \beta r^2 \cdot 8k$.

Take $Y_0 := X \cup V(\mathcal{M})$ and let $\mathcal{P} := Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ be a partition of $V(G)$, where again Y_i, $1 \leqslant i \leqslant \ell$, is now a cluster w.r.t. Y_0, i.e., the vertices in a single Y_i share the same neighborhood in X and the Y_i are of maximal size under this condition. We have one reduction rule, which can be construed as an concrete instantiation of generic the protrusion replacement rule. We would like to stress that this reduction rule relies on the fact that we already have a protrusion decomposition of G, given by Algorithm 1.

> **Twin elimination rule**: If $|Y_i| > |N_{Y_0}(Y_i)|$ for some $i \neq 0$, let G' be the instance obtained by keeping $|N_{Y_0}(Y_i)|$ many vertices of Y_i and removing the rest of Y_i. Take $k' := k$.

Lemma 5.13 *The twin elimination rule is safe.*

Proof: Let G_i be the graph induced by the vertex set $N_{Y_0}(Y_i) \cup Y_i$ and let E_i be its edge set (as $N_{Y_0}(Y_i) = N(Y_i)$, we shall omit the subscript Y_0). For a vertex $v \in V(G)$, we define the set $E(v)$ as the set of edges incident with v. The notations G'_i, E'_i, Y'_i, and $E'(v)$ are defined analogously for the graph G' obtained after the application of twin elimination rule. We say that a vertex $v \in V(G)$ is *covered* by an edge set D if v is incident with an edge of D.

To see the forward direction, suppose that (G, k) is a Yes-instance and let D be an edge dominating set of size at most k. Without loss of generality, we can assume that $|D \cap E_i| \leqslant |N(Y_i)|$. Indeed, it can be easily checked that the edge set $(D \setminus E_i) \cup E(u)$, for an arbitrarily chosen $u \in Y_i$, is an edge dominating set. Hence at most $|N(Y_i)|$ vertices out of Y_i are covered by D, and thus we can apply twin elimination rule so as to delete only those vertices which are not incident with D. It just remains to observe that D is an edge dominating set of G'.

For the opposite direction, let D' be an edge dominating set for G' of size at most k. We first argue that $N(Y'_i)$ is covered by D' without loss of generality. Indeed, suppose $v \in N(Y'_i)$ is not covered by D'. In order for an edge $e = uv \in E'(v) \cap E'_i$ to be dominated by D', at least one edge in $E'(u)$ should be contained in D. Since the sets $\{E'(u) : u \in Y'_i\}$ are mutually disjoint, it follows that $|D' \cap E'_i| \geqslant |Y'_i|$. Now take an alternative edge set $D'' := (D' \setminus E'_i) \cup E'(u)$ for an arbitrary vertex $u \in Y'_i$. It is not difficult to see that D'' is an edge dominating set for G'. Moreover, we have $|D''| \leqslant |D'| \leqslant k$ as $|D' \cap E'_i| \geqslant |Y'_i| = |E'(u)| = |N(Y'_i)|$. Hence D'' is also an edge dominating set of size at most k. Assuming that $N(Y'_i)$ is covered by D', it is easy to see that D' dominates E_i and thus D' is an edge dominating set of G. This completes the proof. □

Back to the partition $\mathcal{P}$, we can apply the twin elimination rule in time $\mathcal{O}(n)$ and ensure that $|Y_i| \leqslant r-1$ for $1 \leqslant i \leqslant \ell$. The bound on ℓ is proved in Lemma 5.11 and taken together with the edge- and clique-bounds from Proposition 5.2 and 5.3, respectively, we obtain

$$\begin{aligned} \ell &\leqslant 2^{\tau r \log r}((2\beta r^2+1)4k) + 2\beta r^2((2\beta r^2+1)4k) + (2\beta r^2+1)4k + 1 \\ &= 4k(\beta r^2 2^{\tau r \log r+1} + 4\beta^2 r^4 + 4\beta r^2 + 2^{\tau r \log r} + 1) + 1 \end{aligned}$$

and thus we get the overall bound

$$\begin{aligned} |G| &\leqslant |Y_0| + |Y_1| + \cdots + |Y_\ell| \\ &\leqslant 4k(\beta r^2+1) + \left(4k(\beta r^2 2^{\tau r \log r+1} + 4\beta^2 r^4 + 4\beta r^2 + 2^{\tau r \log r} + 1) + 1\right)(r-1) \\ &\leqslant 4k((\beta r^2 2^{\tau r \log r+1} + 4\beta^2 r^4 + 4\beta r^2 + 2^{\tau r \log r} + 1)(r-1) + 2\beta r^2 + 1) + (r-1) \\ &< k\left((80r^2 20.8^{r \log r+1} + 6400r^4 + 320r^2 + 4 \cdot 28.8^{r \log r} + 4)(r-1) + 160r^2 + 4\right) + r \end{aligned}$$

on the size of G. We remark that this upper bound can be easily made explicit once H is fixed. Again, we can get better constants on H-minor-free graphs, just by replacing constants βr^2 and $2^{\tau r \log r}$ with $\alpha(r\sqrt{\log r})$ and $2^{\mu r \log\log r}$, respectively. Finally, note that the whole procedure can be carried out in linear time.

5.5 Single-exponential algorithm for Planar-$\mathcal{F}$-Deletion

This section is devoted to the single-exponential algorithm for the Planar-$\mathcal{F}$-Deletion problem. Let henceforth H_p be some fixed (connected or disconnected) arbitrary planar graph in the family $\mathcal{F}$, and let $r := |H_p|$. First of all, using iterative compression, we reduce the problem to obtaining a single-exponential algorithm for the Disjoint Planar-$\mathcal{F}$-Deletion problem, which is defined as follows:

Disjoint Planar-$\mathcal{F}$-Deletion	
Input:	A graph G and a subset of vertices $X \subseteq V(G)$ such that $G - X$ is H-minor-free for every $H \in \mathcal{F}$.
Parameter:	The integer k.
Objective:	Compute a set $\tilde{X} \subseteq V(G)$ disjoint from X such that $\lvert\tilde{X}\rvert < \lvert X\rvert$ and $G - \tilde{X}$ is H-minor-free for every $H \in \mathcal{F}$, if such a set exists.

The input set X is called the *initial solution* and the set $\tilde{X}$ the *alternative solution*. Let $t_{\mathcal{F}}$ be a constant (depending on the family $\mathcal{F}$) such that $\mathsf{tw}(G - X) \leqslant t_{\mathcal{F}} - 1$ (note that such a constant exists by Robertson and Seymour [222], and can be calculated by using the results of Fellows and Langston [125]).

The following lemma relies on the fact that being $\mathcal{F}$-minor-free is a hereditary property with respect to induced subgraphs. For a proof, see for instance [83, 182, 202, J14].

Lemma 5.14 *If the parameterized* Disjoint Planar-$\mathcal{F}$-Deletion *problem can be solved in time $c^k \cdot p(n)$, where c is a constant and $p(n)$ is a polynomial in n, then the parameterized* Planar-$\mathcal{F}$-Deletion *problem can be solved in time $(c+1)^k \cdot p(n) \cdot n$.*

Let us provide a brief sketch of our algorithm to solve Disjoint Planar-$\mathcal{F}$-Deletion. We start by computing a protrusion decomposition using Algorithm 1 with input (G, X, r). But it turns out that the set Y_0 output by Algorithm 1 does not define a *linear* protrusion decomposition of G, which is crucial for our purposes (in fact, it can be only proved that Y_0 defines a *quadratic* protrusion decomposition of G). To circumvent this problem, our strategy is to first use Algorithm 1 to identify a set Y_0 of $\mathcal{O}(k)$ vertices of G, and then guess the intersection I of the alternative solution $\tilde{X}$ with the set Y_0. We prove that if the input is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then $V(\mathcal{M})$ contains a subset I such that the connected components of $G - V(\mathcal{M})$ can be clustered together with respect to their neighborhood in $Y_0 \setminus I$ to form an $(\mathcal{O}(k - |I|), 2t_{\mathcal{F}} + r)$-protrusion decomposition $\mathcal{P}$ of the graph $G - I$. As a result, we obtain Proposition 5.5, which is fundamental in order to prove Theorem 5.2.

Proposition 5.5 (Linear protrusion decomposition) *Let (G, X, k) be a* Yes*-instance of the parameterized* Disjoint Planar-$\mathcal{F}$-Deletion *problem. There exists a $2^{\mathcal{O}(k)} \cdot n$-time algorithm that identifies a set $I \subseteq V(G)$ of size at most k and a $(\mathcal{O}(k), 2t_{\mathcal{F}} + r)$-protrusion decomposition $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $G - I$ such that:*

1. *$X \subseteq Y_0$;*
2. *there exists a set $X' \subseteq V(G) \setminus Y_0$ of size at most $k - |I|$ such that $G - \tilde{X}$, with $\tilde{X} = X' \cup I$, is H-minor-free for every graph $H \in \mathcal{F}$.*

At this stage of the algorithm, we can assume that a subset I of the alternative solution $\tilde{X}$ has been identified, and it remains to solve the instance $(G - I, X, k - |I|)$ of the Disjoint Planar-$\mathcal{F}$-Deletion problem, which comes equipped with a linear protrusion decomposition $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$. In order to solve this problem, we prove the following proposition:

Proposition 5.6 *Let (G, Y_0, k) be an instance of* Disjoint Planar-$\mathcal{F}$-Deletion *and let $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ be an (α, β)-protrusion decomposition of G, for some constant β. There exists an $2^{\mathcal{O}(\ell)} \cdot n$-time algorithm that computes a solution $\tilde{X} \subseteq V(G) \setminus Y_0$ of size at most k if it exists, or correctly decides that there is no such solution.*

The key observation in the proof of Proposition 5.6 is that for every restricted protrusion Y_i, there is a *finite* number of representatives such that any partial solution lying on Y_i can be replaced with one of these while preserving the feasibility of the solution. This follows from the *finite index* of MSO-definable properties (see, e.g., [70]). Then, to solve the problem in single-exponential time we can just use brute-force in the union of these representatives, which has overall size $\mathcal{O}(k)$.

Organization of the section. In Section 5.5.1 we analyze Algorithm 1 when the input graph is a YES-instance of DISJOINT PLANAR-$\mathcal{F}$-DELETION. The branching step guessing the intersection of the alternative solution $\tilde{X}$ with $V(\mathcal{M})$ is described in Section 5.5.2, concluding the proof of Proposition 5.5. Section 5.5.3 gives a proof of Proposition 5.6, and finally Section 5.5.4 proves Theorem 5.2.

5.5.1 Analysis of the bag marking algorithm

We first need two results concerning graphs which exclude a clique as a minor. The following lemma states that graphs excluding a fixed graph as a minor have linear number of edges.

Proposition 5.7 (Thomason [234]) *There is a constant $\alpha < 0.320$ such that every n-vertex graph with no K_r-minor has at most $(\alpha r\sqrt{\log r}) \cdot n$ edges.*

Recall that a *clique* in a graph is a set of pairwise adjacent vertices. For simplicity, we assume that a single vertex and the empty graph are also cliques.

Proposition 5.8 (Fomin, Oum, and Thilikos [142]) *There is a constant $\mu < 11.355$ such that, for $r > 2$, every n-vertex graph with no K_r-minor has at most $2^{\mu r \log\log r} \cdot n$ cliques.*

For the sake of simplicity, let henceforth in this section $\alpha_r := \alpha r\sqrt{\log r}$ and $\mu_r := 2^{\mu r \log\log r}$.

Let us now analyze some properties of Algorithm 1 when the input graph is a YES-instance of the DISJOINT PLANAR-$\mathcal{F}$-DELETION problem. In this case, the bound on the treewidth of $G - X$ is $t_{\mathcal{F}} - 1$. The following two lemmas show that the number of bags identified at the "Large-subgraph marking step" is linearly bounded by k. Their proofs use arguments similar to those used in the proof of Theorem 5.3, but we provide the full proofs here for completeness.

Lemma 5.15 *Let (G, X, k) be a YES-instance of the DISJOINT PLANAR-$\mathcal{F}$-DELETION problem. If $C_1, \ldots, C_\ell$ is a collection of connected pairwise disjoint subsets of $V(G) \setminus X$ such that for all $1 \leqslant i \leqslant \ell$, $|N_X(C_i)| \geqslant r$, then $\ell \leqslant (1 + \alpha_r) \cdot k$.*

Proof: Let $X' \subseteq V(G) \setminus X$ be a solution for (G, X, k), and observe that $\ell' \leqslant k$ of the sets $C_1, \ldots, C_\ell$ contain vertices of X'. Consider the sets $C_{\ell'+1}, \ldots, C_\ell$ which are disjoint with X', and observe that $G[X \cup (\bigcup_{\ell' < j \leqslant \ell} C_j)]$ is an H-minor-free graph. We proceed to construct a family of graphs $\{G_i\}_{\ell' \leqslant i \leqslant \ell}$, with $V(G_i) = X$ for all $\ell' \leqslant i \leqslant \ell$, and such that G_i is a minor of $G[X \cup (\bigcup_{\ell' < j \leqslant i} C_j)]$, in the following way. We start with $E(G_{\ell'}) = E[G(X)]$, and suppose inductively that the graph G_{i-1} has been successfully constructed. Since by assumption G_{i-1} is a minor of $G[X \cup (\bigcup_{\ell' < j \leqslant i-1} C_j)]$, which in turn is a minor of $G[X \cup (\bigcup_{\ell' < j \leqslant \ell} C_j)]$, it follows that G_{i-1} is H-minor-free, and therefore it cannot contain a clique on r vertices. In order to construct G_i from G_{i-1}, let x_i, y_i be two vertices in X such that both x_i and y_i are neighbors in G of some vertex in C_i, and such that x_i and y_i are non-adjacent in G_{i-1}. Note that such two vertices exist, since we can assume

that $r \geqslant 2$ and G_{i-1} is H-minor-free. Then G_i is constructed from G_{i-1} by adding an edge between x_i and y_i. Since C_i is connected by hypothesis, we have that G_i is indeed a minor of $G[X \cup (\bigcup_{\ell' < j \leqslant i} C_j)]$. Since G_ℓ is H-minor-free, it follows by Proposition 5.7 that $|E(G_\ell)| \leqslant \alpha_r \cdot |X|$ edges. Since by construction we have that $\ell - \ell' \leqslant |E(G_\ell)|$, we conclude that $\ell = \ell' + (\ell - \ell') \leqslant k + \alpha_r \cdot k = (1 + \alpha_r) \cdot k$, as we wanted to prove. □

Lemma 5.16 *If (G, X, k) is a* YES*-instance of the* DISJOINT PLANAR-$\mathcal{F}$-DELETION *problem, then the set $Y_0 = V(\mathcal{M}) \cup X$ of vertices returned by Algorithm 1 has size at most $k + 2t_{\mathcal{F}} \cdot (1 + \alpha_r) \cdot k$.*

Proof: As $|X| = k$ and as the algorithm marks bags of an optimal forest-decomposition of $G - X$, which is a graph of treewidth at most $t_{\mathcal{F}}$, in order to prove the lemma it is enough to prove that the number of marked bags is at most $2 \cdot (1 + \alpha_r) \cdot k$. It is an easy observation to see that the set of connected components C_B identified at the Large-subgraph marking step contains pairwise vertex disjoint subset of vertices, each inducing a connected subgraph of $G - X$ with at least r neighbors in X. It follows by Lemma 5.15, that the number of bags marked at the Large-subgraph marking step is at most $(1+\alpha_r)\cdot k$. To conclude it suffices to observe that the number of bags identified at the LCA marking step cannot exceed the number of bags marked at the Large-subgraph marking step. □

5.5.2 Branching step and linear protrusion decomposition

At this stage of the algorithm, we have identified a set $Y_0 = X \cup V(\mathcal{M})$ of $\mathcal{O}(k)$ vertices such that, by Proposition 5.1, every connected component of $G - Y_0$ is a restricted $(2t_{\mathcal{F}}+r)$-protrusion. We would like to note that it can be proved, using ideas similar to the proof of Lemma 5.17 below, that Y_0 together with the clusters of $G - Y_0$ form a *quadratic* protrusion decomposition of the input graph G. But as announced earlier, for time complexity issues we seek a *linear* protrusion decomposition. To this end, the second step of the algorithm consists in a branching to guess the intersection I of the alternative solution $\tilde{X}$ with the set of marked vertices $V(\mathcal{M})$. By Lemma 5.16, this step yields $2^{\mathcal{O}(k)}$ branchings, which is compatible with the desired single-exponential time.

For each guessed set $I \subseteq Y_0$, we denote $G_I := G - I$. Recall that a cluster of $G_I - Y_0$ as a maximal collection of connected components of $G_I - Y_0$ with the same neighborhood in $Y_0 \setminus I$. We use Observation 5.2, a direct consequence of Lemma 5.7, to bound the number of clusters under the condition that G_I contains a vertex subset X' disjoint from Y_0 of size at most $k - |I|$ such that $G_I - X'$ does not contain any graph $H \in \mathcal{F}$ as a minor (and so the graph $G - \tilde{X}$, with $\tilde{X} = X' \cup I$, does not contain either any graph $H \in \mathcal{F}$ as a minor).

Remark 5.2 *For every cluster $\mathcal{C}$ of $G_I - Y_0$, $|N_{Y_0}(\mathcal{C})| < r + 2t_{\mathcal{F}}$.*

The proof of the following lemma has a similar flavor to those of Theorem 5.3 and Lemma 5.15.

Lemma 5.17 *If $(G_I, Y_0 \setminus I, k - |I|)$ is a* YES*-instance of the* DISJOINT PLANAR-$\mathcal{F}$-DELETION *problem, then the number of clusters of $G_I - Y_0$ is at most $(5t_{\mathcal{F}}\alpha_r\mu_r) \cdot k$.*

Proof: Let $\mathcal{C}$ be the collection of all clusters of $G_I - Y_0$. Let X' be a subset of vertices disjoint from Y_0 such that $|X'| \leqslant k - |I| = k_I$ and $G_I - X'$ is H-minor-free for every graph $H \in \mathcal{F}$. Observe that at most k_I clusters in $\mathcal{C}$ contain vertices from X'. Let $C_1, \ldots, C_\ell$ be the clusters in $\mathcal{C}$ that do not contain vertices from X'. So we have that $|\mathcal{C}| \leqslant k_I + \ell \leqslant k + \ell$. Let $G_{\mathcal{C}}$ be the subgraph of G induced by $(Y_0 \setminus I) \cup \bigcup_{i=1}^{\ell} C_i$. Observe that as $(G_I, Y_0 \setminus I, k - |I|)$ is a YES-instance of the DISJOINT PLANAR-$\mathcal{F}$-DELETION problem, $G_{\mathcal{C}}$ is H-minor-free for every graph $H \in \mathcal{F}$.

We greedily construct from $G_{\mathcal{C}}$ a graph $G'_{\mathcal{C}}$, with $V(G'_{\mathcal{C}}) = Y_0 \setminus I$, as follows. We start with $G'_{\mathcal{C}} = G[Y_0 \setminus I]$. As long as there is a non-used cluster $C \in \mathcal{C}$ with two non-adjacent neighbors u, v in $Y_0 \setminus I$, we add to $G'_{\mathcal{C}}$ an edge between u and v and mark C as used. The number of clusters in $\mathcal{C}$ used so far in the construction of $G'_{\mathcal{C}}$ is bounded from above by the number of edges of $G'_{\mathcal{C}}$. Observe that by construction $G'_{\mathcal{C}}$ is clearly a minor of $G_{\mathcal{C}}$. Thereby $G'_{\mathcal{C}}$ is an H-minor-free graph (for every $H \in \mathcal{F}$) on at most $k + 2t_{\mathcal{F}} \cdot (1 + \alpha_r) \cdot k$ vertices (by Lemma 5.16). By Proposition 5.7, it follows that $|E(G'_{\mathcal{C}})| \leqslant \alpha_r \cdot (k + 2t_{\mathcal{F}} \cdot (1 + \alpha_r) \cdot k)$ and so there are the same number of used clusters.

Let us now count the number of non-used clusters. Observe that the neighborhood in $Y_0 \setminus I$ of each non-used cluster induces a (possibly empty) clique in $G'_{\mathcal{C}}$ (as otherwise some further edge could have been added to $G'_{\mathcal{C}}$). As by definition distinct clusters have distinct neighborhoods in $Y_0 \setminus I$, and as $G'_{\mathcal{C}}$ is an H-minor-free graph (for every $H \in \mathcal{F}$) on at most $k + 2t_{\mathcal{F}} \cdot (1 + \alpha_r) \cdot k$ vertices, Proposition 5.8 implies that the number of non-used clusters is at most $\mu_r \cdot (k + 2t_{\mathcal{F}} \cdot (1 + \alpha_r) \cdot k)$. Summarizing, we have that $|\mathcal{C}| \leqslant k + (\alpha_r + \mu_r) \cdot (k + 2t_{\mathcal{F}} \cdot (1 + \alpha_r) \cdot k) \leqslant (5t_{\mathcal{F}} \alpha_r \mu_r) \cdot k$, where in the last inequality we have used that $\mu_r \geqslant \alpha_r$ and we have assumed that $\alpha_r \geqslant 4$. □

Piecing all lemmas together, we can now provide a proof of Proposition 5.5.

Proof of Proposition 5.5: By Lemmas 5.5 and 5.16 and Observation 5.2, we can compute in linear time a set Y_0 of $\mathcal{O}(k)$ vertices containing X such that every cluster of $G - Y_0$ is a restricted $(2t_{\mathcal{F}} + r)$-protrusion. If (G, X, k) is a YES-instance of the DISJOINT PLANAR-$\mathcal{F}$-DELETION problem, then there exists a set $\tilde{X}$ of size less than $|X|$ and disjoint from X such that $G - \tilde{X}$ does not contain any graph $H \in \mathcal{F}$ as a minor. Branching on every possible subset of $Y_0 \setminus X$, one can guess the intersection I of $\tilde{X}$ with $Y_0 \setminus X$. By Lemma 5.16, the branching degree is $2^{\mathcal{O}(k)}$. As (G, X, k) is a YES-instance, for at least one of the guessed subsets I, the instance $(G_I, Y_0 \setminus I, k - |I|)$ is a YES-instance of the DISJOINT PLANAR-$\mathcal{F}$-DELETION problem. By Lemma 5.17, the partition $\mathcal{P} = (Y_0 \setminus I) \uplus Y_1 \uplus \cdots \uplus Y_\ell$, where $\{Y_1, \ldots, Y_\ell\}$ is the set of clusters of $G_I - Y_0$, is an $(\mathcal{O}(k), r + 2t_{\mathcal{F}})$-protrusion decomposition of G_I. □

5.5.3 Solving Planar-$\mathcal{F}$-Deletion with a linear protrusion decomposition

After having proved Proposition 5.5, we can now focus in this subsection on solving DISJOINT PLANAR-$\mathcal{F}$-DELETION in single-exponential time when a linear protrusion decomposition is given. Let $P_\Pi(G, S)$ denote an MSO formula (of bounded size) which holds if and only if $G - S$ is $\mathcal{F}$-minor-free.

Consider an instance (G, Y_0, k) of Disjoint Planar-$\mathcal{F}$-Deletion equipped with a linear protrusion decomposition $\mathcal{P}$ of G. Let $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ be an (α, β)-protrusion decomposition of G for some constant β. The key observation is that for every restricted protrusion Y_i, there is a finite number of representatives such that any partial solution lying on Y_i can be replaced with one of these while preserving the feasibility of the solution.

We fix a constant t. Let $\mathcal{U}_t$ be the universe of t-boundaried graphs, and let $\mathcal{U}_t^{\text{small}}$ denote the universe of t-boundaried graphs having a tree-decomposition of width $t-1$ with all boundary vertices contained in one bag. Throughout this subsection we will assume that all the restricted protrusions belonging to a given protrusion decomposition have the same boundary size, equal to the maximum boundary size over all protrusions. This assumption is licit as if some protrusion has smaller boundary size, we can add dummy independent vertices to it without interfering with the structure of the solutions and without increasing the treewidth.

Definition 5.11 *Let $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ be an (α, t)-protrusion decomposition of G. For each $1 \leqslant i \leqslant \ell$, we define the following equivalence relation $\sim_{\mathcal{F},i}$ on subsets of Y_i: for $Q_1, Q_2 \subseteq Y_i$, we define $Q_1 \sim_{\mathcal{F},i} Q_2$ if for every $H \in \mathcal{U}_t$, $G[Y_i^+ \setminus Q_1] \oplus H$ is $\mathcal{F}$-minor-free if and only if $G[Y_i^+ \setminus Q_2] \oplus H$ is $\mathcal{F}$-minor-free.*

Note that G equipped with $\mathcal{P}$ can be viewed as a gluing of two β-boundaried graphs $G[Y_i^+]$ and $G \ominus G[Y_i^+]$, for any $1 \leqslant i \leqslant \ell$, where $Y_i^+ = N_{G_I}[Y_i]$. Let us consider the equivalence relation $\sim_{\mathcal{F},i}$ applied on Y_i when G is viewed as such gluing. Extending the notation suggested in Section 5.2, we say that $\mathcal{S}$ is a *set of minimum-sized representatives* of the equivalence relation $\approx$ if $\mathcal{S}$ contains exactly one element of minimum cardinality from every equivalence class under $\approx$. Let $\mathcal{R}(Y_i) := \{Q_1^i, \ldots, Q_{q_i}^i\}$ be a set of minimum-sized representatives of equivalence classes under $\sim_{\mathcal{F},i}$ for every $1 \leqslant i \leqslant \ell$. We say that a set $\tilde{X} \subseteq V(G) \setminus Y_0$ is *decomposable* if $\tilde{X} = Q^1 \cup \cdots \cup Q^\ell$ for some $Q^i \in \mathcal{R}(Y_i)$ for $1 \leqslant i \leqslant \ell$.

Lemma 5.18 (Solution decomposability) *Let (G, Y_0, k) be an instance of* Disjoint Planar-$\mathcal{F}$-Deletion *and let $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ be an (α, β)-protrusion decomposition of G. Then, there exists a solution $\tilde{X} \subseteq V(G) \setminus Y_0$ of size at most k if and only if there exists a decomposable solution $\tilde{X}^* \subseteq V(G) \setminus Y_0$ of size at most k.*

Proof: Let $\tilde{X}$ be a subset of $V(G) \setminus Y_0$. Let $S_i := \tilde{X} \cap Y_i$ for every $1 \leqslant i \leqslant \ell$, $\bar{S}_i := \tilde{X} \cap (V(G) \setminus Y_i)$ and let $H := G \ominus G[Y_i^+] - \bar{S}_i$ be the associated t-boundaried graph with $\mathbf{bd}(H) := \mathbf{bd}(G \ominus G[Y_i^+])$. Fix i and choose the (unique) representative $Q^i \in \mathcal{R}(Y_i)$ such that $Q^i \sim_{\mathcal{F},i} S_i$. Note that $S_i \cap \mathbf{bd}(H) = \bar{S}_i \cap \mathbf{bd}(H) = \emptyset$.

We claim that $G - \tilde{X}$ is $\mathcal{F}$-minor-free if and only if $G - (Q^i \cup \bar{S}_i)$ is $\mathcal{F}$-minor-free. Indeed, $G - \tilde{X} = G - (S_i \cup \bar{S}_i)$, which can be written as $G[Y_i^+ \setminus S_i] \oplus H$. From the choice of $Q^i \in \mathcal{R}(Y_i)$ such that $Q^i \sim_{\mathcal{F},i} S_i$, it follows that $G[Y_i^+ \setminus S_i] \oplus H$ is $\mathcal{F}$-minor-free if and only if $G[Y_i^+ \setminus Q^i] \oplus H$ is so. Noting that $G[Y_i^+ \setminus Q^i] \oplus H = G - (Q^i \cup \bar{S}_i)$ proves our claim.

By replacing each S_i with its representative $Q^i \in \mathcal{R}(Y_i)$, we eventually obtain $\tilde{X}^*$ of the form $\tilde{X}^* = \bigcup_{1 \leqslant i \leqslant \ell} Q^i$, where $Q^i \in \mathcal{R}(Y_i)$ is the representative of S_i for every $1 \leqslant i \leqslant \ell$. Finally, it holds that $P_\Pi(G, \tilde{X}^*)$ if and only if $P_\Pi(G, \tilde{X})$. It remains to observe that $|Q^i| \leqslant |S_i|$, as we selected a minimum-sized set of an equivalence class of $\sim_{\mathcal{F},i}$ as its representative. □

We are now ready to prove Proposition 5.6.

Reminder of Proposition 5.6. *Let (G, Y_0, k) be an instance of* Disjoint Planar-$\mathcal{F}$-Deletion *and let $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ be an (α, β)-protrusion decomposition of G, for some constant β. There exists an $2^{\mathcal{O}(\ell)} \cdot n$-time algorithm that computes a solution $\tilde{X} \subseteq V(G) \setminus Y_0$ of size at most k if it exists, or correctly decides that there is no such solution.*

Proof: By Lemma 5.18, either there exists a solution of size at most k which is decomposable or (G, Y_0, k) is a No-instance. Henceforth we will just look for decomposable solutions.

We first need to compute, for all $1 \leqslant i \leqslant \ell$, a set of representatives $\mathcal{R}(Y_i)$ of the equivalence relations $\sim_{\mathcal{F},i}$. To that aim, we define an equivalence relation $\equiv_{\mathcal{F},t}$ on $\mathcal{U}_t^{\text{small}}$, with the objective of capturing all possible behaviors of the graphs $G \ominus G[Y_i^+]$ (we call such graphs the *context* of the restricted protrusion Y_i). For two t-boundaried graphs K_1 and K_2 from $\mathcal{U}_t^{\text{small}}$, we say that:

$K_1 \equiv_{\mathcal{F},t} K_2$ *if for every* $Y \in \mathcal{U}_t^{small}$, $K_1 \oplus Y$ *is* $\mathcal{F}$*-minor-free iff* $K_2 \oplus Y$ *is* $\mathcal{F}$*-minor-free.*

As the property of being $\mathcal{F}$-minor-free is MSO-definable, $\equiv_{\mathcal{F},t}$ has finitely many equivalence classes for each fixed t and there is a finite set $\mathcal{K} = \{K_1, \ldots, K_M\}$ of representatives of $\equiv_{\mathcal{F},t}$ (cf. for instance [91, 115]). Observe that the set $\mathcal{K}$ is independent from the instance (it depends only on the problem). For the sake of readability we now assume that the set $\mathcal{K}$ is given. Note that this assumption would make the proof *non-constructive* and therefore the algorithm *non-uniform* on the family $\mathcal{F}$ (that is, for each family $\mathcal{F}$ we would deduce the existence of a different algorithm). In the paragraph below the end of the proof we briefly explain how this set $\mathcal{K}$ can be efficiently constructed in linear time, yielding a constructive and uniform algorithm.

So given the set $\mathcal{K} = \{K_1, \ldots, K_M\}$, we now proceed to find the set of representatives $\mathcal{R}(Y_i)$ in time $\mathcal{O}(|Y_i|)$ for every $1 \leqslant i \leqslant \ell$. Our strategy is inspired by the method of test sets [39]. We consider the set of all binary vectors with M coordinates, to which we give the following interpretation. Each fixed such vector $\mathbf{v} = (b_1, \ldots, b_M)$ will correspond to a minimum-sized subset $Q_{\mathbf{v}} \subseteq Y_i$ such that, for $1 \leqslant j \leqslant M$, the graph $G[(Y_i^+ \setminus Q_{\mathbf{v}})] \oplus K_j$ is $\mathcal{F}$-minor-free iff $b_j = 1$. Formally, for each binary vector $(b_1, \ldots, b_M)$ of length M we consider the following optimization problem: find a set $Q \subseteq Y_i$ of minimum size such that $\varphi(G[Y_i^+], Q)$ holds. Here $\varphi(G[Y_i^+], Q) := (Q \subseteq Y_i) \wedge \left(\bigwedge_{j=1}^{M} \tilde{b}_j\right)$, where $\tilde{b}_j := \varphi_{K_j}(G[Y_i^+], Q)$ if $b_j = 1$ and $\tilde{b}_j := \neg\varphi_{K_j}(G[Y_i^+], Q)$ if $b_j = 0$, each $\varphi_{K_j}(G[Y_i^+], Q)$ stating that $G[(Y_i^+ \setminus Q)] \oplus K_j$ is $\mathcal{F}$-minor-free. For each fixed $K_j \in \mathcal{U}_t^{\text{small}}$, whether $G[(Y_i^+ \setminus Q)] \oplus K_j$ is $\mathcal{F}$-minor-free or not depends only on $G[Y_i^+]$ and Q, and moreover this property can be expressed as an MSO formula. As φ is an MSO formula, we can apply the linear-time dynamic programming algorithm of Borie et al. [72] on graphs of bounded treewidth to solve the associated optimization problem. Note that the running time is $\mathcal{O}(|Y_i|)$, whose hidden constant depends solely on $|P_\Pi|$ and the treewidth t.

Claim 5.1 *Let $\mathcal{R}_i$ be the set of the optimal solutions over all 2^M binary vectors of length M, obtained as explained above. Then $\mathcal{R}_i$ is a set of minimum-sized representatives of $\sim_{\mathcal{F},i}$.*

Proof: We fix $t := 2t_{\mathcal{F}} + r$, so we can assume that all protrusions Y_i^+ belong to $\mathcal{U}_t^{\text{small}}$. First note that in the definition the equivalence relation $\sim_{\mathcal{F},i}$ (cf. Definition 5.11), one only needs to consider graphs $H \in \mathcal{U}_t^{\text{small}}$. Indeed, if $H \in \mathcal{U}_t \setminus \mathcal{U}_t^{\text{small}}$, then for any $Q \subseteq Y_i$ it follows that $G[Y_i^+ \setminus Q] \oplus H$ is *not* $\mathcal{F}$-minor-free (as $\mathsf{tw}(G[Y_i^+ \setminus Q] \oplus H) \geqslant 2t_{\mathcal{F}} + r > t_{\mathcal{F}}$), so in order to define the equivalence classes of subsets of Y_i it is enough to consider $H \in \mathcal{U}_t^{\text{small}}$. In other words, only the elements of $\mathcal{U}_t^{\text{small}}$ can *distinguish* the subsets of Y_i with respect to $\sim_{\mathcal{F},i}$.

Let $Q \subseteq Y_i$, and we want to prove that there exists $R_Q \in \mathcal{R}_i$ such that $Q \sim_{\mathcal{F},i} R_Q$, that is, such that for any $H \in \mathcal{U}_t$, $G[Y_i^+ \setminus Q] \oplus H$ is $\mathcal{F}$-minor-free iff $G[Y_i^+ \setminus R_Q] \oplus H$ is $\mathcal{F}$-minor-free. By the remark in the above paragraph, we can assume that $H \in \mathcal{U}_t^{\text{small}}$, as otherwise the statement is trivially true. Let $\mathbf{v}_Q = (b_1, \ldots, b_M)$ be the binary vector on M coordinates such that, for $1 \leqslant j \leqslant M$, $b_j = 1$ iff $G[(Y_i^+ \setminus Q) \oplus K_j]$ is $\mathcal{F}$-minor-free. We define R_Q to be the graph in $\mathcal{R}_i$ corresponding to the vector $\mathbf{v}_Q$. As $H \in \mathcal{U}_t^{\text{small}}$, there exists $K_H \in \mathcal{K}$ such that $H \equiv_{\mathcal{F},t} K_H$. Then, $G[Y_i^+ \setminus Q] \oplus H$ is $\mathcal{F}$-minor-free iff $G[Y_i^+ \setminus Q] \oplus K_H$ is $\mathcal{F}$-minor-free, which by construction is $\mathcal{F}$-minor-free iff $G[Y_i^+ \setminus R_Q] \oplus K_H$ is $\mathcal{F}$-minor-free, which is in turn $\mathcal{F}$-minor-free iff $G[Y_i^+ \setminus R_Q] \oplus H$ is $\mathcal{F}$-minor-free, as we wanted to prove. □

To summarize the above discussion, a set of minimum-sized representatives $\mathcal{R}(Y_i)$ can be constructed in time $\mathcal{O}(|Y_i|)$, and therefore all the sets of such representatives can be constructed in time $\mathcal{O}(n)$.

Once the sets $\mathcal{R}(Y_i)$ of representatives for the equivalence relations $\sim_{\mathcal{F},i}$ have been computed for all $1 \leqslant i \leqslant \ell$, it remains to test every possible decomposable set $\tilde{X}$ (see Lemma 5.18). Since we have computed a *minimum-sized* set of representatives $\mathcal{R}(Y_i)$ for each $\sim_{\mathcal{F},i}$, it follows that there exists a solution $\tilde{X} \subseteq V(G) \setminus Y_0$ of size at most k if and only if there exists a decomposable solution $\tilde{X}^* \subseteq V(G) \setminus Y_0$ of size at most k which is made of representatives from the sets $\mathcal{R}(Y_i)$. Observe that for a given decomposable set $\tilde{X}$, one can decide if $\tilde{X}$ is a solution or not in time $\mathcal{O}(h(t_{\mathcal{F}}) \cdot n)$. Indeed, for $\tilde{X}$ to be a solution, the treewidth of $G - \tilde{X}$ is at most $t_{\mathcal{F}} - 1$. Using the algorithm of Bodlaender [58], one can decide in time $2^{\mathcal{O}(t_{\mathcal{F}}^3)} \cdot n$ whether a graph is of treewidth at most $t_{\mathcal{F}}$ and if so, build a tree-decomposition of width at most $t_{\mathcal{F}}$. Courcelle's theorem [90] says that testing an MSO-definable property on treewidth-$t_{\mathcal{F}}$ graphs can be done in linear time, where the hidden constant depends solely on the treewidth $t_{\mathcal{F}}$ and the length of the MSO-sentence. It follows that one can decide whether $G - \tilde{X}$ is $\mathcal{F}$-minor-free or not in time $\mathcal{O}(h(t_{\mathcal{F}}) \cdot n)$. Here $h(t_{\mathcal{F}})$ is an additive function resulting from Bodlaender's treewidth testing algorithm and Courcelle's MSO-model checking algorithm, which depends solely on the treewidth $t_{\mathcal{F}}$ and the MSO formula $|P_\Pi(G, \tilde{X})|$. It remains to observe that there are at most $2^{\mathcal{O}(\ell)}$ decomposable sets to consider. This is because an MSO-definable graph property has finitely many equivalence classes on $\mathcal{U}_t$ for every fixed t [70,90] (hence, also on $\mathcal{U}_t^{\text{small}}$), and being $\mathcal{F}$-minor-free is an MSO-definable property. □

Constructing the sets of representatives for $\equiv_{\mathcal{F},t}$. Let $\Phi_{\mathcal{F}}$ be the MSO-formula expressing that a graph is $\mathcal{F}$-minor free. The set $\mathcal{K}$ of representatives can be efficiently constructed on the universe $\mathcal{U}_t^{\text{small}}$ from $\Phi_{\mathcal{F}}$ and the boundary size t. Let us discuss the

main line of the proof of this fact. Courcelle's theorem is proved[9] by converting an MSO formula φ on tree-decompositions of width t into another MSO formula φ' on labeled trees. Trees, in which every internal node has bounded *fan-in* and every node is labeled with an alphabet chosen from a fixed set, are considered as a *tree language*, which is a natural generalization of the usual string language. It is well-known (as the analogue of Büchi's theorem [74] on tree languages) that the set of labeled trees for which an MSO formula holds form a regular (tree) language[10]. Moreover, based on its proof it is not difficult to construct a finite tree automaton (cf. for instance [131]). In particular, the number of states in the corresponding tree automaton is bounded by a constant depending only on φ and t. From this, it is possible to prove (using a "tree" pumping lemma) that one can assume that the height of a *distinguishing extension* of two labeled trees is bounded by a constant as well (in fact, the size of the tree automaton). Hence we can enumerate all possible labeled trees of bounded height, which will be a test set to construct the set of representatives $\mathcal{K}$. Now one can apply the so-called *method of test sets* (basically implicit in the proof of the Myhill-Nerode theorem [211], see [39, 115] for more details) and retrieve the set of representatives $\mathcal{K}$. Finally, the reader can check that each of these standard procedures can be implemented in time $\mathcal{O}(n)$, which implies that the algorithm of Proposition 5.6 has overall running time $2^{\mathcal{O}(\ell)} \cdot n$. (We note that an approach similar to the one described here can be found in [238, Corollary 3.13].)

5.5.4 Proof of Theorem 5.2

We finally have all the ingredients to prove Theorem 5.2.

Reminder of Theorem 5.2. *The parameterized* Planar-$\mathcal{F}$-Deletion *problem can be solved in time* $2^{\mathcal{O}(k)} \cdot n^2$.

Proof: Lemma 5.14 states that Planar-$\mathcal{F}$-Deletion can be reduced to Disjoint Planar-$\mathcal{F}$-Deletion so that the former can be solved in single-exponential time solvable provided that the latter is so, and the degree of the polynomial function just increases by one. We now proceed to solve Disjoint Planar-$\mathcal{F}$-Deletion in time $2^{\mathcal{O}(k)} \cdot n$. Given an instance (G, X, k) of Disjoint Planar-$\mathcal{F}$-Deletion, we apply Proposition 5.5 to either correctly decide that (G, X, k) is a No-instance, or identify in time $2^{\mathcal{O}(k)} \cdot n$ a set $I \subseteq V(G)$ of size at most k and a $(\mathcal{O}(k), 2t_{\mathcal{F}} + r)$-protrusion decomposition $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $G - I$, with $X \subseteq Y_0$, such that there exists a set $X' \subseteq V(G) \setminus Y_0$ of size at most $k - |I|$ such that $G - \tilde{X}$, with $\tilde{X} = X' \cup I$, is H-minor-free for every graph $H \in \mathcal{F}$. Finally, using Proposition 5.6 we can solve the instance $(G_I, Y_0 \setminus I, k - |I|)$ in time $2^{\mathcal{O}(k)} \cdot n$. □

[9] In fact, there is more than one proof of Courcelle's theorem. The one we depict here is as presented in [131], which differs from the original proof of Courcelle [90].

[10] A regular tree language is an analogue of a regular language on labeled trees. Appropriately defined, most of the nice properties on string regular languages transfer immediately to tree regular languages. It is beyond the scope of this chapter to give details about tree languages and tree automatons. We invite the interested readers to [91, 115].

5.6 Some deferred results

In this section we provide some results that have been deferred in the current chapter.

5.6.1 Edge modification problems are not minor-closed

A graph problem Π is *minor-closed* if whenever G is a YES-instance of Π and G' is a minor of G, then G' is also a YES-instance of Π. It is easy to see that $\mathcal{F}$-(VERTEX-)DELETION is minor-closed, and therefore it is FPT by Robertson and Seymour [226]. Here we show that the edge modification versions, namely, $\mathcal{F}$-EDGE-CONTRACTION and $\mathcal{F}$-EDGE-REMOVAL (defined in the natural way), are not minor-closed.

Edge contraction. In this case, the problem Π is whether one can contract at most k edges from a given graph G so that the resulting graph does not contain any of the graphs in $\mathcal{F}$ as a minor. Let $\mathcal{F} = \{K_5, K_{3,3}\}$, and let G be the graph obtained from K_5 by subdividing every edge k times, and adding an edge e between two arbitrary original vertices of K_5. Then G can be made planar just by contracting edge e, but if G' is the graph obtained from G by deleting e (which is a minor of G), then at least $k+1$ edge contractions are required to make G' planar.

Edge deletion. In this case, the problem Π is whether one can delete at most k edges from a given graph G so that the resulting graph does not contain any of the graphs in $\mathcal{F}$ as a minor. Let G, G', and H be the graphs depicted in Figure 5.4, and let $k = 1$. Then G can be made H-minor-free by deleting edge e, but G', which is the graph obtained from G by contracting edge e, needs at least two edge deletions to be H-minor-free.

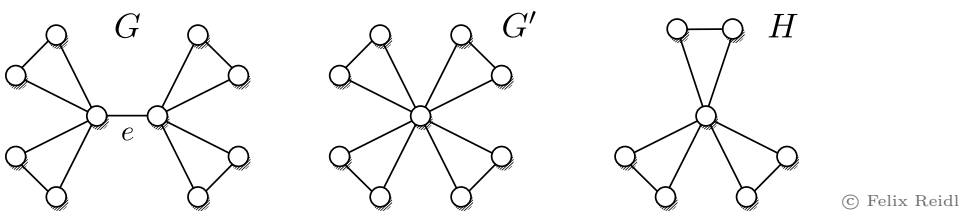

Figure 5.4: Example to show that $\mathcal{F}$-EDGE-REMOVAL is not minor-closed.

5.6.2 Disconnected planar obstructions

Let us argue that there exist natural obstruction sets that contain disconnected planar graphs. Following Dinneen [107], given an integer $\ell \geqslant 0$ and a graph invariant function λ that maps graphs to integers such that whenever $H \preceq_m G$ we also have $\lambda(H) \leqslant \lambda(G)$, we say that the graph class $\mathcal{G}_\lambda^\ell := \{G : \lambda(G) \leqslant \ell\}$ is an *ℓ-parameterized lower ideal*. By Robertson and Seymour [226], we know that for each ℓ-parameterized lower ideal $\mathcal{G}_\lambda^\ell$ there exists a finite graph family $\mathcal{F}$ such that $\mathcal{G}_\lambda^\ell$ has precisely $\mathcal{F}$ as (minor) obstruction set. In this setting, the $\mathcal{F}$-DELETION problem (parameterized by k) asks whether k vertices

can be removed from a graph G so that the resulting graph belongs to the corresponding ℓ-parameterized lower ideal $\mathcal{G}_\lambda^\ell$. For instance, the parameterized Feedback Vertex Set problem corresponds to the 0-parameterized lower ideal with graph invariant **fvs**, namely $\mathcal{G}_{\mathbf{fvs}}^0$, which is characterized by $\mathcal{F} = \{K_3\}$ and therefore $\mathcal{G}_{\mathbf{fvs}}^0$ is the set of all forests. Interestingly, it is proved in [107] that for $\ell \geqslant 1$, the obstruction set of many interesting graph invariants (such as ℓ-Vertex Cover, ℓ-Feedback Vertex Set, or ℓ-Face Cover to name just a few) contains the disjoint union of obstructions for $\ell - 1$. As for the above-mentioned problems there is a planar obstruction for $\ell = 0$, we conclude that for $\ell \geqslant 1$ the corresponding family $\mathcal{F}$ contains *disconnected* planar obstructions.

5.6.3 Disconnected Planar-$\mathcal{F}$-Deletion has not finite integer index

We proceed to prove that if $\mathcal{F}$ is a family of graphs containing some disconnected graph H (planar or non-planar), then the $\mathcal{F}$-Deletion problem has not finite integer index (FII) in general.

We shall use the equivalent definition of FII as suggested for graph optimization problems, see [99]. For a graph problem o-Π, the equivalence relation $\sim_{o\text{-}\Pi,t}$ on t-boundaried graphs is defined as follows. Let G_1 and G_2 be two t-boundaried graphs. We define $G_1 \sim_{\Pi,t} G_2$ if and only if there exists an integer i such that for any t-boundaried graph H, it holds $\pi(G_1 \oplus H) = \pi(G_2 \oplus H) + i$, where $\pi(G)$ denotes the optimal value of problem o-Π on graph G. We claim that $G_1 \sim_{\Pi,t} G_2$ if and only if $G_1 \equiv_{\Pi,t} G_2$ (recall Definition 5.5 of canonical equivalence), where Π is the parameterized version of o-Π with the solution size as the parameter. Suppose $G_1 \sim_{\Pi,t} G_2$ and let $\pi(G_1 \oplus H) = \pi(G_2 \oplus H) + i$. Then

$$(G_1 \oplus H, k) \in \Pi \;\Leftrightarrow\; \pi(G_1 \oplus H) \leqslant k \;\Leftrightarrow\; \pi(G_2 \oplus H) \leqslant k - i (G_2 \oplus H, k - i) \in \Pi,$$

and thus the forward implication holds. The opposite direction is easy to see.

Let F_1 and F_2 be two incomparable graphs with respect to the minor relation, and let F be the disjoint union of F_1 and F_2. For instance, if we want F to be planar, we can take $F_1 = K_4$ and $F_2 = K_{2,3}$. We set $\mathcal{F} = \{F\}$. Let Π be the non-parameterized version of $\mathcal{F}$-Vertex Deletion.

For $i \geqslant 1$, let G_i be the 1-boundaried graph consisting of the boundary vertex v together with i disjoint copies of F_1, and for each such copy, we add and edge between v and an arbitrary vertex of F_1. Similarly, for $j \geqslant 1$, let H_j be the 1-boundaried graph consisting of the boundary vertex u together with j disjoint copies of F_2, and for each such copy, we add and edge between u and an arbitrary vertex of F_2.

By construction, if $i, j \geqslant 1$, it holds $\pi(G_i \oplus H_j) = \min\{i, j\}$. Then, if we take $1 \leqslant n < m$,

$$\begin{aligned} \pi(G_n \oplus H_{n-1}) - \pi(G_m \oplus H_{n-1}) &= (n-1) - (n-1) = 0, \\ \pi(G_n \oplus H_m) - \pi(G_m \oplus H_m) &= n - m < 0. \end{aligned}$$

Therefore, G_n and G_m do not belong to the same equivalence class of $\sim_{\Pi,1}$ whenever $1 \leqslant n < m$, so $\sim_{\Pi,1}$ has infinitely many equivalence classes, and thus Π has not FII.

In particular, the above example shows that if $\mathcal{F}$ may contain some disconnected planar graph H, then Planar-$\mathcal{F}$-Deletion has not FII in general.

5.6.4 MSO formula for topological minor containment

For a fixed graph H we describe and MSO_1-formula Φ_H over the usual structure consisting of the universe $V(G)$ and a binary symmetric relation ADJ modeling $E(G)$ such that $G \models \Phi_H$ iff $H \preceq_{\mathsf{tm}} G$.

$$\begin{aligned}
\Phi_H(G) &:= \exists x_{v_1} \ldots \exists x_{v_r} \exists D_{e_1} \ldots \exists D_{e_\ell} \\
&\Big(\bigwedge_{1 \leqslant i < j \leqslant r} x_{v_i} \neq x_{v_j} \wedge \bigwedge_{\substack{1 \leqslant i \leqslant r \\ 1 \leqslant j \leqslant \ell}} x_{v_i} \notin D_{e_j} \wedge \bigwedge_{1 \leqslant i < j \leqslant r} \mathrm{DIS}(D_{e_i}, D_{e_j}) \wedge \bigwedge_{\substack{1 \leqslant j \leqslant \ell \\ e_j = v_i v_k}} \mathrm{CONN}(x_{v_i}, D_{e_j}, x_{v_k}) \Big) \\
\text{with } \mathrm{DIS}(X,Y) &:= \forall x (x \in X \to x \notin Y) \\
\text{and } \mathrm{CONN}(u,X,v) &:= \exists w (\mathrm{ADJ}(u,w) \wedge w \in X) \wedge \exists w (\mathrm{ADJ}(v,w) \wedge w \in X) \\
&\wedge \forall A \forall B ((A \subseteq X \wedge B \subseteq X \wedge \mathrm{DIS}(A,B)) \to \exists a \exists b (a \in A \wedge b \in B \wedge \mathrm{ADJ}(a,b)))
\end{aligned}$$

The subformula $\mathrm{CONN}(u,X,v)$ expresses that u, v are adjacent to X and that $G[X]$ is connected, which implies that there exists a path from u to v in $G[X \cup \{u,v\}]$. By negation we can now express that G does not contain H as a topological minor, i.e., $G \models \neg\Phi_H$ iff G is H-topological-minor-free.

5.7 Concluding remarks

In this chapter we presented a simple algorithm to compute protrusion decompositions for graphs G that come equipped with a set $X \subseteq V(G)$ such that the treewidth of $G - X$ is at most some fixed constant t. Then we showed that this algorithm can be used in order to achieve two different sets of results: linear kernels on graphs excluding a fixed topological minor, and a single-exponential parameterized algorithm for the Planar-$\mathcal{F}$-Deletion problem.

Concerning our kernelization algorithm, the first main question is whether similar results can be obtained for an even larger class of (sparse) graphs. A natural candidate is the class of graphs of bounded expansion (see [215] for the definition), which strictly contains H-topological-minor-free graphs. Let us now argue that the existence of linear kernels for some of the considered problems on graphs of bounded expansion seems to be as plausible as on general graphs. Indeed, consider for instance the Treewidth-t Vertex Deletion problem, which is clearly treewidth-bounding. Take a general graph G on n vertices as input of Treewidth-t Vertex Deletion, and let G' be the graph obtained from G by subdividing each edge n times. Note that G' has bounded expansion, and that this operation does not increase the treewidth. As far as $t \geqslant 1$, it is easy to see that one can assume that none of the newly added vertices belong to a solution, and thus the size of an optimal solution is the same in G and G'. Therefore, obtaining a kernel for Treewidth-t Vertex Deletion on graphs of bounded expansion is as hard as on general graphs. According to Fomin et al. [137], this problem has a kernel of size $k^{\mathcal{O}(t)}$ on general graphs, and by Giannopoulou et al. [156] no uniform polynomial kernel (that is, a polynomial kernel whose degree does not depend on t) exists unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. Since graphs of bounded expansion strictly contain H-topological-minor-free graphs, and there are not

well-known graph classes in between, our kernelization result may settle the limit of meta-theorems about the existence of linear, or even uniform polynomial, kernels on sparse graph classes.

The second main question is which other problems have linear kernels on H-topological-minor-free graphs. In particular, it has been recently proved by Fomin et al. that (Connected) Dominating Set has a linear kernel on H-minor-free graphs [141] and on H-topological-minor-free graphs [133]. It would be interesting to investigate how the structure theorem by Grohe and Marx [163] can be used in this context.

We would like to note that the degree of the polynomial of the running time of our kernelization algorithm depends linearly on the size of the excluded topological minor H. It seems that the recent *fast protrusion replacer* of Fomin et al. [137] could be applied to get rid of the dependency on H of the running time.

Let us now discuss some further research related to our single-exponential algorithm for Planar-$\mathcal{F}$-Deletion. As mentioned in the introduction, no single-exponential algorithm is known when the family $\mathcal{F}$ does not contain any planar graph. Is it possible to find such a family, or can it be proved that, under some complexity assumption, a single-exponential algorithm is not possible? See [172] for recent advances in this direction. An ambitious goal would be to optimize the constants involved in the function $2^{\mathcal{O}(k)}$, possibly depending on the family $\mathcal{F}$, and maybe even proving lower bounds for such constants, in the spirit of Lokshtanov et al. [199] for problems parameterized by treewidth.

It also makes sense to forbid the family of graphs $\mathcal{F}$ according to another containment relation, like topological minor. Using the fact that if H is a graph with maximum degree at most 3, a graph G contains H as a minor if and only if G contains H as a topological minor (hence, graphs that exclude a planar graph H with maximum degree at most 3 as a topological minor have bounded treewidth), it can be proved that our approach also yields a single-exponential algorithm for Planar-Topological-$\mathcal{F}$-Deletion as far as $\mathcal{F}$ contains some planar (connected or disconnected) graph with maximum degree at most 3.

We showed (in Section 5.5.3) how to obtain single-exponential algorithms for Disjoint Planar-$\mathcal{F}$-Deletion with a given linear protrusion decomposition. This approach seems to be applicable to general vertex deletion problems to attain a property expressible in CMSO (but probably, the fact of having bounded treewidth is needed in order to make the algorithm constructive). It would be interesting to generalize this technique to p-max-CMSO or p-eq-CMSO problems, as well as to edge subset problems.

The running time of the algorithm given in Theorem 5.2 is $2^{\mathcal{O}(k)} \cdot n^2$, which can be improved to $2^{\mathcal{O}(k)} \cdot n \log^2 n$ by using the following trick based on [137]. Let t be a bound on the treewidth of any graph excluding a planar graph in $\mathcal{F}$ as a minor, and let G the the input graph to our problem. Instead of doing iterative compression, we first solve on G in time $2^{\mathcal{O}(k)} \cdot n \log^2 n$ the problem consisting on deleting at most k vertices in order to obtain a graph of treewidth at most t, using the algorithm of [137] (note that we can do so, as all obstructions for treewidth are clearly connected). If we fail, we know that G is a negative instance of Planar-$\mathcal{F}$-Deletion, and we are done. Otherwise, let X be such a set of at most k vertices. Now we can solve Planar-$\mathcal{F}$-Deletion on the graph $G-D$ in time $\mathcal{O}(n)$,

as it has bounded treewidth, and obtain a set X' of size at most k such that $G-(X\cup X')$ is $\mathcal{F}$-minor-free. Finally, we can guess the intersection of the set $X\cup X'$ with the solution of Planar-$\mathcal{F}$-Deletion, and then solve the Disjoint Planar-$\mathcal{F}$-Deletion problem in time $2^{\mathcal{O}(k)}\cdot n$, as it is done in Section 5.5.

In the parameterized dual version of the $\mathcal{F}$-Deletion problem, the objective is to find at least k vertex-disjoint subgraphs of an input graph, each of them containing some graph in $\mathcal{F}$ as a minor. For $\mathcal{F}=\{K_3\}$, the problem corresponds to k-Disjoint Cycle Packing, which does not admit a polynomial kernel on general graphs [69] unless $\mathsf{NP}\subseteq\mathsf{coNP/poly}$. Does this problem, for some non-trivial choice of $\mathcal{F}$, admit a single-exponential parameterized algorithm?

Chapter 6

Explicit linear kernels via dynamic programming

Several algorithmic meta-theorems on kernelization have appeared in the last years, starting with the result of Bodlaender et al. [63] on graphs of bounded genus, then generalized by Fomin et al. [140] to graphs excluding a fixed minor, and by Kim et al. [J15] (see Chapter 5) to graphs excluding a fixed topological minor. Typically, these results guarantee the existence of linear or polynomial kernels on sparse graph classes for problems satisfying some generic conditions but, mainly due to their generality, it is not clear how to derive from them constructive kernels with explicit constants. In this chapter we make a step toward a fully constructive meta-kernelization theory on sparse graphs. Our approach is based on a more explicit protrusion replacement machinery that, instead of expressibility in CMSO logic, uses dynamic programming, which allows us to find an explicit upper bound on the size of the derived kernels. We demonstrate the usefulness of our techniques by providing the first explicit linear kernels for r-Dominating Set and r-Scattered Set on apex-minor-free graphs, and for Planar-$\mathcal{F}$-Deletion on graphs excluding a fixed (topological) minor in the case where all the graphs in $\mathcal{F}$ are connected.

Keywords: parameterized complexity; linear kernels; dynamic programming; protrusion replacement; graph minors.

Contents

6.1 Introduction

As discussed in Section 3.2, a fundamental notion in parameterized complexity is that of *kernelization*, which asks for the existence of polynomial-time preprocessing algorithms that produce equivalent instances whose size depends exclusively (preferably polynomially or event linearly) on the parameter k. Finding kernels of size polynomial or linear in k (called *linear kernels*) is one of the major goals of this area.

An influential work in this direction was the linear kernel of Alber et al. [41] for Dominating Set on planar graphs, which was generalized by Guo and Niedermeier [166] to a family of problems on planar graphs. Several algorithmic meta-theorems on kernelization have appeared in the last years, starting with the result of Bodlaender et al. [63] on graphs of bounded genus. After that, similar results have been obtained on larger sparse graph classes, such as graphs excluding a minor [140] or a topological minor [J15] (see Chapter 5).

Typically, the above results guarantee the *existence* of linear or polynomial kernels on sparse graph classes for a number of problems satisfying some generic conditions but, mainly due to their generality, it is hard to derive from them *constructive* kernels with *explicit* constants. The main reason behind this non-constructibility is that the proofs rely on a property of problems called *Finite Integer Index* (FII) that, roughly speaking, allows to replace large "protrusions" (i.e., large subgraphs with small boundary to the rest of the graph) with "equivalent" subgraphs of constant size. This substitution procedure is known as *protrusion replacer*, and while its *existence* has been proved, so far, there is no generic way to *construct* it. Using the technology developed in [63], there are cases where protrusion replacements can become constructive given the expressibility of the problem in Counting Monadic Second Order (CMSO) logic. This approach is essentially based on extensions of Courcelle's theorem [90] that, even when they offer constructibility, it is hard to extract from them any *explicit constant* that upper-bounds the size of the derived kernel.

Results and techniques. In this chapter we tackle the above issues and make a step toward a fully constructive meta-kernelization theory on sparse graphs with explicit constants. For this, we essentially substitute the algorithmic power of CMSO logic with that of dynamic programming on graphs of bounded decomposability (i.e., bounded treewidth). Our approach provides a dynamic programming framework able to construct a protrusion replacer for a wide variety of problems.

Loosely speaking, the framework that we present can be summarized as follows. First of all, we propose a general definition of a problem encoding for the tables of dynamic programming when solving parameterized problems on graphs of bounded treewidth. Under this setting, we provide general conditions on whether such an encoding can yield a protrusion replacer. While our framework can also be seen as a possible formalization of dynamic programming, our purpose is to use it for constructing protrusion replacement algorithms and linear kernels whose size is explicitly determined.

In order to obtain an explicit linear kernel for a problem Π, the main ingredient is to prove that when solving Π on graphs of bounded treewidth via dynamic programming, we can use tables such that the maximum difference between all the values that need to be

stored is bounded by a function of the treewidth. For this, we prove in Theorem 6.1 that when the input graph excludes a fixed graph H as a (topological) minor, this condition is sufficient for constructing an explicit protrusion replacer algorithm, i.e., a polynomial-time algorithm that replaces a large protrusion with an equivalent one whose size can be bounded by an *explicit* constant. Such a protrusion replacer can then be used, for instance, whenever it is possible to compute a linear protrusion decomposition of the input graph (that is, an algorithm that partitions the graph into a part of size linear in $\mathcal{O}(k)$ and a set of $\mathcal{O}(k)$ protrusions). As there is a wealth of results for constructing such decompositions [63, 137, 140, J15], we can use them as a starting point and, by applying dynamic programming, obtain an explicit linear kernel for Π.

We demonstrate the usefulness of this general strategy by providing the first explicit linear kernels for three distinct families of problems on sparse graph classes. On the one hand, for each integer $r \geqslant 1$, we provide a linear kernel for r-Dominating Set and r-Scattered Set on graphs excluding a fixed apex graph H as a minor. Moreover, for each finite family $\mathcal{F}$ of connected graphs containing at least one planar graph, we provide a linear kernel for Planar-$\mathcal{F}$-Deletion on graphs excluding a fixed graph H as a (topological) minor (note that it has been recently proved that this problem does not admit *uniform* polynomial kernels on general graphs [156]). We chose these families of problems as they are all tuned by a secondary parameter that is either the constant r or the size of the graphs in the family $\mathcal{F}$. That way, we not only capture a wealth of parameterized problems, but we also make explicit the contribution of the secondary parameter in the size of the derived kernels. (We would like to note that the constants involved in the kernels for r-Dominating Set and r-Scattered Set (resp. Planar-$\mathcal{F}$-Deletion) depend on the function f_c (resp. f_m) defined in Proposition 6.2 (resp. Proposition 6.1) in Section 6.2.)

Organization of the chapter. We need some definitions that have been introduced in Chapters 3 and 5, as well as in previous work on this topic [63, 140, J15], including graph minors, parameterized problems, (rooted) tree-decompositions, boundaried graphs, the canonical equivalence relation $\equiv_{\Pi,t}$ for a problem Π and an integer t, FII, protrusions, and protrusion decompositions. For convenience, we restate some of these definitions in Section 6.2. In Section 6.3 we introduce the basic definitions of our framework and present an explicit protrusion replacer. The next three sections are devoted to showing how to apply our methodology to various families of problems, Namely, we focus on r-Dominating Set in Section 6.4, on r-Scattered Set in Section 6.5, and on Planar-$\mathcal{F}$-Deletion in Section 6.6. Finally, we conclude with some directions for further research in Section 6.7.

6.2 Preliminaries

In this section we restate some of the definitions introduced Chapter 5 concerning boundaried graphs, the canonical equivalence relation, and finite integer index. The reason is that, for convenience, there are slight differences between some of the definitions and the notations we use in this chapter with respect to the ones introduced in Chapter 5.

Boundaried graphs and canonical equivalence relation. The following two definitions are taken from [63].

Definition 6.1 (Boundaried graphs) *A* boundaried graph *is a graph G with a set $B \subseteq V(G)$ of distinguished vertices and an injective labeling $\lambda : B \to \mathbb{N}^+$. The set B is called the* boundary *of G and the vertices in B are called* boundary vertices*. Given a boundaried graph G, we denote its boundary by $\partial(G)$, we denote its labeling by λ_G, and we define its label set by $\Lambda(G) = \{\lambda_G(v) \mid v \in \partial(G)\}$. We say that a boundaried graph is a t-*boundaried graph *if $\Lambda(G) \subseteq \{1, \ldots, t\}$.*

Note that a 0-boundaried graph is just a graph with no boundary.

Definition 6.2 (Gluing operation) *Let G_1 and G_2 be two boundaried graphs. We denote by $G_1 \oplus G_2$ the graph obtained by taking the disjoint union of G_1 and G_2 and identifying vertices with the same label of the boundaries of G_1 and G_2. In $G_1 \oplus G_2$ there is an edge between two labeled vertices if there is an edge between them in G_1 or in G_2.*

In the above definition, after identifying vertices with the same label, we may consider the resulting graph as a boundaried graph or not, depending on whether we need the labels for further gluing operations.

We now introduce the canonical equivalence relation on boundaried graphs. Note that we adopt here the original definition given in [70], and reused in [63], rather than the one we used for convenience in Definition 5.5 in Chapter 5. Namely, the main difference is that here we do not require condition 1 of Definition 5.5. Also, for simplicity ask here the graphs to share the same set of labels in order for them to belong to the same equivalence class.

Definition 6.3 (Canonical equivalence on boundaried graphs) *Let Π be a parameterized graph problem and let $t \in \mathbb{N}^+$. Given two t-boundaried graphs G_1 and G_2, we say that $G_1 \equiv_{\Pi,t} G_2$ if $\Lambda(G_1) = \Lambda(G_2)$ and there exists a transposition constant $\Delta_{\Pi,t}(G_1, G_2) \in \mathbb{Z}$ such that for every t-boundaried graph $H \in \mathcal{G}$ and every $k \in \mathbb{Z}$, it holds that $(G_1 \oplus H, k) \in \Pi$ if and only if $(G_2 \oplus H, k - \Delta_{\Pi,t}(G_1, G_2)) \in \Pi$.*

We define in Section 6.3 another equivalence relation on boundaried graphs that refines this canonical one (cf. Definitions 6.9 and 6.10), and that will allow us to perform a constructive protrusion replacement with explicit bounds.

As discussed in Chapter 5, the notion of *Finite Integer Index* was originally defined by Bodlaender and van Antwerpen-de Fluiter [70, 237] (see also [39]). We would like to note that FII does not play any role in the framework that we present for constructing explicit kernels, but we present its definition for completeness, as we will sometimes refer to it throughout the chapter.

Definition 6.4 (Finite Integer Index (FII)) *A parameterized graph problem Π has* Finite Integer Index *(*FII *for short) if for every positive integer t, the equivalence relation $\equiv_{\Pi,t}$ has finite index.*

Note again that the above definition of FII slightly differs from the one given in Definition 5.5 in Chapter 5.

The following definitions concerning protrusions coincide with those given in Chapter 5; we recall them here for completeness.

Protrusions and protrusion decompositions. Given a graph $G = (V, E)$ and a set $W \subseteq V$, we define $\mathbf{bd}(W)$ as the vertices in W that have a neighbor in $V \setminus W$. A set $W \subseteq V(G)$ is a *t-protrusion* if $|\mathbf{bd}(W)| \leqslant t$ and $\mathsf{tw}(G[W]) \leqslant t-1$. We would like to note that a t-protrusion W can be naturally seen as a t-boundaried graph by arbitrarily assigning labels to the vertices in $\mathbf{bd}(W)$. In this case, it clearly holds that $\partial(W) = \mathbf{bd}(W)$. Note also that if G is a t-boundaried graph of treewidth at most $t-1$, we may assume that the boundary vertices are contained in any specified bag of a tree-decomposition, by increasing the width of the given tree-decomposition to at most $2t-1$.

An *(α, t)-protrusion decomposition* of a graph G is a partition $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $V(G)$ such that:

(i) for every $1 \leqslant i \leqslant \ell$, $N(Y_i) \subseteq Y_0$;

(ii) $\max\{\ell, |Y_0|\} \leqslant \alpha$; and

(iii) for every $1 \leqslant i \leqslant \ell$, $Y_i \cup N_{Y_0}(Y_i)$ is a t-protrusion of G.

When G is the input of a parameterized graph problem with parameter k, we say that an (α, t)-protrusion decomposition of G is *linear* whenever $\alpha = \mathcal{O}(k)$.

Large treewidth and grid minors. In our applications in Sections 6.4, 6.5, and 6.6 we will need the following results, which state a *linear* relation between the treewidth and certain grid-like graphs that can be found as minors or contractions in a graph that excludes some fixed (apex) graph as a minor.

Proposition 6.1 (Demaine and Hajiaghayi [104]) *There is a function $f_m : \mathbb{N} \to \mathbb{N}$ such that for every h-vertex graph H and every positive integer r, every H-minor-free graph with treewidth at least $f_m(h) \cdot r$, contains an $(r \times r)$-grid as a minor.*

Before we state the next proposition, we need to define a grid-like graph that is suitable for a contraction counterpart of Proposition 6.1. Let Γ_r ($r \geqslant 2$) be the graph obtained from the $(r \times r)$-grid by triangulating internal faces of the $(r \times r)$-grid such that all internal vertices become of degree 6, all non-corner external vertices are of degree 4, and one corner of degree 2 is joined by edges with all vertices of the external face (the *corners* are the vertices that in the underlying grid have degree 2). The graph Γ_6 is shown in Figure 6.1.

Proposition 6.2 (Fomin *et al.* [134]) *There is a function $f_c : \mathbb{N} \to \mathbb{N}$ such that for every h-vertex apex graph H and every positive integer r, every H-minor-free graph with treewidth at least $f_c(h) \cdot r$, contains the graph Γ_r as a contraction.*

Propositions 6.1 and 6.2 have been the main tools for developing Bidimensionality theory for kernelization [140]. The best known estimation for function f_m has been given by

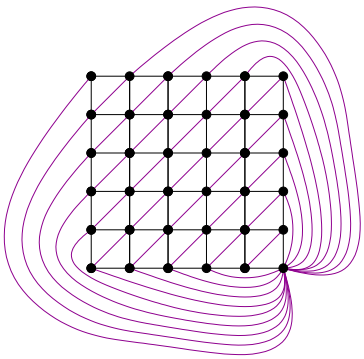

Figure 6.1: The graph Γ_6.

Kawarabayashi and Kobayashi in [178] and is $f_m(r) = 2^{\mathcal{O}(r^2 \cdot \log r)}$. To our knowledge, no reasonable estimation for the function f_c is known up to now. The two functions f_m and f_c will appear in the upper bounds on the size of the kernels presented in Sections 6.4, 6.5, and 6.6. Any improvement on these functions will directly translate to the sizes of our kernels. We would like to stress that, even if we are not aware on any explicit upper bound on the function f_c, this function is a *fixed one*, that is, it does not depend on any particular problem or on the "meta-parameters" associated with the problems.

6.3 An explicit protrusion replacer

In this section we present our strategy to construct an explicit protrusion replacer via dynamic programming. For a positive integer t, we define $\mathcal{B}_t$ as the class of all t-boundaried graphs, and we define $\mathcal{F}_t$ as the class of all t-boundaried graphs of treewidth at most $t-1$ that have a rooted tree-decomposition with all boundary vertices contained in the root-bag[1]. Note that it holds clearly that $\mathcal{F}_t \subseteq \mathcal{B}_t$. We will restrict ourselves to parameterized graph problems such that a solution can be certified by a subset of vertices.

Definition 6.5 (Vertex-certifiable problem) *A parameterized graph problem* Π *is called* vertex-certifiable *if there exists a language* L_Π *(called* certifying language for Π*) defined on pairs* (G, S)*, where* G *is a graph and* $S \subseteq V(G)$*, such that* (G, k) *is a* YES*-instance of* Π *if and only if there exists a subset* $S \subseteq V(G)$ *with* $|S| \leqslant k$ *(or* $|S| \geqslant k$*, depending on the problem) such that* $(G, S) \in L_\Pi$.

Many graph problems are vertex-certifiable, like r-DOMINATING SET, FEEDBACK VERTEX SET, or TREEWIDTH-t VERTEX DELETION. This section is structured as follows. In Section 6.3.1 we define the notion of encoder, the main object that will allow us to formalize in an abstract way the tables of dynamic programming. In Section 6.3.2 we use encoders to define an equivalence relation on graphs in $\mathcal{F}_t$ that, under some natural

[1]Note that the latter condition in the definition of $\mathcal{F}_t$ could be avoided by allowing the width of the tree-decompositions of the graphs in $\mathcal{F}_t$ to be at most $2t-1$, such that all boundary vertices could be added to all bags of any tree-decomposition.

technical conditions, will be a *refinement* of the canonical equivalence relation defined by a problem Π (see Definition 6.3 in Section 6.2). This refined equivalence relation allows us to provide an explicit upper bound on the size of its representatives (Lemma 6.3), as well as a linear-time algorithm to find them (Lemma 6.4). In Section 6.3.3 we use the previous ingredients to present an explicit protrusion replacement rule (Theorem 6.1), which replaces a large enough protrusion with a bounded-size representative from its equivalence class, in such a way that the parameter does not increase.

6.3.1 Encoders

The Dominating Set problem, as a vertex-certifiable problem, will be used hereafter as a running example to particularize our general framework and definitions. Let us start with a description of dynamic programming tables for Dominating Set on graphs of bounded treewidth, which will illustrate the final purpose of the definitions stated below.

Running example: Let B be a bag of a rooted tree-decomposition $(T, \mathcal{X})$ of width $t-1$ of a graph $G \in \mathcal{F}_t$. The dynamic programming (DP) tables for Dominating Set can be defined as follows. The entries of the DP-table for B are indexed by the set of tuples $R \in \{0, \uparrow 1, \downarrow 1\}^{|B|}$, so-called *encodings*. As detailed below, the symbol 0 stands for vertices in the (partial) dominating set, the symbol $\downarrow 1$ for vertices that are already dominated, and $\uparrow 1$ for vertices with no constraints. More precisely, the coordinates of each $|B|$-tuple are in one-to-one correspondence with the vertices of B. For a vertex $v \in B$, we denote by $R(v)$ its corresponding coordinate in the encoding R. A subset $S \subseteq V(G_B)$ is a *partial dominating set satisfying* R if the following conditions are satisfied:

- $\forall v \in V(G_B) \setminus B$, $d_{G_B}(v, S) \leqslant 1$; and
- $\forall v \in B$: $R(v) = 0 \Rightarrow v \in S$, and $R(v) = \downarrow 1 \Rightarrow d_{G_B}(v, S) \leqslant 1$.

Observe that if S is a partial dominating set satisfying R, then $\{v \in B \mid R(v) = 0\} \subseteq S$, but S may also contain vertices with $R(v) \neq 0$. Likewise, the vertices that are not (yet) dominated by S are contained in the set $\{v \in B \mid R(v) = \uparrow 1\}$. $\diamond$

The following definition considers the tables of dynamic programming in an abstract way.

Definition 6.6 (Encoder) *An* encoder $\mathcal{E}$ *is a pair* $(\mathcal{C}, L_{\mathcal{C}})$ *where*

(i) $\mathcal{C}$ *is a function that, for each (possibly empty) finite subset* $I \subseteq \mathbb{N}^+$*, outputs a (possibly empty) finite set* $\mathcal{C}(I)$ *of strings over some alphabet. Each* $R \in \mathcal{C}(I)$ *is called a* $\mathcal{C}$-encoding *of* I*; and*

(ii) $L_{\mathcal{C}}$ *is a computable language whose strings encode triples* (G, S, R)*, where* G *is a boundaried graph,* $S \subseteq V(G)$*, and* $R \in \mathcal{C}(\Lambda(G))$*. If* $(G, S, R) \in L_{\mathcal{C}}$*, we say that* S satisfies *the* $\mathcal{C}$*-encoding* R*.*

As it will become clear with the running example, the set I represents the labels from a bag, $\mathcal{C}(I)$ represents the possible configurations of the vertices in the bag, and $L_{\mathcal{C}}$ contains triples that correspond to solutions to these configurations.

Running example: Each rooted graph G_B can be naturally viewed as a $|B|$-boundaried graph such that $B = \partial(G_B)$ with $I = \Lambda(G_B)$. Let $\mathcal{E}_{\mathrm{DS}} = (\mathcal{C}_{\mathrm{DS}}, L_{\mathcal{C}_{\mathrm{DS}}})$ be the encoder described above for Dominating Set. The tables of the dynamic programming algorithm to solve Dominating Set are obtained by assigning to every $\mathcal{C}_{\mathrm{DS}}$-encoding (that is, DP-table entry) $R \in \mathcal{C}_{\mathrm{DS}}(I)$, the size of a minimum partial dominating set satisfying R, or $+\infty$ if such a set of vertices does not exist. This defines a function $f_G^{\mathcal{E}_{\mathrm{DS}}} : \mathcal{C}_{\mathrm{DS}}(I) \to \mathbb{N} \cup \{+\infty\}$. Observe that if $B = \partial(G_B) = \emptyset$, then the value assigned to the encodings in $\mathcal{C}_{\mathrm{DS}}(\emptyset)$ is indeed the size of a minimum dominating set of G_B. ◇

In the remainder of this subsection we will state several definitions for minimization problems, and we will restate them for maximization problems whenever some change is needed. For a general minimization problem Π, we will only be interested in encoders that permit to solve Π via dynamic programming. More formally, we define a Π-encoder and the values assigned to the encodings as follows.

Definition 6.7 (Π-encoder and its associated function) *Let Π be a vertex-certifiable minimization problem.*

(i) An encoder $\mathcal{E} = (\mathcal{C}, L_{\mathcal{C}})$ is a Π-encoder *if $\mathcal{C}(\emptyset)$ consists of a single $\mathcal{C}$-encoding, namely $R_\emptyset$, such that for every 0-boundaried graph G and every $S \subseteq V(G)$, $(G, S, R_\emptyset) \in L_{\mathcal{C}}$ if and only if $(G, S) \in L_\Pi$.*

(ii) Let G be a t-boundaried graph with $\Lambda(G) = I$. We define the function $f_G^{\mathcal{E}} : \mathcal{C}(I) \to \mathbb{N} \cup \{+\infty\}$ as

$$f_G^{\mathcal{E}}(R) \;=\; \min\{k \;:\; \exists S \subseteq V(G), |S| \leqslant k, (G, S, R) \in L_{\mathcal{C}}\}. \tag{6.1}$$

In Equation (6.1), if such a set S does not exist, we set $f_G^{\mathcal{E}}(R) := +\infty$. We define $\mathcal{C}_G^*(I) := \{R \in \mathcal{C}(I) \mid f_G^{\mathcal{E}}(R) \neq +\infty\}$.

Condition (i) in Definition 6.7 guarantees that, when the considered graph G has no boundary, the language of the encoder is able to *certify* a solution of problem Π. In other words, we ask that the set $\{(G, S) \mid (G, S, R_\emptyset) \in L_{\mathcal{C}})\}$ is a *certifying language* for Π. Observe that for a 0-boundaried graph G, the function $f_G^{\mathcal{E}}(R_\emptyset)$ outputs the minimum size of a set S such that $(G, S) \in L_\Pi$.

For encoders $\mathcal{E}' = (\mathcal{C}', L_{\mathcal{C}'})$ that will be associated with problems where the objective is to find a set of vertices of size *at least* some value, the corresponding function $f_G^{\mathcal{E}'} : \mathcal{C}'(I) \to \mathbb{N} \cup \{-\infty\}$ is defined as

$$f_G^{\mathcal{E}'}(R) \;=\; \max\{k \;:\; \exists S \subseteq V(G), |S| \geqslant k, (G, S, R) \in L_{\mathcal{C}'}\}. \tag{6.2}$$

Similarly, in Equation (6.2), if such a set S does not exist, we set $f_G^{\mathcal{E}}(R) := -\infty$. We define $\mathcal{C}'^*_G(I) := \{R \in \mathcal{C}'(I) \mid f_G^{\mathcal{E}'}(R) \neq -\infty\}$.

The following definition provides a way to control the number of possible distinct values assigned to encodings. This property will play a similar role to FII or *monotonicity* in previous work [63, 140, J15].

Definition 6.8 (Confined encoding) *An encoder $\mathcal{E}$ is* g-confined *if there exists a function $g : \mathbb{N} \to \mathbb{N}$ such that for any t-boundaried graph G with $\Lambda(G) = I$ it holds that either $\mathcal{C}^*_G(I) = \emptyset$ or*

$$\max_{R \in \mathcal{C}^*_G(I)} f^{\mathcal{E}}_G(R) \; - \; \min_{R \in \mathcal{C}^*_G(I)} f^{\mathcal{E}}_G(R) \; \leqslant \; g(t). \tag{6.3}$$

See Figure 6.2 for a schematic illustration of a confined encoder. In this figure, each column of the table corresponds to a $\mathcal{C}$-encoding R, which is filled with the value $f^{\mathcal{E}}_G(R)$.

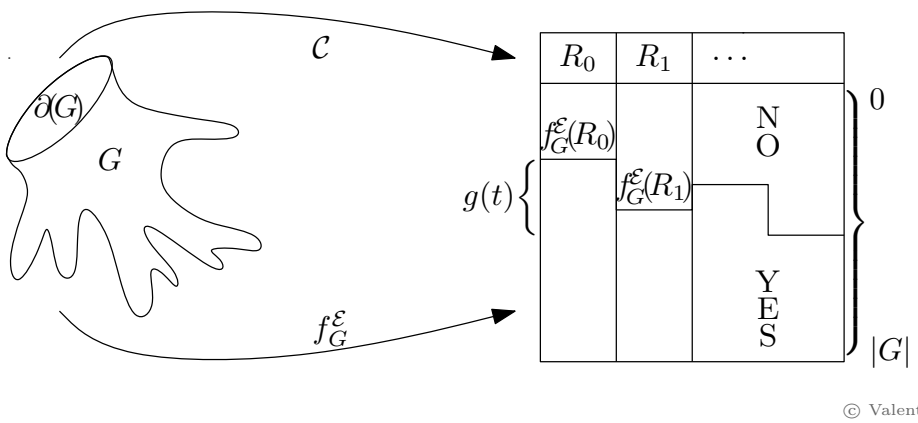

Figure 6.2: Schematic illustration of a g-confined encoding, in which we assume for simplicity that $\mathcal{C}^*_G(\Lambda(G)) = \mathcal{C}(\Lambda(G))$.

Running example: It is easy to observe that the encoder $\mathcal{E}_{\text{DS}}$ described above is g-confined for $g(t) = t$. Indeed, let G be a t-boundaried graph (corresponding to the graph G_B considered before) with $\Lambda(G) = I$. Consider an arbitrary encoding $R \in \mathcal{C}(I)$ and the encoding $R_0 \in \mathcal{C}(I)$ satisfying $R_0(v) = 0$ for every $v \in \partial(G)$. Let $S_0 \subseteq V(G)$ be a minimum-sized partial dominating set satisfying R_0, i.e., such that $(G, S_0, R_0) \in L_{\mathcal{C}_{\text{DS}}}$. Observe that S_0 also satisfies R, i.e., $(G, S_0, R) \in L_{\mathcal{C}_{\text{DS}}}$. It then follows that $f^{\mathcal{E}_{\text{DS}}}_G(R_0) = \max_R f^{\mathcal{E}_{\text{DS}}}_G(R)$. Moreover, let $S \subseteq V(G)$ be a minimum-sized partial dominating set satisfying R, i.e., such that $(G, S, R) \in L_{\mathcal{C}_{DS}}$. Then note that R_0 is satisfied by the set $S \cup \partial(G)$, so we have that for every encoding R, $f^{\mathcal{E}_{\text{DS}}}_G(R) + |\partial(G)| \geqslant f^{\mathcal{E}_{\text{DS}}}_G(R_0)$. It follows that $f^{\mathcal{E}_{\text{DS}}}_G(R_0) - \min_R f^{\mathcal{E}_{\text{DS}}}_G(R) \leqslant |\partial(G)| \leqslant t$, proving that the encoder is indeed g-confined. $\diamond$

For some problems and encoders, we may need to "force" the confinement of an encoder $\mathcal{E}$ that may not be confined according to Definition 6.8, while still preserving its usefulness for dynamic programming, in the sense that no relevant information is removed from the tables (for example, see the encoder for r-Scattered Set in Section 6.5.1). To this end, given a function $g : \mathbb{N} \to \mathbb{N}$, we define the function $f^{\mathcal{E},g}_G : \mathcal{C}(I) \to \mathbb{N} \cup \{+\infty\}$ as

$$f_G^{\mathcal{E},g}(R) \;=\; \begin{cases} +\infty, & \text{if } f_G^{\mathcal{E}}(R) - g(t) > \min_{R' \in \mathcal{C}(I)} f_G^{\mathcal{E}}(R') \\ f_G^{\mathcal{E}}(R), & \text{otherwise.} \end{cases} \tag{6.4}$$

Intuitively, one shall think as the function $f_G^{\mathcal{E},g}$ as a "compressed" version of the function $f_G^{\mathcal{E}}$, which stores only the values that are useful for performing dynamic programming.

For encoders $\mathcal{E}' = (\mathcal{C}', L_{\mathcal{C}'})$ associated with maximization problems, given a function $g : \mathbb{N} \to \mathbb{N}$, we define the function $f_G^{\mathcal{E}',g} : \mathcal{C}(I) \to \mathbb{N} \cup \{-\infty\}$ as

$$f_G^{\mathcal{E}',g}(R) \;=\; \begin{cases} -\infty, & \text{if } f_G^{\mathcal{E}'}(R) + g(t) < \max_{R' \in \mathcal{C}(I)} f_G^{\mathcal{E}'}(R') \\ f_G^{\mathcal{E}'}(R), & \text{otherwise.} \end{cases} \tag{6.5}$$

6.3.2 Equivalence relations and representatives

An encoder $\mathcal{E}$ together with a function $g : \mathbb{N} \to \mathbb{N}$ define an equivalence relation $\sim^*_{\mathcal{E},g,t}$ on t-boundaried graphs as follows. (In fact, in our applications we will use only this equivalence relation on graphs in $\mathcal{F}_t$, but for technical reasons we need to define it on general t-boundaried graphs.)

Definition 6.9 (Equivalence relations $\sim^*_{\mathcal{E},g,t}$ and $\sim_{\mathcal{E},g,t}$) *Let $\mathcal{E}$ be an encoder, let $g : \mathbb{N} \to \mathbb{N}$, and let $G_1, G_2 \in \mathcal{B}_t$. We say that $G_1 \sim^*_{\mathcal{E},g,t} G_2$ if and only if $\Lambda(G_1) = \Lambda(G_2) =: I$ and there exists an integer c, depending only on G_1 and G_2, such that for every $\mathcal{C}$-encoding $R \in \mathcal{C}(I)$ it holds that*

$$f_{G_1}^{\mathcal{E},g}(R) \;=\; f_{G_2}^{\mathcal{E},g}(R) + c. \tag{6.6}$$

If we restrict the graphs G_1, G_2 to belong to $\mathcal{F}_t$, then the corresponding equivalence relation, which is a restriction of $\sim^*_{\mathcal{E},g,t}$, is denoted by $\sim_{\mathcal{E},g,t}$.

Note that if there exists $R \in \mathcal{C}(I)$ such that $f_{G_1}^{\mathcal{E},g}(R) \notin \{-\infty, +\infty\}$, then the integer c satisfying Equation (6.6) is unique, otherwise every integer c satisfies Equation (6.6). We define the following function $\Delta_{\mathcal{E},g,t} : \mathcal{B}_t \times \mathcal{B}_t \to \mathbb{Z}$, which is called, following the terminology from Bodlaender et al. [63], the *transposition function* for the equivalence relation $\sim^*_{\mathcal{E},g,t}$.

$$\Delta_{\mathcal{E},g,t}(G_1, G_2) \;=\; \begin{cases} c, & \text{if } G_1 \sim^*_{\mathcal{E},g,t} G_2 \text{ and Eq. (6.6) holds for a unique integer } c; \\ 0, & \text{if } G_1 \sim^*_{\mathcal{E},g,t} G_2 \text{ and Eq. (6.6) holds for every integer; and} \\ & \text{undefined otherwise} \end{cases} \tag{6.7}$$

Note that we can consider the restriction of the function $\Delta_{\mathcal{E},g,t}$ to couples of graphs in $\mathcal{F}_t$, defined by using the restricted equivalence relation $\sim_{\mathcal{E},g,t}$.

If we are dealing with a problem defined on a graph class $\mathcal{G}$, the protrusion replacement rule has to preserve the class $\mathcal{G}$, as otherwise we would obtain a *bikernel* instead of a kernel. That is, we need to make sure that, when replacing a graph in $\mathcal{B}_t \cap \mathcal{G}$ or in $\mathcal{F}_t \cap \mathcal{G}$ with one of its representatives, we do not produce a graph that does not belong to $\mathcal{G}$ anymore. To this end, we define an equivalence relation $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ (resp. $\sim_{\mathcal{E},g,t,\mathcal{G}}$) on graphs in $\mathcal{B}_t \cap \mathcal{G}$ (resp. $\mathcal{F}_t \cap \mathcal{G}$), which refines the equivalence relation $\sim^*_{\mathcal{E},g,t}$ (resp. $\sim_{\mathcal{E},g,t}$) of Definition 6.9.

Definition 6.10 (Equivalence relations $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ and $\sim_{\mathcal{E},g,t,\mathcal{G}}$) *Let $\mathcal{G}$ be a class of graphs and let $G_1, G_2 \in \mathcal{B}_t \cap \mathcal{G}$.*

(i) $G_1 \sim_{\mathcal{G},t} G_2$ if and only if for any graph $H \in \mathcal{B}_t$, $G_1 \oplus H \in \mathcal{G}$ if and only if $G_2 \oplus H \in \mathcal{G}$.

(ii) $G_1 \sim^*_{\mathcal{E},g,t,\mathcal{G}} G_2$ if and only if $G_1 \sim^*_{\mathcal{E},g,t} G_2$ and $G_1 \sim_{\mathcal{G},t} G_2$.

If we restrict the graphs G_1, G_2 to belong to $\mathcal{F}_t$ (but still $H \in \mathcal{B}_t$), then the corresponding equivalence relation, which is a restriction of $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$, is denoted by $\sim_{\mathcal{E},g,t,\mathcal{G}}$.

It is well-known by Büchi's theorem that regular languages are precisely those definable in Monadic Second Order logic (MSO logic). By Myhill-Nerode's theorem, it follows that if the membership in a graph class $\mathcal{G}$ can be expressed in MSO logic, then the equivalence relation $\sim_{\mathcal{G},t}$ has a finite number of equivalence classes (see for instance [115, 131]). However, we do not have in general an explicit upper bound on the number of equivalence classes of $\sim_{\mathcal{G},t}$, henceforth denoted by $r_{\mathcal{G},t}$. Fortunately, in the context of our applications in Sections 6.4, 6.5, and 6.6, where $\mathcal{G}$ will be a class of graphs that exclude some fixed graph on h vertices as a (topological) minor, this will always be possible, and in this case it holds that $r_{\mathcal{G},t} \leqslant 2^{t \log t} \cdot h^t \cdot 2^{h^2}$ (see Section 6.6.1). Note that (topological-)minor-free graphs constitute a particular case of the classes of graphs whose membership can be expressed in MSO logic. We would like to stress here that we rely on the expressibility of the *graph class* $\mathcal{G}$ in MSO logic, whereas in previous work [63, 140, J15] what is used in the expressibility in CMSO logic of the *problems* defined on a graph class.

For an encoder $\mathcal{E} = (\mathcal{C}, L_{\mathcal{C}})$, we let $s_{\mathcal{E}}(t) := \max_{I \subseteq \{1,\ldots,t\}} |\mathcal{C}(I)|$, where $|\mathcal{C}(I)|$ denotes the number of $\mathcal{C}$-encodings in $\mathcal{C}(I)$. The following lemma gives an upper bound on the number of equivalence classes of $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$, which depends also on $r_{\mathcal{G},t}$.

Lemma 6.1 *Let $\mathcal{G}$ be a graph class whose membership can be expressed in MSO logic. For any encoder $\mathcal{E}$, any function $g : \mathbb{N} \to \mathbb{N}$, and any positive integer t, the equivalence relation $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ has finite index. More precisely, the number of equivalence classes of $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is at most $r(\mathcal{E}, g, t, \mathcal{G}) := (g(t)+2)^{s_{\mathcal{E}}(t)} \cdot 2^t \cdot r_{\mathcal{G},t}$. In particular, the number of equivalence classes of $\sim_{\mathcal{E},g,t,\mathcal{G}}$ is at most $r(\mathcal{E}, g, t, \mathcal{G})$ as well.*

Proof: Let us first show that the equivalence relation $\sim^*_{\mathcal{E},g,t}$ has finite index. Indeed, let $I \subseteq \{1, \ldots, t\}$. By definition, we have that for any graph $G \in \mathcal{B}_t$ with $\Lambda(G) = I$, the function $f_G^{\mathcal{E},g}$ can take at most $g(t)+2$ distinct values ($g(t)+1$ finite values and possibly the value $+\infty$). Therefore, it follows that the number of equivalence classes of $\sim^*_{\mathcal{E},g,t}$ containing all graphs G in $\mathcal{B}_t$ with $\Lambda(G) = I$ is at most $(g(t)+2)^{|\mathcal{C}(I)|}$. As the number of subsets of $\{1, \ldots, t\}$ is 2^t, we deduce that the overall number of equivalence classes of $\sim^*_{\mathcal{E},g,t}$ is at most $(g(t)+2)^{s_{\mathcal{E}}(t)} \cdot 2^t$. Finally, since the equivalence relation $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is the Cartesian product of the equivalence relations $\sim^*_{\mathcal{E},g,t}$ and $\sim_{\mathcal{G},t}$, the result follows from the fact that $\mathcal{G}$ can be expressed in MSO logic. □

In order for an encoding $\mathcal{E}$ and a function g to be useful for performing dynamic programming on graphs in $\mathcal{F}_t$ that belong to a graph class $\mathcal{G}$ (recall that this is our final objective), we introduce the following definition, which captures the natural fact that the

tables of a dynamic programming algorithm should depend exclusively on the tables of the descendants in a rooted tree-decomposition. Before moving to the definition, we note that given a graph $G \in \mathcal{F}_t$ and a rooted tree-decomposition $(T, \mathcal{X})$ of G of width at most $t-1$ such that $\partial(G)$ is contained in the root-bag of $(T, \mathcal{X})$, the labels of $\partial(G)$ can be propagated in a natural way to all bags of $(T, \mathcal{X})$ by introducing, removing, and shifting labels appropriately. Therefore, for any node x of T, the graph G_x can be naturally seen as a graph in $\mathcal{F}_t$. (A brief discussion can be found in the proof of Lemma 6.4, and we refer to [63] for more details.)

Again, for technical reasons (namely, for the proof of Lemma 6.2), we need to state the definition below for graphs in $\mathcal{B}_t$, even if we will only use it for graphs in $\mathcal{F}_t$.

Definition 6.11 (DP-friendly equivalence relation) *An equivalence relation $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is* DP-friendly *if for any graph $G \in \mathcal{B}_t$ with $\partial(G) = A$ and any separator $B \subseteq V(G)$ with $|B| \leqslant t$, the following holds: let $G_B = G[\mathcal{H} \cup B]$ for $\mathcal{H}$ being any collection of connected components of $G - B$ such that $A \cap V(G_B) \subseteq B$. Considering G_B as a t-boundaried graph with boundary B, let G' be the t-boundaried graph with $\partial(G') = A$ obtained from G by replacing the subgraph G_B with a t-boundaried graph G'_B such that $G_B \sim^*_{\mathcal{E},g,t,\mathcal{G}} G'_B$. Then G' satisfies the following conditions:*

(i) $G \sim^*_{\mathcal{E},g,t,\mathcal{G}} G'$; *and*

(ii) $\Delta_{\mathcal{E},g,t}(G, G') = \Delta_{\mathcal{E},g,t}(G_B, G'_B)$.

Note that if an equivalence relation $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly, then by definition its restriction $\sim_{\mathcal{E},g,t,\mathcal{G}}$ to graphs in $\mathcal{F}_t$ is DP-friendly as well.

We would like to note that in the above definition we have used the notation G_B because in all applications the subgraph to be replaced will correspond to a rooted subtree in a tree-decomposition of a graph G. With this in mind, the condition $A \cap V(G_B) \subseteq B$ in Definition 6.11 corresponds to the fact that the boundary A will correspond in the applications to the vertices in the root-bag of a rooted tree-decomposition of G.

In Definition 6.11, as well as in the remainder of the chapter, when we *replace* the graph G_B with the graph G'_B, we do *not* remove from G any of the edges with both endvertices on the boundary of G_B. That is, $G' = (G - (V(G_x) - \partial(V(G_B)))) \oplus G'_B$.

Recall that for the protrusion replacement to be valid for a problem Π, the equivalence relation $\sim_{\mathcal{E},g,t,\mathcal{G}}$ needs to be a refinement of the canonical equivalence relation $\equiv_{\Pi,t}$ (note that this implies, in particular, that if $\sim_{\mathcal{E},g,t,\mathcal{G}}$ has finite index, then Π has FII). The next lemma states a sufficient condition for this property, and furthermore it gives the value of the transposition constant $\Delta_{\Pi,t}(G_1, G_2)$, which will be needed in order to update the parameter after the replacement.

Lemma 6.2 *Let Π be a vertex-certifiable problem. If $\mathcal{E}$ is a Π-encoder and $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is a DP-friendly equivalence relation, then for any two graphs $G_1, G_2 \in \mathcal{B}_t$ such that $G_1 \sim^*_{\mathcal{E},g,t,\mathcal{G}} G_2$, it holds that $G_1 \equiv_{\Pi,t} G_2$ and $\Delta_{\Pi,t}(G_1, G_2) = \Delta_{\mathcal{E},g,t}(G_1, G_2)$. In particular, if $\mathcal{E}$ is a Π-encoder and $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly, then for any two graphs $G_1, G_2 \in \mathcal{F}_t$ such that $G_1 \sim_{\mathcal{E},g,t,\mathcal{G}} G_2$, it holds that $G_1 \equiv_{\Pi,t} G_2$ and $\Delta_{\Pi,t}(G_1, G_2) = \Delta_{\mathcal{E},g,t}(G_1, G_2)$.*

Proof: Assume without loss of generality that Π is a minimization problem, and let $\mathcal{E} = (\mathcal{C}, L_{\mathcal{C}})$. We need to prove that for any t-boundaried graph H and any integer k, $(G_1 \oplus H, k) \in \Pi$ if and only if $(G_2 \oplus H, k + \Delta_{\mathcal{E},g,t}(G_1, G_2)) \in \Pi$. Suppose that $(G_1 \oplus H, k) \in \Pi$ (by symmetry the same arguments apply starting with G_2). This means that there exists $S_1 \subseteq V(G_1 \oplus H)$ with $|S_1| \leqslant k$ such that $(G_1 \oplus H, S_1) \in L_{\Pi}$. And since $G_1 \oplus H$ is a 0-boundaried graph and $\mathcal{E}$ is a Π-encoder, we have that $(G_1 \oplus H, S_1, R_{\emptyset}) \in L_{\mathcal{C}}$, where $\mathcal{C}(\emptyset) = \{R_{\emptyset}\}$. This implies that

$$f^{\mathcal{E}}_{G_1 \oplus H}(R_{\emptyset}) \ \leqslant \ |S_1| \ \leqslant \ k. \tag{6.8}$$

As $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly and $G_1 \sim^*_{\mathcal{E},g,t,\mathcal{G}} G_2$, it follows that $(G_1 \oplus H) \sim^*_{\mathcal{E},g,t,\mathcal{G}} (G_2 \oplus H)$ and that $\Delta_{\mathcal{E},g,t}(G_1 \oplus H, G_2 \oplus H) = \Delta_{\mathcal{E},g,t}(G_1, G_2)$. Since $G_2 \oplus H$ is also a 0-boundaried graph, the latter property and Equation (6.8) imply that

$$f^{\mathcal{E}}_{G_2 \oplus H}(R_{\emptyset}) \ = \ f^{\mathcal{E}}_{G_1 \oplus H}(R_{\emptyset}) + \Delta_{\mathcal{E},g,t}(G_1, G_2) \ \leqslant \ k + \Delta_{\mathcal{E},g,t}(G_1, G_2). \tag{6.9}$$

From Equation (6.9) it follows that there exists $S_2 \subseteq V(G_2 \oplus H)$ with $|S_2| \leqslant k + \Delta_{\mathcal{E},g,t}(G_1, G_2)$ such that $(G_2 \oplus H, S_2, R_{\emptyset}) \in L_{\mathcal{C}}$. Since $G_2 \oplus H$ is a 0-boundaried graph and $\mathcal{E}$ is a Π-encoder, this implies that $(G_2 \oplus H, S_2) \in L_{\Pi}$, which in turn implies that $(G_2 \oplus H, k + \Delta_{\mathcal{E},g,t}(G_1, G_2)) \in \Pi$, as we wanted to prove. □

Note that, in particular, Lemma 6.2 implies that under the same hypothesis, for graphs in $\mathcal{F}_t$ the equivalence relation $\sim_{\mathcal{E},g,t,\mathcal{G}}$ refines the canonical equivalence relation $\equiv_{\Pi,t}$.

In the following, we will only deal with equivalence relations $\sim_{\mathcal{E},g,t,\mathcal{G}}$ defined on graphs in $\mathcal{F}_t$, and therefore we will only use this particular case of Lemma 6.2. The reason why we restrict ourselves to graphs in $\mathcal{F}_t$ is that, while a DP-friendly equivalence relation refines the canonical one for all graphs in $\mathcal{B}_t$ (Lemma 6.2), we need *bounded treewidth* in order to bound the *size* of the progressive representatives (Lemma 6.3) and to explicitly *compute* these representatives for performing the replacement (Lemma 6.4).

The following definition will be important to guarantee that, when applying our protrusion replacement rule, the parameter of the problem under consideration does not increase.

Definition 6.12 (Progressive representatives of $\sim_{\mathcal{E},g,t,\mathcal{G}}$) *Let $\mathfrak{C}$ be some equivalence class of $\sim_{\mathcal{E},g,t,\mathcal{G}}$ and let $G \in \mathfrak{C}$. We say that G is a* progressive representative *of $\mathfrak{C}$ if for any graph $G' \in \mathfrak{C}$ it holds that $\Delta_{\mathcal{E},g,t}(G, G') \leqslant 0$.*

In the next lemma we provide an upper bound on the size of a smallest *progressive* representative of any equivalence class of $\sim_{\mathcal{E},g,t,\mathcal{G}}$.

Lemma 6.3 *Let $\mathcal{G}$ be a graph class whose membership can be expressed in MSO logic. For any encoder $\mathcal{E}$, any function $g : \mathbb{N} \to \mathbb{N}$, and any $t \in \mathbb{N}$ such that $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly, every equivalence class of $\sim_{\mathcal{E},g,t,\mathcal{G}}$ has a progressive representative of size at most $b(\mathcal{E}, g, t, \mathcal{G}) := 2^{r(\mathcal{E},g,t,\mathcal{G})+1} \cdot t$, where $r(\mathcal{E}, g, t, \mathcal{G})$ is the function defined in Lemma 6.1.*

Proof: Let $\mathfrak{C}$ be an arbitrary equivalence class of $\sim_{\mathcal{E},g,t,\mathcal{G}}$, and we want to prove that there exists in $\mathfrak{C}$ a progressive representative of the desired size. Let us first argue that

$\mathfrak{C}$ contains some progressive representative. We construct an (infinite) directed graph $D_{\mathfrak{C}}$ as follows. There is a vertex in $D_{\mathfrak{C}}$ for every graph in $\mathfrak{C}$, and for any two vertices $v_1, v_2 \in V(D_{\mathfrak{C}})$, corresponding to two graphs $G_1, G_2 \in \mathfrak{C}$ respectively, there is an arc from v_1 to v_2 if and only if $\Delta_{\mathcal{E},g,t}(G_1, G_2) > 0$. We want to prove that $D_{\mathfrak{C}}$ has a sink, that is, a vertex with no outgoing arc, which by construction is equivalent to the existence of a progressive representative in $\mathfrak{C}$. Indeed, let v be an arbitrary vertex of $D_{\mathfrak{C}}$, and grow greedily a directed path P starting from v. Because of the transitivity of the equivalence relation $\sim_{\mathcal{E},g,t,\mathcal{G}}$ and by construction of $D_{\mathfrak{C}}$, it follows that $D_{\mathfrak{C}}$ does not contain any finite cycle, so P cannot visit vertex v again. On the other hand, since the function $f_G^{\mathcal{E}}$ takes only positive values (except possibly for the value $-\infty$), it follows that there are no arbitrarily long directed paths in $D_{\mathfrak{C}}$ starting from any fixed vertex, so in particular the path P must be finite, and therefore the last vertex in P is necessarily a sink. (Note that for any two graphs $G_1, G_2 \in \mathfrak{C}$ such that their corresponding vertices $v_1, v_2 \in V(D_{\mathfrak{C}})$ are sinks, it holds by construction of $D_{\mathfrak{C}}$ that $\Delta_{\mathcal{E},g,t}(G_1, G_2) = 0$.)

Now let $G \in \mathcal{F}_t \cap \mathcal{G}$ be a progressive representative of $\mathfrak{C}$ with minimum number of vertices. We claim that G has size at most $2^{r(\mathcal{E},g,t,\mathcal{G})+1} \cdot t$. (We would like to stress that at this stage we only need to care about the *existence* of such representative G, and not about how to *compute* it.) Indeed, let $(T, \mathcal{X})$ be a nice rooted tree-decomposition of G of width at most $t-1$ such that $\partial(G)$ is contained in the root-bag (such a nice tree-decomposition exists by [184]), and let r be the root of T.

We first claim that for any node x of T, the graph G_x is a progressive representative of its equivalence class with respect to $\sim_{\mathcal{E},g,t,\mathcal{G}}$, namely $\mathfrak{A}$. Indeed, assume that it is not the case, and let H be a progressive representative of $\mathfrak{A}$, which exists by the discussion in the first paragraph of the proof. Since H is progressive and G_x is not, $\Delta_{\mathcal{E},g,t}(H, G_x) < 0$. Let G_H be the graph obtained from G by replacing G_x with H. Since $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly, it follows that $G \sim_{\mathcal{E},g,t,\mathcal{G}} G_H$ and that $\Delta_{\mathcal{E},g,t}(G_H, G) = \Delta_{\mathcal{E},g,t}(H, G_x) < 0$, contradicting the fact that G is a progressive representative of the equivalence class $\mathfrak{C}$.

We now claim that for any two nodes $x, y \in V(T)$ lying on a path from r to a leaf of T, it holds that $G_x \nsim_{\mathcal{E},g,t,\mathcal{G}} G_y$. Indeed, assume for contradiction that there are two nodes $x, y \in V(T)$ lying on a path from r to a leaf of T such that $G_x \sim_{\mathcal{E},g,t,\mathcal{G}} G_y$. Let $\mathfrak{A}$ be the equivalence class of G_x and G_y with respect to $\sim_{\mathcal{E},g,t,\mathcal{G}}$. By the previous claim, it follows that both G_x and G_y are progressive representatives of $\mathfrak{A}$, and therefore it holds that $\Delta_{\mathcal{E},g,t}(G_y, G_x) = 0$. Suppose without loss of generality that $G_y \subsetneq G_x$ (that is, G_y is a strict subgraph of G_x), and let G' be the graph obtained from G by replacing G_x with G_y. Again, since $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly, it follows that $G \sim_{\mathcal{E},g,t,\mathcal{G}} G'$ and that $\Delta_{\mathcal{E},g,t}(G', G) = \Delta_{\mathcal{E},g,t}(G_y, G_x) = 0$. Therefore, G' is a progressive representative of $\mathfrak{C}$ with $|V(G')| < |V(G)|$, contradicting the minimality of $|V(G)|$.

Finally, since for any two nodes $x, y \in V(T)$ lying on a path from r to a leaf of T we have that $G_x \nsim_{\mathcal{E},g,t,\mathcal{G}} G_y$, it follows that the height of T is at most the number of equivalence classes of $\sim_{\mathcal{E},g,t,\mathcal{G}}$, which is at most $r(\mathcal{E}, g, t, \mathcal{G})$ by Lemma 6.1. Since T is a binary tree, we have that $|V(T)| \leqslant 2^{r(\mathcal{E},g,t,\mathcal{G})+1} - 1$. Finally, since $|V(G)| \leqslant |V(T)| \cdot t$, it follows that $|V(G)| \leqslant 2^{r(\mathcal{E},g,t,\mathcal{G})+1} \cdot t$, as we wanted to prove. □

The next lemma states that if one is given an upper bound on the size of the progressive representatives of an equivalence relation defined on t-protrusions (that is, on graphs in

$\mathcal{F}_t$)[2], then a *small* progressive representative of a t-protrusion can be explicitly calculated in linear time. In other words, it provides a generic and constructive way to perform a dynamic programming procedure to replace protrusions, without needing to deal with the particularities of each encoder in order to compute the tables. Its proof uses some ideas taken from [63, 140].

Lemma 6.4 *Let $\mathcal{G}$ be a graph class, let $\mathcal{E}$ be an encoder, let $g : \mathbb{N} \to \mathbb{N}$, and let $t \in \mathbb{N}$ such that $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly. Assume that we are given an upper bound b on the size of a smallest progressive representative of any equivalence class of $\sim_{\mathcal{E},g,t,\mathcal{G}}$, with $b \geqslant t$. Then, given an n-vertex t-protrusion G inside some graph, we can output in time $\mathcal{O}(n)$ a t-protrusion H inside the same graph of size at most b such that $G \sim_{\mathcal{E},g,t,\mathcal{G}} H$ and the corresponding transposition constant $\Delta_{\mathcal{E},g,t}(H, G)$ with $\Delta_{\mathcal{E},g,t}(H, G) \leqslant 0$, where the hidden constant in the O-notation depends only on $\mathcal{E}, g, b, \mathcal{G}$, and t.*

Proof: Let $\mathcal{E} = (\mathcal{C}, L_{\mathcal{C}})$ be the given encoder. We start by generating a repository $\mathfrak{R}$ containing all the graphs in $\mathcal{F}_t$ with at most $b + 1$ vertices. Such a set of graphs, as well as a rooted nice tree-decomposition of width at most $t - 1$ of each of them, can be clearly generated in time depending only on b and t. By assumption, the size of a smallest progressive representative of any equivalence class of $\sim_{\mathcal{E},g,t,\mathcal{G}}$ is at most b, so $\mathfrak{R}$ contains a progressive representative of any equivalence class of $\sim_{\mathcal{E},g,t,\mathcal{G}}$ with at most b vertices. We now partition the graphs in $\mathfrak{R}$ into equivalence classes of $\sim_{\mathcal{E},g,t,\mathcal{G}}$ as follows. For each graph $H \in \mathfrak{R}$ and each $\mathcal{C}$-encoding $R \in \mathcal{C}(\Lambda(G))$, as $L_{\mathcal{C}}$ is a computable language, we can compute the value $f_G^{\mathcal{E},g}(R)$ in time depending only on $\mathcal{E}, g, t$, and b. Therefore, for any two graphs $H_1, H_2 \in \mathfrak{R}$, we can decide in time depending only on $\mathcal{E}, g, t, b$, and $\mathcal{G}$ whether $H_1 \sim_{\mathcal{E},g,t,\mathcal{G}} H_2$, and if this is the case, we can compute the transposition constant $\Delta_{\mathcal{E},g,t}(H_1, H_2)$ within the same running time.

Given a t-protrusion G on n vertices with boundary $\partial(G)$, we first compute a rooted nice tree-decomposition $(T, \mathcal{X})$ of G such that $\partial(G)$ is contained in the root bag in time $f(t) \cdot n$, by using the linear-time algorithm of Bodlaender [58, 184]. Such a t-protrusion G equipped with a tree-decomposition can be naturally seen as a graph in $\mathcal{F}_t$ by assigning distinct labels from $\{1, \ldots, t\}$ to the vertices in the root-bag. These labels from $\{1, \ldots, t\}$ can be transferred to the vertices in all the bags of $(T, \mathcal{X})$ by performing a standard shifting procedure when a vertex is introduced or removed from the nice tree-decomposition (see [63] for more details). Therefore, each node $x \in V(T)$ defines in a natural way a graph $G_x \subseteq G$ in $\mathcal{F}_t$ with its associated rooted nice tree-decomposition. Let us now proceed to the description of the replacement algorithm.

We process the bags of $(T, \mathcal{X})$ in a bottom-up way until we encounter the first node x in $V(T)$ such that $|V(G_x)| = b + 1$. (Note that as $(T, \mathcal{X})$ is a nice tree-decomposition, when processing the bags in a bottom-up way, at most one new vertex is introduced at every step, and recall that by hypothesis $t \leqslant b$.) Let $\mathfrak{C}$ be the equivalence class of G_x according to $\sim_{\mathcal{E},g,t,\mathcal{G}}$. As $|V(G_x)| = b+1$, the graph G_x is contained in the repository $\mathfrak{R}$, so in constant time we can find in $\mathfrak{R}$ a progressive representative F of $\mathfrak{C}$ with at most b vertices and the corresponding transposition constant $\Delta_{\mathcal{E},g,t}(F, G_x) \leqslant 0$, where the inequality holds because

[2]Note that we slightly abuse notation when identifying t-protrusions and graphs in $\mathcal{F}_t$, as protrusions are defined as subsets of vertices of a graph. Nevertheless, this will not cause any confusion.

F is a progressive representative. Let G' be the graph obtained from G by replacing G_x with F, so we have that $|V(G')| < |V(G)|$. (Note that this replacement operation directly yields a rooted nice tree-decomposition of width at most $t-1$ of G'.) Since $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly, it follows that $G \sim_{\mathcal{E},g,t,\mathcal{G}} G'$ and that $\Delta_{\mathcal{E},g,t}(G', G) = \Delta_{\mathcal{E},g,t}(F, G_x) \leqslant 0$.

We recursively apply this replacement procedure on the resulting graph until we eventually obtain a t-protrusion H with at most b vertices such that $G \sim_{\mathcal{E},g,t,\mathcal{G}} H$. The corresponding transposition constant $\Delta_{\mathcal{E},g,t}(H, G)$ can be easily computed by summing up all the transposition constants given by each of the performed replacements. Since each of these replacements introduces a progressive representative, we have that $\Delta_{\mathcal{E},g,t}(H, G) \leqslant 0$. As we can assume that the total number of nodes in a nice tree-decomposition of G is $\mathcal{O}(n)$ [184, Lemma 13.1.2], the overall running time of the algorithm is $\mathcal{O}(n)$, where the constant hidden in the O-notation depends indeed exclusively on $\mathcal{E}, g, b, \mathcal{G}$, and t. □

6.3.3 Explicit protrusion replacer

We are now ready to piece everything together and state our main technical result, which can be interpreted as a generic *constructive* way of performing protrusion replacement with *explicit* size bounds. For our algorithms to be fully constructive, we restrict $\mathcal{G}$ to be the class of graphs that exclude some fixed graph H as a (topological) minor.

Theorem 6.1 *Let H be a fixed graph and let $\mathcal{G}$ be the class of graphs that exclude H as a (topological) minor. Let Π be a vertex-certifiable parameterized graph problem defined on $\mathcal{G}$, and suppose that we are given a Π-encoder $\mathcal{E}$, a function $g : \mathbb{N} \to \mathbb{N}$, and an integer $t \in \mathbb{N}$ such that $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly. Then, given an input graph (G, k) and a t-protrusion Y in G, we can compute in time $\mathcal{O}(|Y|)$ an equivalent instance $((G - (Y - \partial(Y))) \oplus Y', k')$, where $k' \leqslant k$ and Y' is a t-protrusion with $|Y'| \leqslant b(\mathcal{E}, g, t, \mathcal{G})$, where $b(\mathcal{E}, g, t, \mathcal{G})$ is the function defined in Lemma 6.3.*

Proof: By Lemma 6.1, the number of equivalence classes of the equivalence relation $\sim_{\mathcal{E},g,t,\mathcal{G}}$ is finite, and by Lemma 6.3 the size of a smallest progressive representative of any equivalence class of $\sim_{\mathcal{E},g,t,\mathcal{G}}$ is at most $b(\mathcal{E}, g, t, \mathcal{G})$. Therefore, we can apply Lemma 6.4 and deduce that, in time $\mathcal{O}(|Y|)$, we can find a t-protrusion Y' of size at most $b(\mathcal{E}, g, t, \mathcal{G})$ such that $Y \sim_{\mathcal{E},g,t,\mathcal{G}} Y'$, and the corresponding transposition constant $\Delta_{\mathcal{E},g,t}(Y', Y)$ with $\Delta_{\mathcal{E},g,t}(Y', Y) \leqslant 0$. Since $\mathcal{E}$ is a Π-encoder and $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly, it follows from Lemma 6.2 that $Y \equiv_{\Pi,t} Y'$ and that $\Delta_{\Pi,t}(Y', Y) = \Delta_{\mathcal{E},g,t}(Y', Y) \leqslant 0$. Therefore, if we set $k' := k + \Delta_{\Pi,t}(Y', Y)$, it follows that (G, k) and $((G - (Y - \partial(Y))) \oplus Y', k')$ are indeed equivalent instances of Π with $k' \leqslant k$ and $|Y'| \leqslant b(\mathcal{E}, g, t, \mathcal{G})$. □

The general recipe to use our framework on a parameterized problem Π defined on a class of graphs $\mathcal{G}$ is as follows: one has just to define the tables to solve Π via dynamic programming on graphs of bounded treewidth (that is, the encoder $\mathcal{E}$ and the function g), check that $\mathcal{E}$ is a Π-encoder and that $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly, and then Theorem 6.1 provides a linear-time algorithm that replaces large protrusions with graphs whose size is bounded by an explicit constant, and that updates the parameter of Π accordingly. This protrusion replacer can then be used, for instance, whenever one is able to find a linear protrusion decomposition of the input graphs of Π on some sparse graph class $\mathcal{G}$. In particular, Theorem 6.1 yields the following corollary.

Corollary 6.1 *Let H be a fixed graph, and let $\mathcal{G}$ be the class of graphs that exclude H as a (topological) minor. Let Π be a vertex-certifiable parameterized graph problem on $\mathcal{G}$, and suppose that we are given a Π-encoder $\mathcal{E}$, a function $g : \mathbb{N} \to \mathbb{N}$, and an integer $t \in \mathbb{N}$ such that $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ is DP-friendly. Then, given an instance (G,k) of Π together with an $(\alpha \cdot k, t)$-protrusion decomposition of G, we can construct a linear kernel for Π of size at most $(1 + b(\mathcal{E},g,t,\mathcal{G})) \cdot \alpha \cdot k$, where $b(\mathcal{E},g,t,\mathcal{G})$ is the function defined in Lemma 6.3.*

Proof: For $1 \leqslant i \leqslant \ell$, we apply the polynomial-time algorithm given by Theorem 6.1 to replace each t-protrusion Y_i with a graph Y_i' of size at most $b(\mathcal{E},g,t,\mathcal{G})$, and to update the parameter accordingly. In this way we obtain an equivalent instance (G',k') such that $G' \in \mathcal{G}$, $k' \leqslant k$, and $|V(G')| \leqslant |Y_0| + \ell \cdot b(\mathcal{E},g,t,\mathcal{G}) \leqslant (1 + b(\mathcal{E},g,t,\mathcal{G}))\alpha \cdot k$. □

Notice that once we fix the problem Π and the class of graphs $\mathcal{G}$ where Corollary 6.1 is applied, a kernel of size $c \cdot k$ can be derived with a concrete upper bound for the value of c. Notice that such a bound depends on the problem Π and the excluded (topological) minor H. In general, the bound can be quite big as it depends on the bound of Lemma 6.3, and this, in turn, depends on the bound of Lemma 6.1. However, as we see in the next section, more moderate estimates can be extracted for particular families of parameterized problems.

Before demonstrating the applicability of our framework by providing linear kernels for several families of problems on graphs excluding a fixed graph as a (topological) minor, we need another ingredient. Namely, the following result will be fundamental in order to find linear protrusion decompositions when a treewidth-modulator X of the input graph G is given, with $|X| = \mathcal{O}(k)$. It is a consequence of [J15, Lemma 3, Proposition 1, and Theorem 1] and, loosely speaking, the algorithm consists in marking the bags of a tree-decomposition of $G - X$ according to the number of neighbors in the set X. When the graph G is restricted to exclude a fixed graph H as a topological minor, it can be proved that the obtained protrusion decomposition is linear. All the details can be found in Chapter 5.

Theorem 6.2 (Kim et al. [J15]) *Let c, t be two positive integers, let H be an h-vertex graph, let G be an n-vertex H-topological-minor-free graph, and let k be a positive integer (typically corresponding to the parameter of a parameterized problem). If we are given a set $X \subseteq V(G)$ with $|X| \leqslant c \cdot k$ such that $\mathsf{tw}(G - X) \leqslant t$, then we can compute in time $\mathcal{O}(n)$ an $((\alpha_H \cdot t \cdot c) \cdot k, 2t + h)$-protrusion decomposition of G, where α_H is a constant depending only on H, which is upper-bounded by $40h^2 2^{5h \log h}$.*

As mentioned in Section 6.3.2, if $\mathcal{G}$ is a graph class whose membership can be expressed in MSO logic, then $\sim_{\mathcal{G},t}$ has a finite number of equivalence classes, namely $r_{\mathcal{G},t}$. In our applications, we will be only concerned with families of graphs $\mathcal{G}$ that exclude some fixed h-vertex graph H as a (topological) minor. In this case, using standard dynamic programming techniques, it can be shown that $r_{\mathcal{G},t} \leqslant 2^{t \log t} \cdot h^t \cdot 2^{h^2}$. The details can be found in the encoder described in Section 6.6.1 for the $\mathcal{F}$-Deletion problem.

6.4 An explicit linear kernel for r-Dominating Set

Let $r \geqslant 1$ be a fixed integer. We define the r-Dominating Set problem as follows.

r-Dominating Set	
Instance:	A graph $G = (V, E)$ and a non-negative integer k.
Parameter:	The integer k.
Question:	Does G have a set $S \subseteq V$ with $\|S\| \leqslant k$ and such that every vertex in $V \setminus S$ is within distance at most r from some vertex in S?

For $r = 1$, the r-Dominating Set problem corresponds to Dominating Set. Our encoder for r-Dominating Set is strongly inspired by the work of Demaine et al. [102], and it generalizes the one given for Dominating Set in the running example of Section 6.3. The encoder for r-Dominating Set, which we call $\mathcal{E}_{r\mathrm{DS}} = (\mathcal{C}_{r\mathrm{DS}}, L_{\mathcal{C}_{r\mathrm{DS}}})$, is described in Section 6.4.1, and we show how to construct the linear kernel in Section 6.4.2. We will use the shortcut rDS for r-Dominating Set.

6.4.1 Description of the encoder

Let G be a boundaried graph with boundary $\partial(G)$ and let $I = \Lambda(G)$. The function $\mathcal{C}_{r\mathrm{DS}}$ maps I to a set $\mathcal{C}_{r\mathrm{DS}}(I)$ of $\mathcal{C}_{r\mathrm{DS}}$-encodings. Each $R \in \mathcal{C}_{r\mathrm{DS}}(I)$ maps I to an $|I|$-tuple in $\{0, \uparrow 1, \downarrow 1, \ldots, \uparrow r, \downarrow r\}^{|I|}$, and thus the coordinates of the tuple are in one-to-one correspondence with the vertices of $\partial(G)$. For a vertex $v \in \partial(G)$ we denote by $R(v)$ its coordinate in the $|I|$-tuple. For a subset S of vertices of G, we say that (G, S, R) belongs to the language $L_{\mathcal{C}_{r\mathrm{DS}}}$ (or that S is a *partial r-dominating set satisfying R*) if :

- for every vertex $v \in V(G) \setminus \partial(G)$, either $d_G(v, S) \leqslant r$ or there exists $w \in \partial(G)$ such that $R(w) = \uparrow j$ and $d_G(v, w) + j \leqslant r$; and
- for every vertex $v \in \partial(G)$: $R(v) = 0$ implies that $v \in S$, and if $R(v) = \downarrow i$ for $1 \leqslant i \leqslant r$, then there exists either $w \in S$ such that $d_G(v, w) \leqslant i$ or $w \in \partial(G)$ such that $R(w) = \uparrow j$ and $d_G(v, w) + j \leqslant i$.

Observe that if S is a partial r-dominating set satisfying R, then $S \cap \partial(G)$ contains the set of vertices $\{v \in \partial(G) \mid R(v) = 0\}$, but it may also contain other vertices of $\partial(G)$. As the optimization version of r-Dominating Set is a minimization problem, by Equation (6.1) the function $f_G^{\mathcal{C}_{r\mathrm{DS}}}(R)$ associates with a $\mathcal{C}_{r\mathrm{DS}}$-encoding R the minimum size of a partial r-dominating set S satisfying R. By definition of $\mathcal{E}_{r\mathrm{DS}}$, it is clear that

$$s_{\mathcal{E}_{r\mathrm{DS}}}(t) \leqslant (2r+1)^t. \tag{6.10}$$

Lemma 6.5 *The encoder $\mathcal{E}_{r\mathrm{DS}}$ is an rDS-encoder. Furthermore, if $\mathcal{G}$ is an arbitrary class of graphs and $g(t) = t$, then the equivalence relation $\sim^*_{\mathcal{E}_{r\mathrm{DS}},g,t,\mathcal{G}}$ is DP-friendly.*

Before providing the proof of Lemma 6.5, we will first state a general fact, which will be useful in order to prove that an encoder is DP-friendly.

Fact 1 *To verify that an equivalence relation $\sim^*_{\mathcal{E},g,t,\mathcal{G}}$ satisfies Definition 6.11, property (i) can be replaced with $G \sim^*_{\mathcal{E},g,t} G'$. That is, if $G \sim^*_{\mathcal{E},g,t} G'$, then $G \sim^*_{\mathcal{E},g,t,\mathcal{G}} G'$ as well.*

Proof: Assume that $G \sim^*_{\mathcal{E},g,t} G'$, and we want to deduce that $G \sim^*_{\mathcal{E},g,t,\mathcal{G}} G'$, that is, we just have to prove that $G \sim_{\mathcal{G},t} G'$. Let H be a t-boundaried graph, and we need to prove that $G \oplus H \in \mathcal{G}$ if and only if $G' \oplus H \in \mathcal{G}$. Let G_B, G'_B, G^- be such that $G = G_B \oplus G^-$ and $G' = G'_B \oplus G^-$. We have that $G \oplus H = (G_B \oplus G^-) \oplus H = G_B \oplus (G^- \oplus H)$, and similarly we have that $G'_B \oplus (G^- \oplus H) = (G'_B \oplus G^-) \oplus H = G' \oplus H$. Since $G_B \sim_{\mathcal{G},t} G'_B$, it follows that $G \oplus H = G_B \oplus (G^- \oplus H) \in \mathcal{G}$ if and only if $G'_B \oplus (G^- \oplus H) = G' \oplus H \in \mathcal{G}$. □

Proof of Lemma 6.5: Let us first prove that $\mathcal{E}_{r\mathrm{DS}} = (\mathcal{C}_{r\mathrm{DS}}, L_{\mathcal{C}_{r\mathrm{DS}}})$ is an rDS-encoder. Note that there is a unique 0-tuple $R_\emptyset \in \mathcal{C}_{r\mathrm{DS}}(\emptyset)$, and by definition of $L_{\mathcal{C}_{r\mathrm{DS}}}$, $(G, S, R_\emptyset) \in L_{\mathcal{C}_{r\mathrm{DS}}}$ if and only if S is an r-dominating set of G. Let us now prove that the equivalence relation $\sim^*_{\mathcal{E}_{r\mathrm{DS}},g,t,\mathcal{G}}$ is DP-friendly for $g(t) = t$.

As in Definition 6.11, let G be a t-boundaried graph with boundary A, let B be any separator $B \subseteq V(G)$ with $|B| \leqslant t$, and let G_B be obtained from any collection of connected components of $G - B$ by adding the set B, such that $A \cap V(G_B) \subseteq B$, which we consider as a t-boundaried graph with boundary B. We define H to be the t-boundaried graph induced by $V(G) \setminus (V(G_B) \setminus B)$, and with boundary B (that is, we forget boundary A) labeled as in G_B. Let G'_B be a t-boundaried graph such that $G_B \sim^*_{\mathcal{E}_{r\mathrm{DS}},g,t,\mathcal{G}} G'_B$. Let $G' := H \oplus G'_B$ with boundary A. See Figure 6.3 for an illustration.

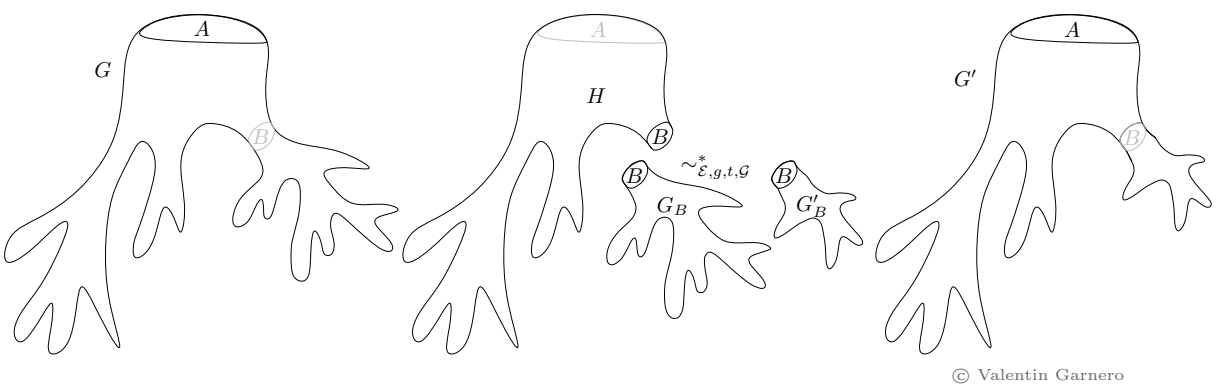

Figure 6.3: Graphs G and G' in the proof of Lemma 6.5.

We claim that the encoder $\mathcal{E}_{r\mathrm{DS}}$ is g-confined for $g(t) = t$. Indeed, consider an arbitrary encoding $R_A \in \mathcal{C}_{r\mathrm{DS}}(\Lambda(G))$ and the encoding R_0 satisfying $R_0(v) = 0$ for every $v \in A$. Let $S_0 \subseteq V(G)$ be a minimum-sized partial r-dominating set satisfying R_0, i.e., such that $(G, S_0, R_0) \in L_{\mathcal{C}_{r\mathrm{DS}}}$. Observe that S_0 also satisfies R_A, i.e., $(G, S_0, R_A) \in L_{\mathcal{C}_{r\mathrm{DS}}}$. It then follows that $f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_0) = \max_{R_A} f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_A)$. Moreover, let $S \subseteq V(G)$ be a minimum-sized partial r-dominating set satisfying R_A, i.e., such that $(G, S, R_A) \in L_{\mathcal{C}_{r\mathrm{DS}}}$. Then R_0 is also

satisfied by $S \cup A$. It follows that $f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_0) - \min_{R_A} f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_A) \leqslant |A| \leqslant t$, proving that the encoder is indeed g-confined.

We want to show that $G \sim^*_{\mathcal{E}_{r\mathrm{DS}},g,t,\mathcal{G}} G'$ and that $\Delta_{\mathcal{E}_{r\mathrm{DS}},g,t}(G,G') = \Delta_{\mathcal{E}_{r\mathrm{DS}},g,t}(G_B,G'_B)$. According to Fact 1, we can consider the relation $\sim^*_{\mathcal{E}_{r\mathrm{DS}},g,t}$ (that is, we do not need to consider the refinement with respect to the class of graphs $\mathcal{G}$), and due to the g-confinement it holds that $f_G^{\mathcal{E}_{r\mathrm{DS}},g} = f_G^{\mathcal{E}_{r\mathrm{DS}}}$ for $g(t) = t$. Hence it suffices to prove that $f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_A) = f_{G'}^{\mathcal{E}_{r\mathrm{DS}}}(R_A) + \Delta_{\mathcal{E}_{r\mathrm{DS}},g,t}(G_B,G'_B)$ for all $R_A \in \mathcal{C}_{r\mathrm{DS}}(\Lambda(G))$.

Let $R_A \in \mathcal{C}_{r\mathrm{DS}}(\Lambda(G))$ be a $\mathcal{C}_{r\mathrm{DS}}$-encoding defined on A. First assume that $f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_A) \neq +\infty$, that is, $R_A \in \mathcal{C}^*_{r\mathrm{DS},G}(\Lambda(G))$. Let $S = D \cup D_H$ be a partial r-dominating set of size $f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_A)$ of G satisfying R_A, with $D \subseteq V(G_B)$ and $D_H \subseteq V(H) \setminus B$. We use S to construct a $\mathcal{C}_{r\mathrm{DS}}$-encoding $R_B \in \mathcal{C}_{r\mathrm{DS}}(\Lambda(G_B))$ defined on B, satisfied by D as follows. Let $v \in B$:

- if $v \in S$, then $R_B(v) = 0$;
- otherwise, if there is either a shortest path from v to S of length i with its first edge in G_B, or a path from v to any $a \in A$ such that $R_A(a) =\uparrow j$ of length $i - j$, also with its first edge in G_B, then $R_B(v) =\downarrow i$;
- otherwise, $R_B(v) =\uparrow i$ where $i = d_G(v,S)$ or $i = d_G(v,a) + j$ such that $R_A(a) =\uparrow j$ (the first edge of any shortest path from v to S is not in G_B).

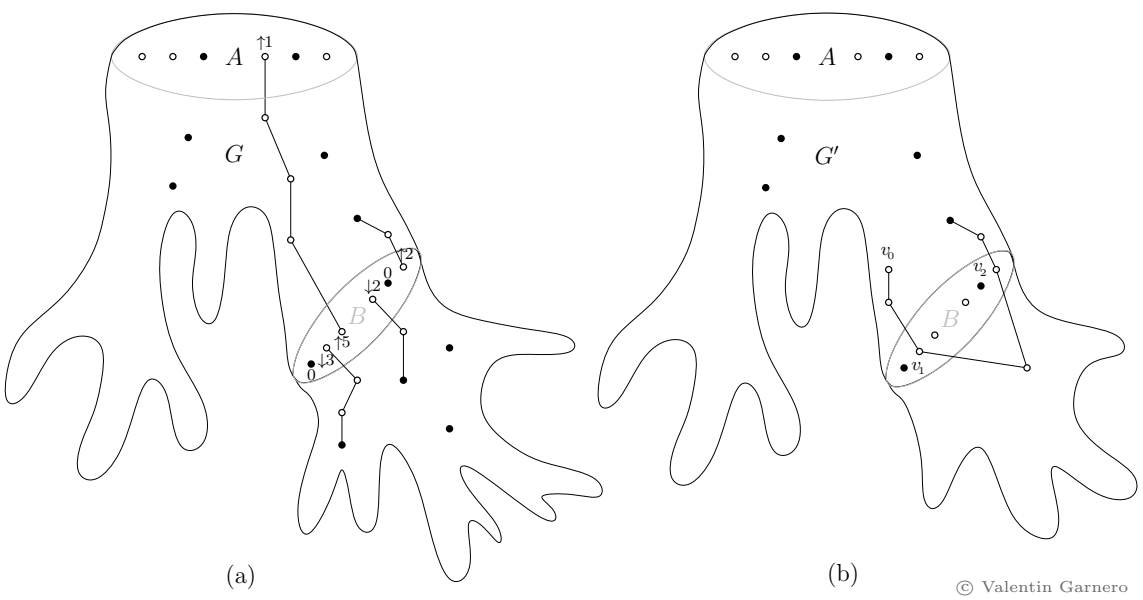

Figure 6.4: Illustration of the proof of Lemma 6.5. Black vertices belong to the solution: (a) construction of the $\mathcal{C}_{r\mathrm{DS}}$-encoding $R_B \in \mathcal{C}_{r\mathrm{DS}}(\Lambda(G_B))$; and (b) construction of the corresponding paths.

See Figure 6.4(a) for an illustration of the construction of the $\mathcal{C}_{r\mathrm{DS}}$-encoding $R_B \in \mathcal{C}_{r\mathrm{DS}}(\Lambda(G_B))$ described above.

Observe that by construction of R_B, $|D| \geqslant f_{G_B}^{\mathcal{E}_{r\mathrm{DS}}}(R_B)$. Let D' be a subset of vertices of G'_B of minimum size such that $(G'_B, D', R_B) \in L_{\mathcal{C}_{r\mathrm{DS}}}$, that is, $|D'| = f_{G'_B}^{\mathcal{E}_{r\mathrm{DS}}}(R_B)$. As $G_B \sim_{\mathcal{E}_{r\mathrm{DS}},g,t} G'_B$, we have $|D'| = f_{G_B}^{\mathcal{E}_{r\mathrm{DS}}}(R_B) + \Delta_{\mathcal{E}_{r\mathrm{DS}},g,t}(G_B, G'_B)$ and therefore $|D' \cup D_H| = f_{G_B}^{\mathcal{E}_{r\mathrm{DS}}}(R_A) + \Delta_{\mathcal{E}_{r\mathrm{DS}},g,t}(G_B, G'_B) + |D_H| \leqslant f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_A) + \Delta_{\mathcal{E}_{r\mathrm{DS}},g,t}(G_B, G'_B)$.

Let us now prove that $S' = D' \cup D_H$ is a partial r-dominating set of G' satisfying R_A. According to the definition of $\mathcal{E}_{r\mathrm{DS}}$, we distinguish vertices in $V(G') \setminus A$ and in A.

We start with vertices not in A. For any vertex $v \in V(G') \setminus (A \cup S')$, we consider the following iterative process that builds a path of length at most r from v to S' or a path of length at most $r - i$ from v to $a \in A$ such that $R(a) =\uparrow i$. At step $j \geqslant 0$, we identify a vertex $v_j \in B$. We initially set $v_0 = v$. If $v_0 \in V(G'_B)$, we can assume that $d_{G'_B}(v_0, D') > r$, as otherwise we are done. As D' satisfies R_B, this implies that B contains a vertex v_1 such that $R_B(v_1) =\uparrow i_1$ and $d_{G'_B}(v_0, v_1) + i_1 \leqslant r$. Similarly, if $v_0 \in V(H) \setminus B$, we can assume that $d_H(v_0, D_H) > r$ and $d_H(v_0, a) > r - i$ for any $a \in A$ such that $R_A(a) =\uparrow i$, as otherwise we are done. As $S = D \cup D_H$ is a partial r-dominating set of G satisfying R_A, any shortest path P (of length at most r) between v_0 and S and any path (of length $r - i$) between v_0 and $a \in A$ such that $R_A(a) =\uparrow i$, contains a vertex of B incident to an edge of G_B. Let v_1 be the first such vertex of P. By definition of R_B, we have that $R_B(v_1) =\downarrow i_1$ with $d_H(v_0, v_1) + i_1 \leqslant r$. Let us now consider v_j with $j \geqslant 1$, and denote by l_j the length of the path we discovered from v_0 to v_j. We need to prove that $l_j + i_j \leqslant r$ (or $l_j + i_j \leqslant r - i$ in the other case) is an invariant of the process. As we argued, it is true for $j = 1$, so assume it holds at step j. We consider two cases:

1. $R_B(v_j) =\downarrow i_j$: We can assume that $d_{G'_B}(v_j, D') > i_j$, otherwise we are done as by construction it holds that $l_j + i_j \leqslant r$ (or $l_j + i_j \leqslant r - i$ in the other case). So as D' is a partial r-dominating set satisfying R_B, there exists a vertex $v_{j+1} \in B$ such that $R_B(v_{j+1}) =\uparrow i_{j+1}$ and $d_{G'_B}(v_i, v_{j+1}) + i_{j+1} \leqslant i_j$. As $l_{j+1} = l_j + d_{G'_B}(v_i, v_{j+1})$, it follows that $l_{j+1} + i_{j+1} \leqslant r$ (or $l_{j+1} + i_{j+1} \leqslant r - i$ in the other case). See Figure 6.4(b) for an illustration of this case.

2. $R_B(v_j) =\uparrow i_j$: We can assume that $d_H(v_j, D_H) > i_j$ and $d_H(v_j, a) > i_j - i$ for any $a \in A$ such that $R_A(a) =\uparrow i$, otherwise we are done as by construction it holds that $l_j + i_j \leqslant r$ (or $l_j + i_j \leqslant r - i$ in the other case). As by definition of the encoding R_B, $d_G(v_j, S) = i_j$, any shortest path P between v_j and S (or $a \in A$) uses a vertex of B incident to an edge of G_B. Let v_{j+1} be the first such vertex of P. Then $R_B(v_{j+1}) =\downarrow i_{j+1}$ with $d_H(v_j, v_{j+1}) + i_{j+1} \leqslant i_j$. As $l_{j+1} = l_j + d_H(v_i, v_{j+1})$, it follows that $l_{j+1} + i_{j+1} \leqslant r$ (or $l_j + i_j \leqslant r - i$ in the other case).

Observe that the process ends, since the parameter $r - (l_j + i_j)$ is strictly decreasing.

We now consider vertices of A. Note that as $\partial(G_B) = \partial(G'_B)$, it holds that $\partial(G) = \partial(G')$ as well. In particular, any vertex $v \in A$ is also in H. If $R_A(v) = 0$ since $S = D \cup D_H$ satisfies A, $v \in D_H$ and hence, $v \in S' = D' \cup D_H$. If $R_A(v) =\downarrow i$, the iterative process above built a path from v to S' of length at most r, or from v to $a \in A$ with $R_A(a) =\uparrow j$ of length at most $r - i - j$.

It follows that $S' = D' \cup D_H$ is a partial r-dominating set of size at most $f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_A) + \Delta_{\mathcal{E}_{r\mathrm{DS}},g,t}(G_B, G'_B)$ satisfying R_A, as we wanted to prove.

Finally, assume that $f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_A) = +\infty$. Then it holds that $f_{G'}^{\mathcal{E}_{r\mathrm{DS}}}(R_A) = +\infty$ as well. Indeed, suppose that $f_{G'}^{\mathcal{E}_{r\mathrm{DS}}}(R_A)$ is finite. Then, given a partial r-dominating set of G' satisfying R_A, by the argument above we could construct a partial r-dominating set of G satisfying R_A, contradicting that $f_G^{\mathcal{E}_{r\mathrm{DS}}}(R_A) = +\infty$.

Therefore, we can conclude that $G \sim^*_{\mathcal{E}_{r\mathrm{DS}},g,t} G'$, and hence the equivalence relation $\sim^*_{\mathcal{E}_{r\mathrm{DS}},g,t,\mathcal{G}}$ is DP-friendly for $g(t) = t$. □

6.4.2 Construction of the kernel

We proceed to construct a linear kernel for r-Dominating Set when the input graph excludes a fixed apex graph H as a minor. Toward this end, we use the fact that this problem satisfies the contraction-bidimensionality and separability conditions required in order to apply the results of Fomin et al. [140]. In the following proposition we specify the result in [140, Lemma 3.3] for the case of r-Dominating Set while making visible the dependance on r and the size h of the excluded apex graph. The polynomial-time algorithm follows from Fomin et al. [139, Lemma 3.2], whose proof makes use of the polynomial-time approximation algorithm for the treewidth of general graphs by Feige et al. [122].

Proposition 6.3 *Let $r \geqslant 1$ be an integer, let H be an h-vertex apex graph, and let $r\mathrm{DS}_H$ be the restriction of the r-Dominating Set problem to input graphs which exclude H as a minor. If $(G,k) \in r\mathrm{DS}_H$, then there exists a set $X \subseteq V(G)$ such that $|X| = \mathcal{O}(r \cdot f_c(h) \cdot k)$ and $\mathsf{tw}(G - X) = \mathcal{O}(r \cdot (f_c(h))^2)$, where f_c is the function in Proposition 6.2. Moreover, given an instance (G,k) of $r\mathrm{DS}_H$, there is a polynomial-time algorithm that either finds such a set X or correctly reports that (G,k) is a* No*-instance.*

We are now ready to present the linear kernel for r-Dominating Set.

Theorem 6.3 *Let $r \geqslant 1$ be an integer, let H be an h-vertex apex graph, and let $r\mathrm{DS}_H$ be the restriction of the r-Dominating Set problem to input graphs which exclude H as a minor. Then $r\mathrm{DS}_H$ admits a constructive linear kernel of size at most $f(r,h) \cdot k$, where f is an explicit function depending only on r and h, defined in Equation (6.11) below.*

Proof: Given an instance (G,k) of $r\mathrm{DS}_H$, we run the polynomial-time algorithm given by Proposition 6.3 to either conclude that (G,k) is a No-instance or to find a set $X \subseteq V(G)$ such that $|X| = \mathcal{O}(r \cdot f_c(h) \cdot k)$ and $\mathsf{tw}(G - X) = \mathcal{O}(r \cdot (f_c(h))^2)$. In the latter case, we use the set X as input to the algorithm given by Theorem 6.2, which outputs in linear time a $(r^2 \cdot 2^{\mathcal{O}(h \log h)} \cdot (f_c(h))^3 \cdot k, \mathcal{O}(r \cdot (f_c(h))^2))$-protrusion decomposition of G. We now consider the encoder $\mathcal{E}_{r\mathrm{DS}} = (\mathcal{C}_{r\mathrm{DS}}, L_{\mathcal{C}_{r\mathrm{DS}}})$ defined in Section 6.4.1. By Lemma 6.5, $\mathcal{E}_{r\mathrm{DS}}$ is an rDS-encoder and $\sim^*_{\mathcal{E}_{r\mathrm{DS}},g,t,\mathcal{G}}$ is DP-friendly, where $\mathcal{G}$ is the class of H-minor-free graphs and $g(t) = t$. By Equation (6.10) in Section 6.4.1, we have that $s_{\mathcal{E}_{r\mathrm{DS}}}(t) \leqslant (2r+1)^t$. Therefore, we are in position to apply Corollary 6.1 and obtain a linear kernel for $r\mathrm{DS}_H$ of size at most

$$r^2 \cdot 2^{\mathcal{O}(h \log h)} \cdot (f_c(h))^3 \cdot b\left(\mathcal{E}_{r\mathrm{DS}}, g, \mathcal{O}(r \cdot (f_c(h))^2), \mathcal{G}\right) \cdot k \,, \tag{6.11}$$

where $b\left(\mathcal{E}_{r\mathrm{DS}}, g, \mathcal{O}(r \cdot (f_c(h))^2), \mathcal{G}\right)$ is the function defined in Lemma 6.3. □

It can be routinely checked that, once the excluded apex graph H is fixed, the dependance on r of the multiplicative constant involved in the upper bound of Equation (6.11) is of the form $2^{2^{2^{\mathcal{O}(r \cdot \log r)}}}$, that is, it depends triple-exponentially on the integer r.

6.5 An explicit linear kernel for r-Scattered Set

Let $r \geqslant 1$ be a fixed integer. Given a graph G and a set $S \subseteq V(G)$, we say that S is an *r-independent set* if any two vertices in S are at distance greater than r in G. We define the r-Scattered Set problem, which can be seen as a generalization of Independent Set, as follows.

r-Scattered Set	
Instance:	A graph G and a non-negative integer k.
Parameter:	The integer k.
Question:	Does G have a $2r$-independent set of size at least k?

Our encoder for r-Scattered Set (or equivalently, for $2r$-Independent Set) is inspired from the proof of Fomin et al. [63] that the problem has FII, and can be found in Section 6.5.1. We then show how to construct the linear kernel in Section 6.5.2.

6.5.1 Description of the encoder

Equivalently, we proceed to present an encoder for the r-Independent Set problem, which we abbreviate as rIS. Let G be a boundaried graph with boundary $\partial(G)$ and denote $I = \Lambda(G)$. The function $\mathcal{C}_{r\mathrm{IS}}$ maps I to a set $\mathcal{C}_{r\mathrm{IS}}(I)$ of $\mathcal{C}_{r\mathrm{IS}}$-encodings. Each $R \in \mathcal{C}_{r\mathrm{IS}}(I)$ maps I to an $|I|$-tuple the coordinates of which are in one-to-one correspondence with the vertices of $\partial(G)$. The coordinate $R(v)$ of vertex $v \in \partial(G)$ is a $(|I|+1)$-tuple in $(d_S, d_{v_1}, \ldots, d_{v_{|I|}}) \in \{0, 1, \ldots, r, r+1\}^{|I|+1}$. For a subset S of vertices of G, we say that (G, S, R) belongs to the language $L_{\mathcal{C}_{r\mathrm{IS}}}$ (or that S is a *partial r-independent set satisfying R*) if:

- for every pair of vertices $v \in S$ and $w \in S$, $d_G(v, w) > r$;
- for every vertex $v \in \partial(G)$: $d_G(v, S) \geqslant d_S$ and for every $w \in \partial(G)$, $d_G(v, w) \geqslant d_w$.

As r-Independent Set is a maximization problem, by Equation (6.2) the function $f_G^{\mathcal{E}_{r\mathrm{IS}}}$ associates to each encoding R the maximum size of a partial r-independent set S satisfying R. By definition of $\mathcal{E}_{r\mathrm{IS}}$ it is clear that

$$s_{\mathcal{E}_{r\mathrm{IS}}}(t) \leqslant (r+2)^{t(t+1)}. \tag{6.12}$$

Lemma 6.6 *The encoder $\mathcal{E}_{r\mathrm{IS}} = (\mathcal{C}_{r\mathrm{IS}}, L_{\mathcal{C}_{r\mathrm{IS}}})$ described above is an rIS-encoder. Furthermore, if $\mathcal{G}$ is an arbitrary class of graphs and $g(t) = 2t$, then the equivalence relation $\sim^*_{\mathcal{E}_{r\mathrm{IS}},g,t,\mathcal{G}}$ is DP-friendly.*

Proof: We first prove that $\mathcal{E}_{r\mathrm{IS}} = (\mathcal{C}_{r\mathrm{IS}}, L_{\mathcal{C}_{r\mathrm{IS}}})$ is an rIS-encoder. There is a unique 0-tuple $R_\emptyset \in \mathcal{C}_{r\mathrm{IS}}(\emptyset)$, and by definition of $L_{\mathcal{C}_{r\mathrm{IS}}}$, $(G, S, R_\emptyset) \in L_{\mathcal{C}_{r\mathrm{IS}}}$ if and only if S is an r-independent set of G.

Let G, G' with boundary A and H, G_B, G'_B with boundary B be the graphs as defined in the proof of Lemma 6.5 (see Figure 6.3).

Let R_0 be the encoding satisfying $R_0(v) = (0, 0, \ldots, 0)$ for every $v \in B$. Observe that if S is a maximum partial r-independent set satisfying an encoding $R_B \in \mathcal{C}_{r\mathrm{IS}}(\Lambda(G_B))$, then S also satisfies R_0. It follows that $f_G^{\mathcal{E}_{r\mathrm{IS}}}(R_0) = \max_{R_B} f_G^{\mathcal{E}_{r\mathrm{IS}}}(R_B)$ (and thus $f_G^{\mathcal{E}_{r\mathrm{IS}}}(R_0) = f_G^{\mathcal{E}_{r\mathrm{IS}},g}(R_0)$).

We want to show that $G \sim^*_{\mathcal{E}_{r\mathrm{IS}},g,t,\mathcal{G}} G'$ and that $\Delta_{\mathcal{E}_{r\mathrm{IS}},g,t}(G, G') = \Delta_{\mathcal{E}_{r\mathrm{IS}},g,t}(G_B, G'_B)$. According to Fact 1, it is enough to consider the relation $\sim^*_{\mathcal{E}_{r\mathrm{IS}},g,t}$. To that aim, we will prove that $f_G^{\mathcal{E}_{r\mathrm{IS}},g}(R_A) = f_{G'}^{\mathcal{E}_{r\mathrm{IS}},g}(R_A) + \Delta_{\mathcal{E}_{r\mathrm{IS}},g,t}(G_B, G'_B)$ for all $R_A \in \mathcal{C}_{r\mathrm{IS}}(\Lambda(G))$ and for $g(t) = 2t$.

Let $R_A \in \mathcal{C}_{r\mathrm{IS}}(\Lambda(G))$ be a $\mathcal{C}_{r\mathrm{IS}}$-encoding defined on A. First assume that $f_G^{\mathcal{E}_{r\mathrm{IS}},g}(R_A) \neq -\infty$, that is, $R_A \in \mathcal{C}^*_{r\mathrm{IS},G}(\Lambda(G))$. Let $S = I \cup I_H$ be a partial r-independent set of size $f_G^{\mathcal{E}_{r\mathrm{IS}},g}(R_A)$ of G, with $I \subseteq V(G_B)$ and $I_H \subseteq V(H) \setminus B$. An encoding $R_B \in \mathcal{C}_{r\mathrm{IS}}(\Lambda(G_B))$, satisfied by S is defined as follows. Let $v \in B$, then $R_B(v) = (d_S, d_{v_1}, \ldots, d_{v_{|B|}})$ where

- $d_S = d_{G_B}(v, I)$; and
- for $i \in \{1, \ldots, |B|\}$, $d_{v_i} = \min\{d_{G_B}(v, v_i), r + 1\}$ (remind that $v_i \in \partial(G)$).

Fact 2 *For the R_B defined above, it holds that $f_{G_B}^{\mathcal{E}_{r\mathrm{IS}},g}(R_B) \neq -\infty$, where $g(t) = 2t$.*

Proof: Let $I_0 \subseteq V(G_B)$ be a maximum partial r-independent set satisfying R_0, i.e., $(G_B, I_0, R_0) \in L_{\mathcal{C}_{r\mathrm{IS}}}$. Let us define $I^* = I \setminus N_{r/2}(B)$, $I_0^* = I_0 \setminus N_{r/2}(B)$ and $I_H^* = I_H \setminus N_{r/2}(B)$. By the pigeon-hole principle, it is easy to see that $|I_0^*| \geqslant |I_0| - t$ (otherwise B would contain a vertex at distance at most $r/2$ from two distinct vertices of I_0). Likewise, $|I_H^*| \geqslant |I_H| - t$. Now observe that $I_0^* \cup I_H^*$ is an r-independent set of G and therefore $|I_0^*| + |I_H^*| \leqslant |S|$ (1) (as S was chosen as a maximum r-independent set of G). As S is the disjoint union of I and I_H, we also have that $|S| \leqslant |I| + |I_H^*| + t$ (2). Combining (1) and (2), we obtain that $|I_0^*| \leqslant |I| + t$ and therefore $|I_0| \leqslant |I| + 2t$. It follows that $f_{G_B}^{\mathcal{E}_{r\mathrm{IS}}}(R_B) = f_{G_B}^{\mathcal{E}_{r\mathrm{IS}},g}(R_B)$, proving the fact. □

Observe that by construction of R_B, $|I| \leqslant f_{G_B}^{\mathcal{E}_{r\mathrm{IS}},g}(R_B)$. Consider a subset of vertices I' of G'_B of maximum size such that $(G'_B, I', R_B) \in L_{\mathcal{C}_{r\mathrm{IS}}}$, that is $|I'| = f_{G'_B}^{\mathcal{E}_{r\mathrm{IS}},g}(R_B)$. As $G_B \equiv_{\mathcal{E}_{r\mathrm{IS}},t} G'_B$, by the above claim, we have $|I'| = f_{G_B}^{\mathcal{E}_{r\mathrm{IS}},g}(R_B) + \Delta_{\mathcal{E}_{r\mathrm{IS}},g,t}(G_B, G'_B)$ and therefore $|I' \cup I_H| = f_{G_B}^{\mathcal{E}_{r\mathrm{IS}},g}(R_B) + \Delta_{\mathcal{E}_{r\mathrm{IS}},g,t}(G_B, G'_B) + |I_H| \geqslant f_G^{\mathcal{E}_{r\mathrm{IS}},g}(R_A) + \Delta_{\mathcal{E}_{r\mathrm{IS}},g,t}(G_B, G'_B)$.

Let us prove that $S' = I' \cup I_H$ is a partial r-independent set of G' satisfying R_A. Following the definition of $\mathcal{E}_{r\mathrm{IS}}$, we have to verify two kinds of conditions: those on vertices in S' and those on vertices in A. We start with vertices in S'. Let P be a shortest path in G between two vertices $v \in S'$ and $w \in S'$. We partition P into maximal subpaths $P_1, \ldots, P_q$

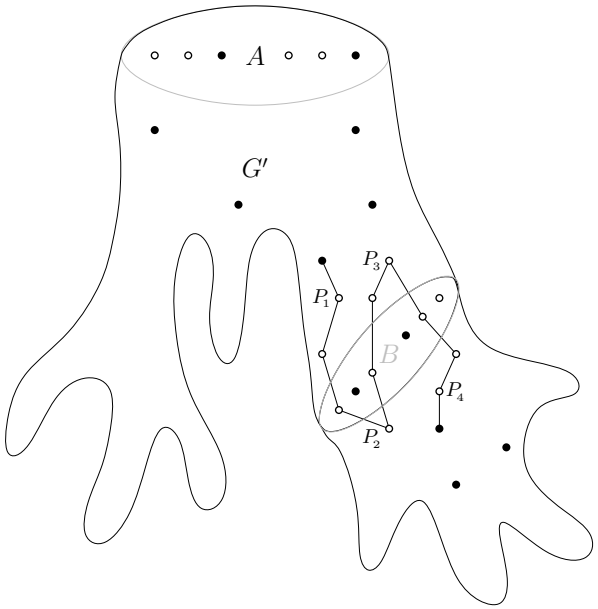

Figure 6.5: Illustration in the proof of Lemma 6.6. The black vertices belong to the solution.

such that P_j (for $1 \leqslant j \leqslant q$) is either a path of G'_B (called a G'_B-path) or of H (called an H-path). An illustration of these paths can be found in Figure 6.5. If $q = 1$, then $d_{G'}(v, w) > r$ follows from the fact that I_H and I' are respectively r-independent sets of H and G'_B (a partial r-independent set is an r-independent set). So assume that $q > 1$. Observe that every H-subpath is a path in G. By the choice of S', observe that the length of every G'_B-subpath is at least the distance in G_B between its extremities. We consider three cases:

- $v, w \in V(H) \setminus B$: By the observations above, the length of P is at least $d_G(v, w)$. As $v, w \in I_H$, we obtained that $d_{G'}(v, w) \geqslant d_G(v, w) > r$.
- $v \in V(H) \setminus B$ and $w \in V(G'_B)$: Let u be the last vertex of P_{q-1}. By the same argument as in the previous case we have $d_{G'}(v, u) \geqslant d_G(v, u)$. Now by the choice of S', observe that $d_{G'_B}(u, w) \geqslant d_{G_B}(u, I)$. So the length of P is at least the distance in G from v to a vertex $w' \in I$, we can conclude that $d_{G'}(v, w) > r$.
- $v, w \in V(G'_B)$: Let u_1 and u_q be respectively the last vertex of P_1 and the first vertex of P_q. By the same argument as above, we have that $d_{G'}(u_1, u_q) \geqslant d_G(u_1, u_q)$. By the choice of S', we have that $d_{G'_B}(u_1, v) \geqslant d_{G_B}(u_1, I)$ and $d_{G'_B}(u_q, w) \geqslant d_{G_B}(u_q, I)$. So the length of P is a least the distance in G between two vertices $v' \in I$ and $w' \in I$. We can therefore conclude that $d_{G'}(v, w) > r$.

We now consider vertices of A. Let $v \in A$ such that $R_A(v) = (d_S, d_{v_1}, \ldots, d_{v_{|A|}})$. Let P be a shortest path in G' between vertices $v \in A$ and $w \in S'$, similarly to the previous argumentation (two first items) $d_{G'}(v, w) > d_S$. Now let P be a a shortest path in G' between vertices $v \in A$ and $v_i \in A$ similarly to the previous argumentation (first item) $d_{G'}(v, w) > d_{v_i}$.

It follows that $S' = I' \cup I_H$ is a partial r-independent set of size at least $f_G^{\mathcal{E}_{r\mathrm{IS}},g}(R_A) + \Delta_{\mathcal{E}_{r\mathrm{IS}},g,t}(G_B, G'_B)$ satisfying R_A, as we wanted to prove.

Finally, assume that $f_G^{\mathcal{E}_{r\mathrm{IS}}}(R_A) = -\infty$. Then it holds that $f_{G'}^{\mathcal{E}_{r\mathrm{IS}}}(R_A) = -\infty$ as well. Indeed, suppose that $f_{G'}^{\mathcal{E}_{r\mathrm{IS}}}(R_A)$ is finite. Then, given a partial r-independent set of G' satisfying R_A, by the argument above we could construct a partial r-independent set of G satisfying R_A, contradicting that $f_G^{\mathcal{E}_{r\mathrm{IS}}}(R_A) = -\infty$.

Therefore, we can conclude that $G \sim^*_{\mathcal{E}_{r\mathrm{IS}},g,t} G'$, and hence the equivalence relation $\sim^*_{\mathcal{E}_{r\mathrm{IS}},g,t,\mathcal{G}}$ is DP-friendly for $g(t) = 2t$. □

6.5.2 Construction of the kernel

For constructing a linear kernel, we use the following observation, also noted in [63]. Suppose that (G, k) is a No-instance of r-Scattered Set. Then, if for $1 \leqslant i \leqslant k$ we greedily choose a vertex v_i in $G - \bigcup_{j<i} N_{2r}[v_j]$, the graph $G - \bigcup_{1\leqslant i\leqslant k} N_{2r}[v_i]$ is empty. Thus, $\{v_1, \ldots, v_k\}$ is a $2r$-dominating set.

Lemma 6.7 (Fomin et al. [63]) *If (G, k) is a* No*-instance of the r-*Scattered Set *problem, then (G, k) is a* Yes*-instance of the $2r$-*Dominating Set *problem.*

We are ready to present the linear kernel for r-Scattered Set on apex-minor-free graphs.

Theorem 6.4 *Let $r \geqslant 1$ be an integer, let H be an h-vertex apex graph, and let $r\mathrm{SS}_H$ be the restriction of the r-*Scattered Set *problem to input graphs which exclude H as a minor. Then $r\mathrm{SS}_H$ admits a constructive linear kernel of size at most $f(r, h) \cdot k$, where f is an explicit function depending only on r and h, defined in Equation (6.13) below.*

Proof: Given an instance (G, k) of $r\mathrm{SS}_H$, we run on it the algorithm given by Proposition 6.3 for the r'-Dominating Set problem with $r' := 2r$. If the algorithm is not able to find a set X of the claimed size, then by Lemma 6.7 we can conclude that $(G, k) \in r\mathrm{SS}_H$. Otherwise, we use again the set X as input to the algorithm given by Theorem 6.2, which outputs in linear time a $(r^2 \cdot 2^{\mathcal{O}(h \log h)} \cdot (f_c(h))^3 \cdot k, \mathcal{O}(r \cdot (f_c(h))^2))$-protrusion decomposition. We now consider the encoder $\mathcal{E}_{r\mathrm{IS}} = (\mathcal{C}_{r\mathrm{IS}}, L_{\mathcal{C}_{r\mathrm{IS}}})$ defined in Section 6.5.1. By Lemma 6.6, $\mathcal{E}_{r\mathrm{IS}}$ is an rIS-encoder and $\sim^*_{\mathcal{E}_{r\mathrm{IS}},g,t,\mathcal{G}}$ is DP-friendly, where $\mathcal{G}$ is the class of H-minor-free graphs and $g(t) = 2t$, and furthermore by Equation (6.12) it satisfies $s_{\mathcal{E}_{r\mathrm{IS}}}(t) \leqslant (r+2)^{t(t+1)}$. Therefore, we are again in position to apply Corollary 6.1 and obtain a linear kernel for $r\mathrm{SS}_H$ of size at most

$$r^2 \cdot 2^{\mathcal{O}(h \log h)} \cdot (f_c(h))^3 \cdot b\left(\mathcal{E}_{r\mathrm{IS}}, g, \mathcal{O}(r \cdot (f_c(h))^2), \mathcal{G}\right) \cdot k \ , \tag{6.13}$$

where $b\left(\mathcal{E}_{r\mathrm{IS}}, g, \mathcal{O}(r \cdot (f_c(h))^2), \mathcal{G}\right)$ is the function defined in Lemma 6.3. □

6.6 An explicit linear kernel for Planar-$\mathcal{F}$-Deletion

Let $\mathcal{F}$ be a finite set of graphs. We recall the $\mathcal{F}$-Deletion problem, defined as follows.

$\mathcal{F}$-Deletion	
Instance:	A graph G and a non-negative integer k.
Parameter:	The integer k.
Question:	Does G have a set $S \subseteq V(G)$ such that $\|S\| \leqslant k$ and $G - S$ is F-minor-free for every $F \in \mathcal{F}$?

When all the graphs in $\mathcal{F}$ are connected, the corresponding problem is called Connected-$\mathcal{F}$-Deletion, and when $\mathcal{F}$ contains at least one planar graph, we call it Planar-$\mathcal{F}$-Deletion. When both conditions are satisfied, the problem is called Connected-Planar-$\mathcal{F}$-Deletion. Note that Connected-Planar-$\mathcal{F}$-Deletion encompasses, in particular, Vertex Cover and Feedback Vertex Set.

Our encoder for the $\mathcal{F}$-Deletion problem uses the dynamic programming machinery developed by Adler et al. [J1], and it is described in Section 6.6.1. The properties of this encoder also guarantee that the equivalence relation $\sim_{\mathcal{G},t}$ has finite index (see the last paragraph of Section 6.3.3). We prove that this encoder is indeed an $\mathcal{F}$-Deletion-encoder and that the corresponding equivalence relation is DP-friendly, under the constraint that all the graphs in $\mathcal{F}$ are *connected*. Interestingly, this phenomenon concerning the connectivity seems to be in strong connection with the fact that the $\mathcal{F}$-Deletion problem has FII if all the graphs in $\mathcal{F}$ are connected [63, 137], but for some families $\mathcal{F}$ containing disconnected graphs, $\mathcal{F}$-Deletion has not FII (see [J15] for an example of such a family).

We then obtain a linear kernel for the problem using two different approaches. The first one, described in Section 6.6.1, follows the same scheme as the one used in the previous sections (Sections 6.4 and 6.5), that is, we first find a treewidth-modulator X in polynomial time, and then we use this set X as input to the algorithm of Theorem 6.2 to find a linear protrusion decomposition of the input graph. In order to find the treewidth-modulator X, we need that the input graph G excludes a fixed graph H as a minor.

With our second approach, which can be found in Section 6.6.3, we obtain a linear kernel on the larger class of graphs that exclude a fixed graph H as a *topological* minor. We provide two variants of this second approach. One possibility is to use the randomized constant-factor approximation for Planar-$\mathcal{F}$-Deletion by Fomin et al. [137] as treewidth-modulator, which yields a randomized linear kernel (in the sense that the kernelization algorithm is randomized, and the expected size of the kernel is linear) that can be found in uniform polynomial time. The second possibility consists in arguing just about the *existence* of a linear protrusion decomposition in Yes-instances, and then greedily finding large protrusions to be reduced by the protrusion replacer given by Theorem 6.1. This yields a deterministic linear kernel that can be found in time $n^{f(H,\mathcal{F})}$, where f is a function depending on H and $\mathcal{F}$.

6.6.1 The encoder for $\mathcal{F}$-Deletion and the index of $\sim_{\mathcal{G},t}$

In this subsection we define an encoder $\mathcal{E}_{\mathcal{F}\mathrm{D}} = (\mathcal{C}_{\mathcal{F}\mathrm{D}}, L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}})$ for $\mathcal{F}$-Deletion, and along the way we will also prove that when $\mathcal{G}$ is the class of graphs excluding a fixed graph on h vertices as a minor, then the index of the equivalence relation $\sim_{\mathcal{G},t}$ is bounded by $2^{t \log t} \cdot h^t \cdot 2^{h^2}$.

Recall first that a *model* of a graph F in a graph G is a mapping ϕ that assigns to every edge $e \in E(F)$ an edge $\phi(e) \in E(G)$, and to every vertex $v \in V(F)$ a non-empty connected subgraph $\phi(v) \subseteq G$, such that

(i) the graphs $\{\phi(v) \mid v \in V(F)\}$ are mutually vertex-disjoint and the edges $\{\phi(e) \mid e \in E(F)\}$ are pairwise distinct;

(ii) for $e = \{u, v\} \in E(F)$, $\phi(e)$ has one end-vertex in $V(\phi(u))$ and the other in $V(\phi(v))$.

Assume first for simplicity that $\mathcal{F} = \{F\}$ consists of a single connected graph F. Following [J1], we introduce a combinatorial object called *rooted packing*. These objects are originally defined for branch decompositions, but we can directly translate them to tree-decompositions. Loosely speaking, rooted packings capture how "potential models" of F intersect the separators that the algorithm is processing. It is worth mentioning that the notion of rooted packing is related to the notion of *folio* introduced by Robertson and Seymour in [226], but more suited to dynamic programming. See [J1] for more details.

Formally, let $S_F^* \subseteq V(F)$ be a subset of the vertices of the graph F, and let $S_F \subseteq S_F^*$. Given a bag B of a tree-decomposition $(T, \mathcal{X})$ of the input graph G, we define a *rooted packing* of B as a quintuple $\mathbf{rp} = (\mathcal{A}, S_F^*, S_F, \psi, \chi)$, where $\mathcal{A}$ is a (possible empty) collection of mutually disjoint non-empty subsets of B (that is, a *packing* of B), $\psi : \mathcal{A} \to S_F$ is a surjective mapping (called the *rooting*) assigning vertices of S_F to the sets in $\mathcal{A}$, and $\chi : S_F \times S_F \to \{0, 1\}$ is a binary symmetric function between pairs of vertices in S_F.

The intended meaning of a rooted packing $(\mathcal{A}, S_F^*, S_F, \psi, \chi)$ is as follows. In a given separator B, a packing $\mathcal{A}$ represents the intersection of the connected components of the potential model with B. The subsets $S_F^*, S_F \subseteq V(F)$ and the function χ indicate that we are looking in the graph G_B for a potential model of $F[S_F^*]$ containing the edges between vertices in S_F given by the function χ. Namely, the function χ captures which edges of $F[S_F^*]$ have been realized so far in the processed graph. Since we allow the vertex-models intersecting B to be disconnected, we need to keep track of their connected components. The subset $S_F \subseteq S_F^*$ tells us which vertex-models intersect B, and the function ψ associates the sets in $\mathcal{A}$ with the vertices in S_F. We can think of ψ as a coloring that colors the subsets in $\mathcal{A}$ with colors given by the vertices in S_F. Note that several subsets in $\mathcal{A}$ can have the same color $u \in S_F$, which means that the vertex-model of u in G_B is not connected yet, but it may get connected in further steps of the dynamic programming. Again, see [J1] for the details.

It is proved in [J1] that rooted packings allow to carry out dynamic programming in order to determine whether an input graph G contains a graph F as a minor. It is easy to see that the number of distinct rooted packings at a bag B is upper-bounded by $f(t, F) := 2^{t \log t} \cdot r^t \cdot 2^{r^2}$, where $t = \mathsf{tw}(G)$ and $r = |V(F)|$. In particular, this proves that

when $\mathcal{G}$ is the class of graphs excluding a fixed graph H on h vertices as a minor, then the index of the equivalence relation $\sim_{\mathcal{G},t}$ is bounded by $2^{t \log t} \cdot h^t \cdot 2^{h^2}$.

Nevertheless, in order to solve the $\mathcal{F}$-Deletion problem, we need a more complicated data structure. The intuitive reason is that it is inherently more difficult to cover *all* models of a graph F with at most k vertices, rather than just finding one. We define $\mathcal{C}_{\mathcal{F}\mathrm{D}}$ as the function which maps $I \subseteq \{1, \ldots, t\}$ to a subspace of $\{0,1\}^{f(|I|,F)}$. That is, each $\mathcal{C}_{\mathcal{F}\mathrm{D}}$-encoding $R \in \mathcal{C}(I)$ is a vector of $f(|I|, \mathcal{F})$ bits, which when interpreted as the tables of a dynamic programming algorithm at a given bag B such that $\Lambda(G_B) = I$, prescribes which rooted packings exist in the graph G_B once the corresponding vertices of the desired solution to $\mathcal{F}$-Deletion have been removed. More precisely, the language $L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}}$ contains the triples (G, S, R) (recall from Definition 6.6 that here G is a boundaried graph with $\Lambda(G) \subseteq I$, $S \subseteq V(G)$, and $R \in \mathcal{C}(I)$) such that the graph $G - S$ contains precisely the rooted packings prescribed by R (namely, those whose corresponding bit equals 1 in R), and such that the graph $G - (\partial G \cup S)$ does *not* contain F as a minor.

When the family $\mathcal{F} = \{F_1, \ldots, F_\ell\}$ may contain more than one graph, let $f(t, \mathcal{F}) = \sum_{i=1}^{\ell} f(t, F_i)$, and we define $\mathcal{C}_{\mathcal{F}\mathrm{D}}$ as the function which maps $I \subseteq \{1, \ldots t\}$ to a subspace of $\{0,1\}^{f(|I|,\mathcal{F})}$. The language is defined $L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}}$ is defined accordingly, that is, such that the graph $G - S$ contains precisely the rooted packings of F_i prescribed by R, for each $1 \leqslant i \leqslant \ell$, and such that the graph $G - (\partial G \cup S)$ does *not* contain any of the graphs in $\mathcal{F}$ as a minor. By definition of $\mathcal{E}_{\mathcal{F}\mathrm{D}}$, it clearly holds that

$$s_{\mathcal{E}_{\mathcal{F}\mathrm{D}}}(t) \;\leqslant\; 2^{f(t,F_1)} \cdot 2^{f(t,F_2)} \cdots 2^{f(t,F_\ell)} \;=\; 2^{f(t,\mathcal{F})}. \tag{6.14}$$

Assume henceforth that all graphs in the family $\mathcal{F}$ are *connected*. This assumption is crucial because for a connected graph $F \in \mathcal{F}$ and a potential solution S, as the graph $G - (\partial G \cup S)$ does not contain F as a minor, we can assume that the packing $\mathcal{A}$ corresponding to a potential model of F rooted at $\partial G \setminus S$ is *nonempty*. Indeed, as F is connected, a rooted packing which does *not* intersect $\partial G \setminus S$ can never be extended to a (complete) model of F in $G \oplus K$ for any t-boundaried graph K. Therefore, we can directly discard these empty rooted packings. We will use this property in the proof of Lemma 6.8 below. Note that this assumption is not safe if F contains more than one connected component. As mentioned before, this phenomenon seems to be in strong connection with the fact that the $\mathcal{F}$-Deletion problem has FII if all the graphs in $\mathcal{F}$ are connected [63, 137], but for some families $\mathcal{F}$ containing disconnected graphs, $\mathcal{F}$-Deletion has not FII.

Lemma 6.8 *The encoder $\mathcal{E}_{\mathcal{F}\mathrm{D}}$ is a* Connected-$\mathcal{F}$-Deletion*-encoder. Furthermore, if $\mathcal{G}$ is an arbitrary class of graphs and $g(t) = t$, then the equivalence relation $\sim^*_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t,\mathcal{G}}$ is DP-friendly.*

Proof: The fact that $\mathcal{E}_{\mathcal{F}\mathrm{D}} = (\mathcal{C}_{\mathcal{F}\mathrm{D}}, L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}})$ is a Connected-$\mathcal{F}$-Deletion-encoder follows easily from the above discussion, as if G is a 0-boundaried graph, then $\mathcal{C}_{\mathcal{F}\mathrm{D}}(\emptyset)$ consists of a single $\mathcal{C}_{\mathcal{F}\mathrm{D}}$-encoding $R_\emptyset$, and $(G, S, R_\emptyset) \in L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}}$ if and only if the graph $G - S$ contains none of the graphs in $\mathcal{F}$ as a minor. It remains to prove that the equivalence relation $\sim^*_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t,\mathcal{G}}$ is DP-friendly for $g(t) = t$.

The proof is similar to the proofs for r-Dominating Set and r-Scattered Set, so we will omit some details. As in the proof of Lemma 6.5, we start by proving that the

encoder $\mathcal{E}_{\mathcal{F}\mathrm{D}}$ for Connected-$\mathcal{F}$-Deletion is g-confined for the identity function $g(t) = t$. Similarly to the encoder we presented for r-Dominating Set, $\mathcal{E}_{\mathcal{F}\mathrm{D}} = (\mathcal{C}_{\mathcal{F}\mathrm{D}}, L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}})$ has the following monotonicity property. For $R_1, R_2 \in \mathcal{C}_{\mathcal{F}}(I)$ such that $f_G^{\mathcal{C}_{\mathcal{F}\mathrm{D}}}(R_1) < \infty$ and $f_G^{\mathcal{C}_{\mathcal{F}\mathrm{D}}}(R_2) < \infty$,

$$\text{if } R_1^{-1}(0) \subseteq R_2^{-1}(0), \quad \text{then} \quad f_G^{\mathcal{C}_{\mathcal{F}\mathrm{D}}}(R_1) \leqslant f_G^{\mathcal{C}_{\mathcal{F}\mathrm{D}}}(R_2), \tag{6.15}$$

where for $i \in \{1, 2\}$, $R_i^{-1}(0)$ denotes the set of rooted packings whose corresponding bit equals 0 in R_i. Indeed, Equation (6.15) holds because any solution S in G that covers all the rooted packings forbidden by R_2 also covers those forbidden by R_1 (as by hypothesis $R_1^{-1}(0) \subseteq R_2^{-1}(0)$), so it holds that $f_G^{\mathcal{C}_{\mathcal{F}\mathrm{D}}}(R_1) \leqslant f_G^{\mathcal{C}_{\mathcal{F}\mathrm{D}}}(R_2)$.

Let $R_0 = \{0, 0, \ldots, 0\}$ be the $\mathcal{C}_{\mathcal{F}\mathrm{D}}(I)$-encoding will all the bits set to 0. The key observation is that, since each graph in $\mathcal{F}$ is *connected*, by the discussion above the lemma we can assume that each packing $\mathcal{A}$ in a rooted packing is nonempty. This implies that if $R \in \mathcal{C}_{\mathcal{F}\mathrm{D}}(I)$ such that $(G, S, R) \in L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}}$ for some set $S \subseteq V(G)$, then $(G, S \cup \partial G, R_0) \in L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}}$. In other words, any solution S for an arbitrary $\mathcal{C}_{\mathcal{F}\mathrm{D}}$-encoding R can be transformed into a solution for R_0 by adding a set of vertices of size at most $|\partial(G)| \leqslant t$. As by Equation (6.15), for any $\mathcal{C}_{\mathcal{F}\mathrm{D}}$-encoding R with $f_G^{\mathcal{C}_{\mathcal{F}\mathrm{D}}}(R) < \infty$ it holds that $f_G^{\mathcal{C}_{\mathcal{F}\mathrm{D}}}(R) \leqslant f_G^{\mathcal{C}_{\mathcal{F}\mathrm{D}}}(R_0)$, it follows that for any graph G with $\Lambda(G) = I$,

$$\max_{R \in \mathcal{C}^*_{\mathcal{F}\mathrm{D},G}(I)} f_G^{\mathcal{E}_{\mathcal{F}\mathrm{D}}}(R) \;-\; \min_{R \in \mathcal{C}^*_{\mathcal{F}\mathrm{D},G}(I)} f_G^{\mathcal{E}_{\mathcal{F}\mathrm{D}}}(R) \;\leqslant\; t, \quad \text{as we wanted to prove.}$$

Once we have that $\mathcal{E}_{\mathcal{F}\mathrm{D}} = (\mathcal{C}_{\mathcal{F}\mathrm{D}}, L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}})$ is g-confined, the proof goes along the same lines of that of Lemma 6.5. That is, the objective is to show that, in the setting depicted in Figure 6.3, $G \sim^*_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t,\mathcal{G}} G'$ (due to Fact 1) and $\Delta_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t}(G, G') = \Delta_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t}(G_B, G'_B)$. Due to the g-confinement, it suffices to prove that $f_G^{\mathcal{E}_{\mathcal{F}\mathrm{D}}}(R_A) = f_{G'}^{\mathcal{E}_{\mathcal{F}\mathrm{D}}}(R_A) + \Delta_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t}(G_B, G'_B)$ for all $R_A \in \mathcal{C}_{\mathcal{F}\mathrm{D}}(\Lambda(G))$. Since $G_B \sim^*_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t} G'_B$, the definition of $\mathcal{E}_{\mathcal{F}\mathrm{D}}$ it implies that the graphs G_B and G'_B contain exactly the same set of rooted packings, so their behavior with respect to H (see Figure 6.3) in terms of the existence of models of graphs in $\mathcal{F}$ is exactly the same. For more details, it is proved in [J1] that using the encoder $\mathcal{E}_{\mathcal{F}\mathrm{D}} = (\mathcal{C}_{\mathcal{F}\mathrm{D}}, L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}})$, the tables of a given bag in a tree- or branch-decomposition can indeed be computed from the tables of their children. Therefore, we have that $G \sim^*_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t,\mathcal{G}} G'$. Finally, the fact that $f_G^{\mathcal{E}_{\mathcal{F}\mathrm{D}}}(R_A) = f_{G'}^{\mathcal{E}_{\mathcal{F}\mathrm{D}}}(R_A) + \Delta_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t}(G_B, G'_B)$ can be easily proved by noting that any set $S \in V(G)$ satisfying R_A can be transformed into a set $S' \in V(G')$ satisfying R_A such that $|S'| \leqslant |S| - \Delta_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t}(G_B, G'_B)$ (by just replacing $S \cap V(G_B)$ with the corresponding set of vertices in $V(G'_B)$, using that $G_B \sim^*_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t} G'_B$), and vice versa. □

6.6.2 Construction of the kernel on H-minor-free graphs

The objective of this subsection is to prove the following theorem.

Theorem 6.5 *Let $\mathcal{F}$ be a finite set of connected graphs containing at least one r-vertex planar graph F, let H be an h-vertex graph, and let* CP$\mathcal{F}$D$_H$ *be the restriction of the* Connected-Planar-$\mathcal{F}$-Deletion *problem to input graphs which exclude H as a minor. Then* CP$\mathcal{F}$D$_H$ *admits a constructive linear kernel of size at most $f(r, h) \cdot k$, where f is an explicit function depending only on r and h, defined in Equation (6.16) below.*

Similarly to the strategy that we presented in Section 6.4.2 for r-Dominating Set, in order to construct a linear kernel for Connected-Planar-$\mathcal{F}$-Deletion when the input graph excludes a fixed graph H as a minor, we use the fact that this problem satisfies the minor-bidimensionality and separability conditions required in order to apply the results of Fomin et al. [140]. Namely, in the following proposition we specify the result in [140, Lemma 3.3] for the case of Planar-$\mathcal{F}$-Deletion while making visible the dependance on r and the size h of the excluded graph. Again, the polynomial-time algorithm follows from Fomin et al. [139, Lemma 3.2].

Proposition 6.4 *Let $\mathcal{F}$ be a finite set of graphs containing at least one r-vertex planar graph F, let H be an h-vertex graph, and let $\mathrm{P}\mathcal{F}\mathrm{D}_H$ be the restriction of the* Planar-$\mathcal{F}$-Deletion *problem to input graphs which exclude H as a minor. If $(G,k) \in \mathrm{P}\mathcal{F}\mathrm{D}_H$, then there exists a set $X \subseteq V(G)$ such that $|X| = \mathcal{O}(r \cdot f_m(h) \cdot k)$ and $\mathsf{tw}(G-X) = \mathcal{O}(r \cdot (f_m(h))^2)$, where f_m is the function in Proposition 6.1. Moreover, given an instance (G,k) of $\mathrm{P}\mathcal{F}\mathrm{D}_H$, there is a polynomial-time algorithm that either finds such a set X or correctly reports that (G,k) is a* No*-instance.*

We are ready to present a linear kernel for Connected-Planar-$\mathcal{F}$-Deletion when the input graph excludes a fixed graph H as a minor.

Proof of Theorem 6.5: The proof is very similar to the one of Theorem 6.3. Given an instance (G,k), we run the polynomial-time algorithm given by Proposition 6.4 to either conclude that (G,k) is a No-instance or to find a set $X \subseteq V(G)$ such that $|X| = \mathcal{O}(r \cdot f_m(h) \cdot k)$ and $\mathsf{tw}(G-X) = \mathcal{O}(r \cdot (f_m(h))^2)$. In the latter case, we use the set X as input to the algorithm given by Theorem 6.2, which outputs in linear time a $(r^2 \cdot 2^{\mathcal{O}(h \log h)} \cdot (f_m(h))^3 \cdot k, \mathcal{O}(r \cdot (f_m(h))^2))$-protrusion decomposition of G. We now consider the encoder $\mathcal{E}_{\mathcal{F}\mathrm{D}} = (\mathcal{C}_{\mathcal{F}\mathrm{D}}, L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}})$ defined in Section 6.6.1. By Lemma 6.8, $\mathcal{E}_{\mathcal{F}\mathrm{D}}$ is a $\mathrm{CP}\mathcal{F}\mathrm{D}_H$-encoder and $\sim^*_{\mathcal{E}_{\mathcal{F}\mathrm{D}},g,t,\mathcal{G}}$ is DP-friendly, where $g(t) = t$ and $\mathcal{G}$ is the class of H-minor-free graphs. An upper bound on $s_{\mathcal{E}_{\mathcal{F}\mathrm{D}}}(t)$ is given in Equation (6.14). Therefore, we are in position to apply Corollary 6.1 and obtain a linear kernel for $\mathrm{CP}\mathcal{F}\mathrm{D}_H$ of size at most

$$r^2 \cdot 2^{\mathcal{O}(h \log h)} \cdot (f_m(h))^3 \cdot b\left(\mathcal{E}_{\mathcal{F}\mathrm{D}}, g, \mathcal{O}(r \cdot (f_m(h))^2), \mathcal{G}\right) \cdot k \ , \tag{6.16}$$

where $b\left(\mathcal{E}_{\mathcal{F}\mathrm{D}}, g, \mathcal{O}(r \cdot (f_m(h))^2), \mathcal{G}\right)$ is the function defined in Lemma 6.3. □

6.6.3 Linear kernels on H-topological-minor-free graphs

In this subsection we explain how to obtain linear kernels for Planar-$\mathcal{F}$-Deletion on graphs excluding a topological minor. We first describe a uniform randomized kernel and then a nonuniform deterministic one. We would like to note that in the case that $\mathcal{G}$ is the class of graphs excluding a fixed h-vertex graph H as a topological minor, by using a slight variation of the rooted packings described in Section 6.6.1 it can be proved, using standard dynamic techniques, that the index of the equivalence relation $\sim_{\mathcal{G},t}$ is also upper-bounded by $2^{t \log t} \cdot h^t \cdot 2^{h^2}$.

Before presenting the uniform randomized kernel, we need the following two results.

Theorem 6.6 (Fomin et al. [137]) *The optimization version of the* Planar-$\mathcal{F}$-Deletion *problem admits a randomized constant-factor approximation.*

Theorem 6.7 (Leaf and Seymour [193]) *For every simple planar graph F on r vertices, every F-minor-free graph G satisfies* $\mathsf{tw}(G) \leqslant 2^{15r+8r\log r}$.

Theorem 6.8 *Let $\mathcal{F}$ be a finite set of connected graphs containing at least one r-vertex planar graph F, let H be an h-vertex graph, and let* CP$\mathcal{F}$D$_{H\text{-top}}$ *be the restriction of the* Connected-Planar-$\mathcal{F}$-Deletion *problem to input graphs which exclude H as a topological minor. Then* CP$\mathcal{F}$D$_{H\text{-top}}$ *admits a linear randomized kernel of size at most $f(r,h)\cdot k$, where f is an explicit function depending only on r and h, defined in Equation (6.17) below.*

Proof: Given an instance (G,k) of CP$\mathcal{F}$D$_{H\text{-top}}$, we first run the randomized polynomial-time approximation algorithm given by Theorem 6.6, which achieves an expected constant ratio $c_{\mathcal{F}}$. If we obtain a solution $X \subseteq V(G)$ such that $|X| > c_{\mathcal{F}}\cdot k$, we declare that (G,k) is a No-instance. Otherwise, if $|X| \leqslant c_{\mathcal{F}}\cdot k$, we use the set X as input to the algorithm given by Theorem 6.2. As by Theorem 6.7 we have that $\mathsf{tw}(G-X) \leqslant 2^{15r+8r\log r}$, we obtain in this way a $\left(c_{\mathcal{F}}\cdot 40h^2\cdot 2^{15r+8r\log r+5h\log h}\cdot k, 2^{15r+8r\log r+1}+h\right)$-protrusion decomposition of G. We now consider again the encoder $\mathcal{E}_{\mathcal{F}\mathrm{D}} = (\mathcal{C}_{\mathcal{F}\mathrm{D}}, L_{\mathcal{C}_{\mathcal{F}\mathrm{D}}})$ defined in Section 6.6.1, and by Corollary 6.1 we obtain a kernel of size at most

$$\left(1+b\left(\mathcal{E}_{\mathcal{F}\mathrm{D}}, g, 2^{15r+8r\log r+1}+h, \mathcal{G}\right)\right)\cdot\left(c_{\mathcal{F}}\cdot 40h^2\cdot 2^{15r+8r\log r+5h\log h}\right)\cdot k\,, \tag{6.17}$$

where $b\left(\mathcal{E}_{\mathcal{F}\mathrm{D}}, g, 2^{15r+8r\log r+1}+h, \mathcal{G}\right)$ is the function defined in Lemma 6.3 and $\mathcal{G}$ is the class of H-topological-minor-free graphs. □

We finally present a deterministic kernel, whose drawback is that the running time is nonuniform on $\mathcal{F}$ and H.

Theorem 6.9 *Let $\mathcal{F}$ be a finite set of connected graphs containing at least one r-vertex planar graph F, let H be an h-vertex graph, and let* CP$\mathcal{F}$D$_{H\text{-top}}$ *be the restriction of the* Connected-Planar-$\mathcal{F}$-Deletion *problem to input graphs which exclude H as a topological minor. Then* CP$\mathcal{F}$D$_{H\text{-top}}$ *admits a linear kernel of size at most $f(r,h)\cdot k$, where f is an explicit function depending only on r and h, defined in Equation (6.18) below.*

Proof: The main observation is that if $(G,k)\in$ CP$\mathcal{F}$D$_{H\text{-top}}$, then there exists a set $X\subseteq V(G)$ with $|X|\leqslant k$ such that $G-X$ is $\mathcal{F}$-minor-free. In particular, by Theorem 6.7 it holds that $\mathsf{tw}(G-X)\leqslant 2^{15r+8r\log r}$. Therefore, we know by Theorem 6.2 that if $(G,k)\in$ CP$\mathcal{F}$D$_{H\text{-top}}$, then G admits a $(40\cdot h^2\cdot 2^{15r+8r\log r+5h\log h}\cdot k, 2^{15r+8r\log r+1}+h)$-protrusion decomposition. Nevertheless, we do not have tools to efficiently find such linear decomposition. However, we use that, as observed in [63], a t-protrusion of size more than a prescribed number x in an n-vertex graph can be found in $n^{\mathcal{O}(t)}$ steps, it if exists. Our kernelization algorithm proceeds as follows. We try to find a $(2^{15r+8r\log r+1}+h)$-protrusion

Y of size strictly larger than $x := b(\mathcal{E}_{\mathcal{F}\mathrm{D}}, g, 2^{15r+8r\log r+1}+h, \mathcal{G})$, where $\mathcal{E}_{\mathcal{F}\mathrm{D}}$ is the encoder for $\mathcal{F}$-Deletion described in Section 6.6.1, $b\left(\mathcal{E}_{\mathcal{F}\mathrm{D}}, g, 2^{15r+8r\log r+1}+h, \mathcal{G}\right)$ is the function defined in Lemma 6.3, and $\mathcal{G}$ is the class of H-topological-minor-free graphs. If we succeed, we apply the protrusion replacement algorithm given by Theorem 6.1 and replace Y with another t-boundaried graph Y' such that $|Y'| \leqslant b\left(\mathcal{E}_{\mathcal{F}\mathrm{D}}, g, 2^{15r+8r\log r+1}+h, \mathcal{G}\right)$. The algorithm continues as far as we are able to find such large protrusion. At the end of this procedure, we either obtain an equivalent instance of size at most

$$b\left(\mathcal{E}_{\mathcal{F}\mathrm{D}}, g, 2^{15r+8r\log r+1}+h, \mathcal{G}\right) \cdot 40 \cdot h^2 \cdot 2^{15r+8r\log r+5h\log h} \cdot k\ , \tag{6.18}$$

or otherwise we can correctly declare that (G,k) is a No-instance. This kernelization algorithm runs in time $n^{\mathcal{O}(2^{15r+8r\log r+1}+h)}$. □

To conclude this section, we would like to note that the recent results of Chekuri and Chuzhoy [81] show that in Theorem 6.7, the inequality $\mathsf{tw}(G) \leqslant 2^{15r+8r\log r}$ can be replaced with $\mathsf{tw}(G) = r^{\mathcal{O}(1)}$. This directly implies that in Equations (6.17) and (6.18), as well as in the running time of the algorithm of Theorem 6.9, the term $2^{15r+8r\log r}$ can be replaced with $r^{\mathcal{O}(1)}$. Nevertheless, we decided to keep the current bounds in order to be able to give explicit constants.

6.7 Concluding remarks

The methodology for performing explicit protrusion replacement via dynamic programming that we have presented is quite general, and it could also be used to obtain polynomial kernels (not necessarily linear). We have restricted ourselves to vertex-certifiable problems, but is seems plausible that our approach could be also extended to edge-certifiable problems or to problems on directed graphs. In this direction, we provided in [S37] an extension of our framework to so-called *packing-certifiable* problems, which, in a nutshell, are problems whose solutions can be certified by a collection of subgraphs.

We have presented in Section 6.6 a linear kernel for Connected-Planar-$\mathcal{F}$-Deletion when the input graph excludes a fixed graph H as a (topological) minor. The Planar-$\mathcal{F}$-Deletion problem is known to admit a polynomial kernel on general graphs [137]. Nevertheless, this kernel has size $\mathcal{O}(k^c)$, where c is a constant depending on $\mathcal{F}$ that is upper-bounded by $2^{2^{r^{10}}}$, where r is the size of a largest graph in $\mathcal{F}$. This dependance is justified by the recent results of Giannopoulou et al. [156], ruling out the existence of a *uniform* polynomial kernel (that is, a polynomial kernel whose degree does not depend on the family $\mathcal{F}$) for Planar-$\mathcal{F}$-Deletion on general graphs.

As mentioned above, our linear kernel for Planar-$\mathcal{F}$-Deletion requires that all graphs in the family $\mathcal{F}$ are *connected*. It would be interesting to get rid of this assumption, as we did in Chapter 5 to deduce the *existence* of a linear kernel. On the other hand, in the linear kernel for Connected-Planar-$\mathcal{F}$-Deletion on H-topological-minor-free graphs given in Theorem 6.8, the randomization appears because we use the randomized constant-factor approximation for Planar-$\mathcal{F}$-Deletion on general graphs [137], but for

our kernel to be deterministic, it would be enough with a constant-factor approximation on H-topological-minor-free graphs, which is not known.

All the applications examined in this chapter concerned parameterized problems tuned by a secondary parameter, i.e., r for the case of r-Dominating Set and r-Scattered Set and the size of the graphs in $\mathcal{F}$ for the case of $\mathcal{F}$-Deletion. In all kernels derived for these problems, the dependance on this secondary parameter is triple-exponential, while the dependance on the excluded graph H involves the functions f_m and f_c defined in Section 6.2. Two questions arise:

- Extend our results to larger graph classes and more general problems, other than the ones considered in [S37]. Also, improve the dependance of the size of the kernels on the "meta-parameters" associated with the problems (that is, r, $\mathcal{F}$, and H). Probably the recent results of Chekuri and Chuzhoy [81] can be used in this direction. Moreover, provide refinements of this framework that can lead to reasonable explicit bounds for the kernels for particular problems.

- Examine to what extent this exponential dependance is unavoidable under some assumptions based on automata theory or (parameterized) complexity theory. We suspect that the unification between dynamic programming and kernelization that we propose in this chapter might offer a common understanding of the lower bounds in the running time of dynamic programming algorithms for certain problems (see [196, 197]) and the sizes of their corresponding kernels (see for instance [59, 61, 64, 101]). Finally, we refer the reader to [46] for constructibility issues of algebraic graph reduction.

Finally, it is worth mentioning the recent work of Jansen and Wulms [173], which studies the size of the graphs in the set of representatives $\mathcal{R}_t$ that we consider in our approach. Namely, in the framework we presented in this chapter, the size of the graphs in the set $\mathcal{R}_t$ grows at most triple-exponentially with the boundary size t. They complement this bound by proving that each set of planar representatives for Independent Set or Dominating Set contains a graph with $\Omega(2t/\sqrt{4t})$ vertices. Jansen and Wulms [173] also show that the number of equivalence classes for Independent Set on t-boundaried graphs is at most 2^{2^t}, improving over the bound of $(t+1)^{2^t}$ that follows from our Lemma 6.1.

Chapter 7

On the number of labeled graphs of bounded treewidth

Let $T_{n,k}$ be the number of labeled graphs on n vertices and treewidth at most k (equivalently, the number of labeled partial k-trees). In this chapter we show that

$$\left(c\frac{k\ 2^k n}{\log k}\right)^n 2^{-\frac{k(k+3)}{2}} k^{-2k-2} \leqslant T_{n,k} \leqslant \left(k\ 2^k n\right)^n 2^{-\frac{k(k+1)}{2}} k^{-k},$$

for $k > 1$ and some explicit absolute constant $c > 0$. Disregarding lower-order terms, the gap between the lower and upper bound is of order $(\log k)^n$. The upper bound is a direct consequence of the well-known formula for the number of labeled k-trees, while the lower bound is obtained from an explicit construction. It follows from this construction that both bounds also apply to graphs of pathwidth and proper-pathwidth at most k.

Keywords: treewidth; partial k-trees; enumeration; pathwidth; proper-pathwidth.

Contents

7.1 Introduction

Given an integer $k > 0$, a *k-tree* is a graph that can be constructed starting from a $(k+1)$-clique and iteratively adding a vertex connected to k vertices that form a clique. They are natural extensions of trees, which correspond to 1-trees. A formula for the number of labeled k-trees on n vertices was first found by Beineke and Pippert [50], and alternative proofs were given by Moon [209] and Foata [132].

Theorem 7.1 *The number of n-vertex labeled k-trees is equal to*

$$\binom{n}{k}(kn - k^2 + 1)^{n-k-2}. \tag{7.1}$$

A *partial k-tree* is a subgraph of a k-tree. For integers n, k with $0 < k + 1 \leqslant n$, let $T_{n,k}$ denote the number of n-vertex labeled partial k-trees. While the number of n-vertex labeled k-trees is given by Theorem 7.1, it appears that very little is known about $T_{n,k}$. Indeed, to the best of our knowledge, only the cases $k = 1$ (forests) and $k = 2$ (series-parallel graphs) have been studied. The number of n-vertex labeled forests is asymptotically $T_{n,1} \sim \sqrt{e}n^{n-2}$ [231], and the number of n-vertex labeled series-parallel graphs is asymptotically $T_{n,2} \sim g \cdot n^{-5/2}\gamma^n n!$ for some explicit constants g and $\gamma \approx 9.07$ [54].

Partial k-trees are exactly the graphs of *treewidth* at most k; see Section 3.1.3 for the definition of treewidth and pathwidth. The following lemma is well-known and a proof can be found, for instance, in [184].

Lemma 7.1 *A graph has treewidth at most k if and only if it is a partial k-tree.*

In this chapter we are interested in counting n-vertex labeled graphs that have treewidth at most k. By Lemma 7.1, this number is equal to $T_{n,k}$, and actually our approach relies heavily on the definition of partial k-trees.

An easy upper bound on $T_{n,k}$ is obtained as follows. Since every partial k-tree is a subgraph of a k-tree, and a k-tree has exactly $kn - k(k+1)/2$ edges, Theorem 7.1 gives

$$T_{n,k} \leqslant 2^{kn-\frac{k(k+1)}{2}}\binom{n}{k}(kn - k^2 + 1)^{n-k-2}. \tag{7.2}$$

Simple calculations yield, disregarding lower-order terms, that

$$T_{n,k} \leqslant (k2^k n)^n 2^{-\frac{k(k+1)}{2}} k^{-k} \leqslant (k2^k n)^n. \tag{7.3}$$

We can provide a lower bound with the following construction. Starting from an $(n-k+1)$-vertex forest, we add $k-1$ *apices*, that is, $k-1$ vertices with an arbitrary neighborhood in the forest. Every graph created in this way has exactly n vertices and is of treewidth at most k, since adding an apex increases treewidth by at most one. The number of labeled forests on $n-k+1$ vertices is at least the number of trees on $n-k+1$ vertices, which is well-known to be $(n-k+1)^{n-k-1}$. Since each apex can be connected to the ground forest in 2^{n-k+1} different ways, we obtain

$$T_{n,k} \geqslant (n-k+1)^{n-k-1}2^{(k-1)(n-k+1)}. \tag{7.4}$$

If we assume that n/k tends to infinity then asymptotically

$$T_{n,k} \geqslant \left(2^{k-1}n\right)^{n-o(1)}. \tag{7.5}$$

We conclude that $T_{n,k}$ is essentially between $(2^k n)^n$ and $(k2^k n)^n$. These bounds differ by a factor k^n. For constant k this does not matter much since (except when $k = 1, 2$) we do not have a precise estimate on $T_{n,k}$. However, when k goes to infinity, the gap k^n is quite significant. Our main result of this chapter considerably reduces the gap.

Theorem 7.2 *For integers n, k with $k \geqslant 2$ and $k+1 \leqslant n$, the number $T_{n,k}$ of n-vertex labeled graphs with treewidth at most k satisfies*

$$T_{n,k} \;\geqslant\; \left(\frac{1}{128e} \cdot \frac{k2^k n}{\log k}\right)^n 2^{-\frac{k(k+3)}{2}} k^{-2k-2}. \tag{7.6}$$

It follows that $T_{n,k}$ is asymptotically between $\left(\frac{k}{\log k} 2^k n\right)^n$ and $(k2^k n)^n$ when n and k grow. Thus the gap is now of order $(\log k)^n$ instead of k^n.

In order to prove Theorem 7.2, we present in Section 7.2 an algorithmic construction of a family of n-vertex labeled partial k-trees, which is inspired by the definition of k-trees. When exhibiting such a construction toward a lower bound, one has to play with the trade-off of, on the one hand, constructing as many graphs as possible and, on the other hand, being able to bound the number of duplicates; we perform this analysis in Section 7.3. Namely, we first count the number of elements created by the construction, and then we bound the number of times that the same element may have been created. We conclude in Section 7.4 with some remarks and a discussion of further research.

7.2 The construction

Let n and k be fixed positive integers with $0 < k \leqslant n-1$. In this section we construct a set $\mathcal{R}_{n,k}$ of n-vertex labeled partial k-trees. We let $R_{n,k} = |\mathcal{R}_{n,k}|$. In Section 7.2.1 we introduce some notation and definitions used in the construction, in Section 7.2.2 we describe the construction, and in Section 7.2.3 we prove that the treewidth of the graphs generated this way is indeed at most k. In fact, we prove a stronger property, namely that the graphs we construct have *proper-pathwidth* at most k, where the proper-pathwidth, defined later, is a graph invariant that is at least the pathwidth, which is at least the treewidth.

7.2.1 Notation and definitions

For the construction, we use a *labeling function* σ defined by a permutation of $\{1, \ldots, n\}$ with the constraint that $\sigma(1) = 1$. Inspired by the definition of k-trees, we will introduce vertices $\{v_1, v_2, \ldots, v_n\}$ one by one following the order $\sigma(1), \sigma(2), \ldots, \sigma(n)$ given by σ. If $i, j \in \{1, \ldots, n\}$, then i is called the *index* of $v_{\sigma(i)}$, the vertex $v_{\sigma(i)}$ is the i-th introduced vertex and, if $j < i$, the vertex $v_{\sigma(j)}$ is said to be *to the left* of $v_{\sigma(i)}$.

In order to build explicitly a class of partial k-trees, for every $i \geqslant k+1$ we define:

1. A set $A_i \subseteq \{j \mid j < i\}$ of *active* vertices, corresponding to the clique to which a new vertex can be connected in the definition of k-trees, such that $|A_i| = k$.

2. A vertex $a_i \in A_i$, called the *anchor*, whose role will be described in the next paragraph.

3. An element $f(i) \in A_i$, called the *frozen* vertex, which corresponds to a vertex that will not be active anymore.
4. A set $N(i) \subseteq A_i$, which corresponds to the indices of the neighbors of $v_{\sigma(i)}$ to the left.

The construction works with *blocks* of size s, for some integer s depending of n and k, to be specified later. Namely, we insert the vertices by consecutive blocks of size s, with the property that all vertices of the same block share the same anchor and are adjacent to it.

In the description of the construction, we use the term *choose* for the elements for which there are several choices, which will allow us to provide a lower bound on the number of elements in $\mathcal{R}_{n,k}$. This will be the case for the functions σ, f, and N. As will become clear later (see Section 7.3), once σ, f, and N are fixed, all the other elements of the construction are uniquely defined.

For every index $i \geqslant k+2$, we impose that

$$|N(i)| \geqslant \frac{k+1}{2},$$

in order to have simultaneously enough choices for $N(i)$ and enough choices for the frozen vertex $f(i)$, which will be chosen among the vertices in $N(i-1)$. On the other hand, as will become clear later, the role of the anchor vertices is to determine uniquely the vertices belonging to "its" block. To this end, when a new block starts, its anchor is defined as the smallest currently active vertex.

7.2.2 Description of the construction

We say that a triple (σ, f, N), with σ a permutation of $\{1, \dots, n\}$, $f : \{k+2, \dots, n\} \to \{1, \dots, n\}$, and $N : \{2, \dots, n\} \to 2^{\{1,\dots,n\}}$, is *constructible* if it is one of the possible outputs of the following algorithm:

Choose σ, a permutation of $\{1, \dots, n\}$ such that $\sigma(1) = 1$.
for *i=2* **to** k **do**
  Choose $N(i) \subseteq \{j \mid j < i\}$, such that $1 \in N(i)$.
for *i=k+1* **do**
  Define $A_{k+1} = \{j \mid j < k+1\}$.
  Define $a_{k+1} = 1$.
  Choose $N(k+1) \subseteq \{j \mid j < i\}$, such that $1 \in N(k+1)$.
for *i=k+2* **to** n **do**
  if $i \equiv k+2 \pmod{s}$ **then**
    Define $f(i) = a_{i-1}$.
    Define $A_i = (A_{i-1} \setminus \{f(i)\}) \cup \{i-1\}$.
    Define $a_i = \min A_i$.
    Choose $N(i) \subseteq A_i$ such that $a_i \in N(i)$ and $|N(i)| \geqslant \frac{k+1}{2}$; cf. Figure 7.1.
  else
    Choose $f(i) \in (A_{i-1} \setminus \{a_{i-1}\}) \cap N(i-1)$.
    Define $A_i = (A_{i-1} \setminus \{f(i)\}) \cup \{i-1\}$.
    Define $a_i = a_{i-1}$.
    Choose $N(i) \subseteq A_i$ such that $a_i \in N(i)$ and $|N(i)| \geqslant \frac{k+1}{2}$; cf. Figure 7.2.

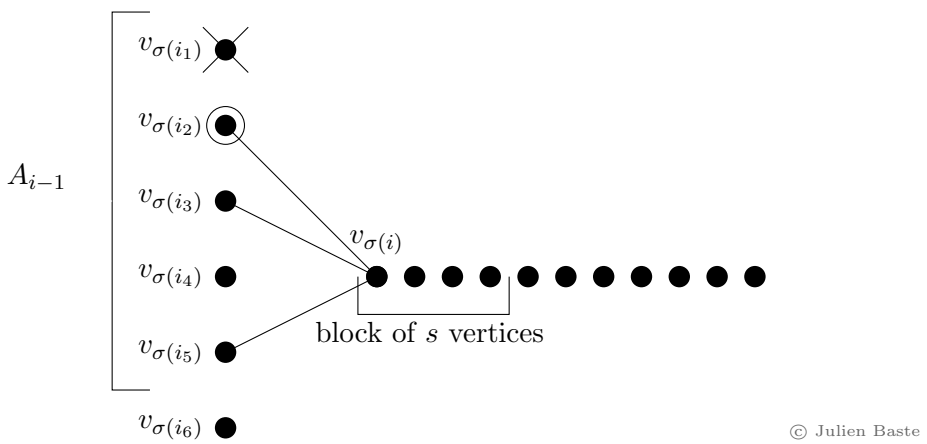

Figure 7.1: Introduction of $v_{\sigma(i)}$ with $k+2 \leqslant i \leqslant n$ and $i \equiv k+2 \pmod{s}$, $s = 4$, and $k = 5$. We assume that $i_1 < i_2 < i_3 < i_4 < i_5 < i_6 < i$, and note that $i_5 = i-2$ and $i_6 = i-1$. We have defined $f(i) = v_{\sigma(i_1)}$ and $a_i = v_{\sigma(i_2)}$. The frozen vertex $f(i)$ is marked with a cross, and the anchor a_i is marked with a circle. We choose $N(i) = \{i_2, i_3, i_5\}$.

Let (σ, f, N) be a constructible triple. We define the graph $G(\sigma, f, N) = (V, E)$ such that $V = \{v_i \mid i \in \{1, \dots, n\}\}$, and $E = \{\{v_{\sigma(i)}, v_{\sigma(j)}\} \mid j \in N(i)\}$. Note that, given (σ, f, N), the graph $G(\sigma, f, N)$ is well-defined. We denote by $\mathcal{R}_{n,k}$ the set of all graphs $G(\sigma, f, N)$ such that (σ, f, N) is constructible.

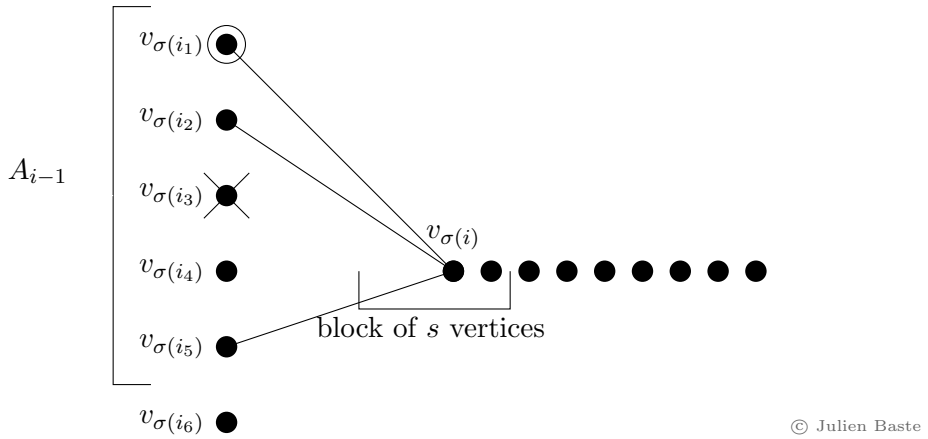

Figure 7.2: Introduction of $v_{\sigma(i)}$ with $k+2 \leqslant i \leqslant n$ and $i \not\equiv k+2 \pmod{s}$, $s = 4$, and $k = 5$. We assume that $i_1 < i_2 < i_3 < i_4 < i_5 < i_6 < i$, and note that $i_5 = i-2$ and $i_6 = i-1$. We have defined $a_i = a_{i-1} = v_{\sigma(i_1)}$. The frozen vertex $f(i)$ is marked with a cross, and the anchor a_i is marked with a circle. We choose $f(i) = v_{\sigma(i_3)}$, assuming $v_{\sigma(i_3)}$ is a neighbor of $v_{\sigma(i_5)}$, and $N(i) = \{i_1, i_2, i_5\}$.

7.2.3 Bounding the treewidth

We start by defining the notion of proper-pathwidth of a graph. This parameter was introduced by Takahashi et al. [232] and its relation with search games has been studied in [233].

Let G be a graph and let $\mathcal{X} = \{X_1, X_2, \ldots, X_r\}$ be a sequence of subsets of $V(G)$. The *width* of $\mathcal{X}$ is $\max_{1 \leqslant i \leqslant r} |X_i| - 1$. $\mathcal{X}$ is called a *proper-path decomposition* of G if the following conditions are satisfied:

1. For any distinct i and j, $X_i \not\subseteq X_j$.
2. $\bigcup_{i=1}^{r} X_i = V(G)$.
3. For every edge $\{u, v\} \in E(G)$, there exists an i such that $u, v \in X_i$.
4. For all a, b, and c with $1 \leqslant a \leqslant b \leqslant c \leqslant r$, $X_a \cap X_c \subseteq X_b$.
5. For all a, b, and c with $1 \leqslant a < b < c \leqslant r$, $|X_a \cap X_c| \leqslant |X_b| - 2$.

The *proper-pathwidth* of G, denoted by $\mathsf{ppw}(G)$, is the minimum width over all proper-path decompositions of G. Note that if $\mathcal{X}$ satisfies only conditions 1-4 above, then $\mathcal{X}$ is a path-decomposition as defined in Section 3.1.3.

From the definitions, for any graph G it clearly holds that

$$\mathsf{ppw}(G) \geqslant \mathsf{pw}(G) \geqslant \mathsf{tw}(G). \tag{7.7}$$

Let us show that any element of $\mathcal{R}_{n,k}$ has proper-pathwidth at most k. Let (σ, f, N) be constructible such that $G(\sigma, f, N) \in \mathcal{R}_{n,k}$ and let A_i be defined as in Section 7.2.2. We

define for every $i \in \{k+1, \ldots, n\}$ the bag $X_i = \{v_{\sigma(j)} \mid j \in A_i \cup \{i\}\}$. The sequence $\mathcal{X} = \{X_{k+1}, X_{k+2}, \ldots, X_n\}$ is a path-decomposition satisfying the five conditions of the above definition, and for every $i \in \{k+1, \ldots, n\}$, $|X_i| = k+1$. It follows that $G(\sigma, f, N)$ has proper-pathwidth at most k, so it also has treewidth at most k, and therefore $G(\sigma, f, N)$ is a partial k-tree by Lemma 7.1.

7.3 Proof of the main result

In this section we analyze our construction and give a lower bound on $R_{n,k}$. We first start by counting the number of constructible triples (σ, f, N) generated by the algorithm, and then we provide an upper bound on the number of duplicates. Finally, we determine the best choice for the parameter s defined in the construction.

7.3.1 Number of constructible triples (σ, f, N)

We proceed to count the number of constructible triples (σ, f, N) created by the algorithm given in Section 7.2.2. As σ is a permutation of $\{1, \ldots, n\}$ with the constraint that $\sigma(1) = 1$, there are $(n-1)!$ distinct possibilities for the choice of σ. The function f can take more than one value only for $k+2 \leqslant i \leqslant n$ and $i \not\equiv k+2 \pmod{s}$. This represents $n-(k+1)-\lceil \frac{n-(k+1)}{s} \rceil$ cases. In each of these cases, there are at least $\frac{k-1}{2}$ distinct possible values for $f(i)$. Thus, we have at least $(\frac{k-1}{2})^{(n-(k+1)-\lceil \frac{n-(k+1)}{s} \rceil)}$ distinct possibilities for the choice of f. For every $i \in \{2, \ldots, k+1\}$, $N(i)$ can be chosen as any subset of $i-1$ vertices containing the fixed vertex $v_{\sigma(1)}$. This yields $\prod_{i=2}^{k+1} 2^{i-2} = 2^{\frac{k(k-1)}{2}}$ ways to define N over $\{2, \ldots, k+1\}$. For $i \geqslant k+2$, $N(i)$ can be chosen as any subset of size at least $\frac{k+1}{2}$ of a set of k elements with one element that is imposed. This results in $\sum_{i=\lceil \frac{k+1}{2} \rceil}^{k} \binom{k-1}{i-1} \geqslant 2^{k-2}$ possible choices for $N(i)$. Thus, we have $2^{\frac{k(k+1)}{2}} 2^{(n-(k+1))(k-2)}$ distinct possibilities to construct N.

By combining everything, we obtain at least

$$(n-1)! \left(\frac{k-1}{2}\right)^{n-(k+1)-\lceil \frac{n-(k+1)}{s} \rceil} 2^{\frac{k(k-1)}{2}} 2^{(n-(k+1))(k-2)} \tag{7.8}$$

distinct possible constructible triples (σ, f, N).

7.3.2 Bounding the number of duplicates

Let H be an element of $\mathcal{R}_{n,k}$. Our objective is to obtain an upper bound on the number of constructible triples (σ, f, N) such that $H = G(\sigma, f, N)$.

Given H, we start by reconstructing σ. Firstly, we know by construction that $\sigma(1) = 1$. Secondly, we know that $f(k+2) = 1$ and so, for every $i > k+1$, $1 \notin A_i$, implying that $1 \notin N(i)$. It follows that the only neighbors of $v_{\sigma(1)}$ are the vertices $\{v_{\sigma(i)} \mid 1 < i \leqslant k+1\}$.

So the set of images under σ of $\{2, \dots, k+1\}$ is uniquely determined. Then we guess the function σ over this set $\{2, \dots, k+1\}$. Overall, this results in $k!$ possible guesses for σ.

Thirdly, assume that we have correctly guessed σ on $\{1, \dots, k+1+ps\}$ for some non-negative integer p with $k+1+ps < n$. Then $a_{k+1+ps+1}$ is the smallest active vertex that is adjacent to at least one element that is still not introduced after step $k+1+ps$. Then the neighbors of $a_{k+1+ps+1}$ over the elements that are not introduced yet after step $k+1+ps$ are the elements whose indices are between $k+1+ps+1$ and $k+1+(p+1)s$, and these vertices constitute the next block of the construction; see Figure 7.3 for an illustration. As before, the set of images by σ of $\{k+1+ps+1, \dots, k+1+(p+1)s\}$ is uniquely determined, and we guess σ over this set. We have at most $s!$ possible such guesses. Fourthly, if $k+1+(p+1)s > n$ (that is, for the last block, which may have size smaller than s), we have $t!$ possible guesses with $t = n-(k+1)-s\lfloor\frac{n-(k+1)}{s}\rfloor$.

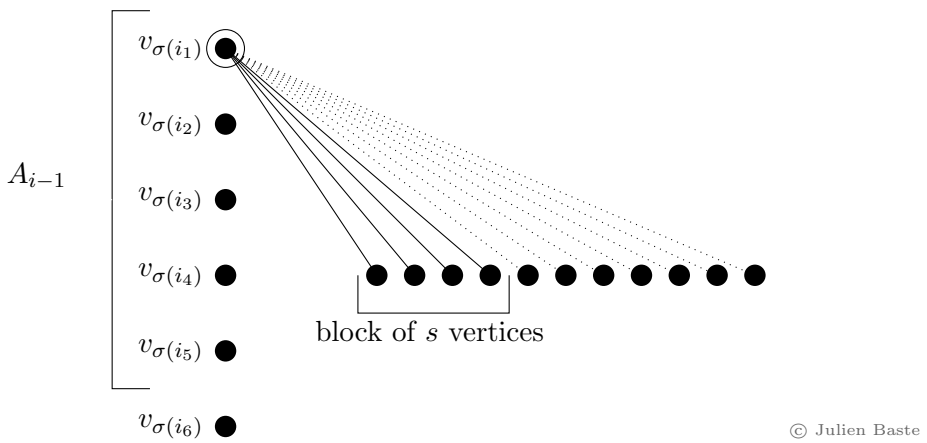

Figure 7.3: The current anchor $v_{\sigma(i_1)}$ is connected to all the s vertices of the current block but will not be connected to any of the remaining non-introduced vertices.

We know that the first, the second, and the fourth cases can occur only once in the construction, and the third case can occur at most $\lfloor\frac{n-(k+1)}{s}\rfloor$ times. Therefore, an upper bound on the number of distinct possible guesses for σ is $k!(s!)^{\lfloor\frac{n-(k+1)}{s}\rfloor}t!$, where $t = n-(k+1)-s\lfloor\frac{n-(k+1)}{s}\rfloor$.

Let us now fix σ. Then the function N is uniquely determined. Indeed, for every $i \in \{1, \dots, n\}$, $N(i)$ corresponds to the neighbors of $v_{\sigma(i)}$ to the left. It remains to bound the number of possible functions f. In order to do this, we define for every $i > 1$, $D_i = \{j \in N(i) \mid \forall j' > i, \{v_{\sigma(j)}, v_{\sigma(j')}\} \notin E(H)\}$. Then, for every $i \geqslant k+2$, by definition of $f(i)$, $f(i) \in D_{i-1}$. Moreover, for $i, j > k+1$ with $i \neq j$, it holds that, by definition of D_i and D_j, $D_i \cap D_j = \emptyset$. Indeed, assume w.l.o.g. that $i < j$, and suppose for contradiction that there exists $a \in D_i \cap D_j$. As $a \in D_j$, it holds that $a \in N(j)$, but as $a \in D_i$, for every $j' > i$, $a \notin N(j')$, hence $a \notin N(j)$, a contradiction.

We obtain that the number of distinct functions f is bounded by $\prod_{i=k+1}^{n} |D_i|$. As $D_i \cap D_j = \emptyset$ for every $i, j \geqslant k+1$ with $i \neq j$ and $D_i \subseteq \{1, \dots, n\}$ for every $i \geqslant k+1$, we have that $\sum_{i=k+1}^{n} |D_i| \leqslant n$. Let $I = \{i \in \{k+1, \dots, n\} \mid |D_i| \geqslant 2\}$, and note that $|I| \leqslant k$.

By the previous discussion, it holds that $\sum_{i\in I}|D_i| \leqslant 2k$. So it follows that, by using Cauchy-Schwarz inequality,

$$\prod_{i=k+1}^{n}|D_i| \;=\; \prod_{i\in I}|D_i| \;\leqslant\; \left(\frac{\sum_{i\in I}|D_i|}{k}\right)^k \;\leqslant\; \left(\frac{2k}{k}\right)^k \;=\; 2^k. \tag{7.9}$$

To conclude, the number of constructible triples that can give rise to H is at most $2^k(s!)^{\lfloor\frac{n-(k+1)}{s}\rfloor}t!$ where $t = n-(k+1)-s\lfloor\frac{n-(k+1)}{s}\rfloor$. Thus, we obtain that

$$R_{n,k} \;\geqslant\; \frac{(n-1)!\left(\frac{k-1}{2}\right)^{n-(k+1)-\lceil\frac{n-(k+1)}{s}\rceil}2^{\frac{k(k-1)}{2}}2^{(n-(k+1))(k-2)}}{2^k k!(s!)^{\lfloor\frac{n-(k+1)}{s}\rfloor}(n-(k+1)-s\lfloor\frac{n-(k+1)}{s}\rfloor)!}. \tag{7.10}$$

For better readability, we bound separately each of the terms on the right-hand side:

- $(n-1)! \geqslant (\frac{n}{e})^n 2^{-n}$, $\quad 2^{\frac{k(k-1)}{2}}2^{(n-(k+1))(k-2)} \geqslant 2^{kn-\frac{k(k+3)}{2}}2^{-2n}$.
- $(k-1)^{(n-(k+1)-\lceil\frac{n-(k+1)}{s}\rceil)} \geqslant 2^{-n}k^{(n-\frac{n}{s}-k-2)}$, since $k \geqslant 2$.
- $2^k k! \leqslant 2^n k^k$, $\;(s!)^{\lfloor\frac{n-(k+1)}{s}\rfloor}(n-(k+1)-s\lfloor\frac{n-(k+1)}{s}\rfloor)! \leqslant s^n$.

Applying these relations to (7.10) gives

$$R_{n,k} \;\geqslant\; \left(\frac{1}{64e}\cdot\frac{k2^k n}{k^{1/s}s}\right)^n 2^{-\frac{k(k+3)}{2}}k^{-2k-2}. \tag{7.11}$$

7.3.3 Choosing the parameter s

We now discuss how to choose the size s of the blocks in the construction. In order to obtain the largest possible lower bound for $R_{n,k}$, we would like to choose s minimizing the factor $k^{1/s}s$ in the denominator of (7.11). To be as general as possible, assume that s is a function $s(n,k)$ that may depend on n and k, and we define $t(n,k) = \frac{s(n,k)}{\log k}$. With this definition, it follows that

$$\log\left(k^{\frac{1}{s(n,k)}}s(n,k)\right) \;=\; \frac{\log k}{s(n,k)} + \log s(n,k) \;=\; \frac{1}{t(n,k)} + \log t(n,k) + \log\log k. \tag{7.12}$$

It is elementary that the minimum of $\frac{1}{t(n,k)} + \log t(n,k)$ is achieved for $t(n,k) = 1$. Thus, we obtain that $s(n,k) = \log k$ is the function that maximizes the lower bound given by Equation (7.11). Therefore, we obtain that

$$R_{n,k} \;\geqslant\; \left(\frac{1}{128e}\cdot\frac{k2^k n}{\log k}\right)^n 2^{-\frac{k(k+3)}{2}}k^{-2k-2}, \tag{7.13}$$

concluding the proof of Theorem 7.2, where we assume that $k \geqslant 2$.

7.4 Concluding remarks

Comparing Equations (7.3) and (7.6), there is still a gap of $(128e \cdot \log k)^n$ in the dominant term of $T_{n,k}$, and closing this gap remains an open problem. The factor $(\log k)^n$ appears because, in our construction, when a new block starts, we force the frozen vertex to be the previous anchor. Therefore, this factor is somehow artificial, and we believe that it could be avoided.

One way to improve the upper bound would be to show that *every* partial k-tree with n vertices and m edges can be extended to at least a large number $\alpha(n, m)$ of k-trees, and then use *double counting*. This is the approach taken in [216] for bounding the number of planar graphs, but so far we have not been able to obtain a significant improvement using this technique.

As mentioned before, our results also apply to other relevant graph parameters such as pathwidth and proper-pathwidth. For both parameters, besides improving the lower bound given by our construction, it may be also possible to improve the upper bound given by Equation (7.3). For proper-pathwidth, a modest improvement can be obtained as follows. It follows easily from the definition of proper-pathwidth that the edge-maximal graphs of proper-pathwidth k, which we call *proper linear k-trees*, can be constructed starting from a $(k+1)$-clique and iteratively adding a vertex v_i connected to a clique K_{v_i} of size k, with the constraints that $v_{i-1} \in K_{v_i}$ and $K_{v_i} \setminus \{v_{i-1}\} \subseteq K_{v_{i-1}}$. From this observation, and taking into account that the order of the first k vertices is not relevant and that there are $2k$ initial cliques giving rise to the same graph, it follows that the number of n-vertex labeled proper linear k-trees is equal to

$$n! k^{n-k-1} \frac{1}{(2k)k!}. \tag{7.14}$$

From this and the fact that a k-tree has $kn - \frac{k(k+1)}{2}$ edges, an easy calculation yields that the number of n-vertex labeled graphs of proper-pathwidth at most k is at most $\left(\frac{k2^k n}{c}\right)^n$, for some absolute constant $c \geqslant 1.88$.

It would be interesting to count graphs of bounded "width" in other cases. For instance, branchwidth seems to be a good candidate, as it is known that, if we denote by $\mathsf{tw}(G)$ the branchwidth of a graph G and $|E(G)| \geqslant 3$, then $\mathsf{tw}(G) \leqslant \mathsf{tw}(G)+1 \leqslant \frac{3}{2}\mathsf{tw}(G)$ [224]. Other relevant graph parameters are cliquewidth, rankwidth, tree-cutwidth, or booleanwidth. For any of these parameters, a first natural step would be to find a "canonical" way to build such graphs, as in the case of partial k-trees.

Our results find algorithmic applications, specially in the area of parameterized complexity [94]. When designing a parameterized algorithm, usually a crucial step is to solve the problem at hand restricted to graphs decomposable along small separators by performing dynamic programming (see [172] for a recent example). For instance, precise bounds on $T_{n,k}$ are useful when dealing with the Treewidth-k Vertex Deletion problem, which has recently attracted significant attention in the area [137, 147, J15]. In this problem, given a graph G and a fixed integer $k > 0$, the objective is to remove as few vertices from G as possible in order to obtain a graph of treewidth at most k. When solving

Treewidth-k Vertex Deletion by dynamic programming, the natural approach is to enumerate, for any partial solution at a given separator of the decomposition, all possible graphs of treewidth at most k that are "rooted" at the separator. In this setting, the value of $T_{n,k}$, as well as an explicit construction to generate such graphs, may be crucial in order to speed-up the running time of the algorithm. Other recent algorithmic applications of knowing the precise number of graphs of bounded treewidth are finding path- or tree-decompositions with minimum number of bags [65] and subgraph embedding problems on sparse graphs [66].

Finally, a challenging open problem is to count the number of *unlabeled* partial k-trees, for which nothing is known except for some results concerning random models [55, 153, 207]. Note that the number of unlabeled k-trees was an open problem for long time, until it was recently solved by Gainer-Dewar [145] (see also [117, 146]).

Chapter 8

Finding a spanning tree with minimum reload cost diameter

In this chapter we study the minimum diameter spanning tree problem under the reload cost model (Diameter-Tree for short) introduced by Wirth and Steffan [241]. In this problem, given an undirected edge-colored graph G, reload costs on a path arise at a node where the path uses consecutive edges of different colors. The objective is to find a spanning tree of G of minimum diameter with respect to the reload costs. We initiate a systematic study of the parameterized complexity of the Diameter-Tree problem by considering the following parameters: the cost of a solution, and the treewidth and the maximum degree Δ of the input graph. We prove that Diameter-Tree is para-NP-hard for any combination of two of these three parameters, and that it is FPT parameterized by the three of them. We also prove that the problem can be solved in polynomial time on cactus graphs. This result is somehow surprising since we prove Diameter-Tree to be NP-hard on graphs of treewidth two, which is best possible as the problem can be trivially solved on forests. When the reload costs satisfy the triangle inequality, Wirth and Steffan [241] proved that the problem can be solved in polynomial time on graphs with $\Delta = 3$, and Galbiati [148] proved that it is NP-hard if $\Delta = 4$. Our results show, in particular, that without the requirement of the triangle inequality, the problem is NP-hard if $\Delta = 3$, which is also best possible. Finally, in the case where the reload costs are polynomially bounded by the size of the input graph, we prove that Diameter-Tree is in XP and W[1]-hard parameterized by the treewidth plus Δ.

Keywords: reload cost problems; minimum diameter spanning tree; parameterized complexity; FPT algorithm; treewidth; dynamic programming.

Contents

8.1 Introduction

Numerous network optimization problems can be modeled by edge-colored graphs. Wirth and Steffan introduced in [241] the concept of *reload cost*, which refers to the cost that arises in an edge-colored graph while traversing a vertex via two consecutive edges of different colors. The value of the reload cost depends on the colors of the traversed edges. Although the reload cost concept has many important applications in telecommunication networks, transportation networks, and energy distribution networks, it has surprisingly received attention only recently.

In heterogeneous communication networks, routing requires switching among different technologies such as cables, fibers, and satellite links. Due to data conversion between incompatible subnetworks, this switching causes high costs, largely outweighing the cost of routing the packets within each subnetwork. The recently popular concept of vertical handover [105], which allows a mobile user to have undisrupted connection during transitioning between different technologies such as 3G (third generation) and wireless local area network (WLAN), constitutes another application area of the reload cost concept. Even within the same technology, switching between different service providers incurs switching costs. Another paradigm that has received significant attention in the wireless networks research community is *cognitive radio networks* (CRN), a.k.a. *dynamic spectrum access networks*. Unlike traditional wireless technologies, CRNs operate across a wide frequency range in the spectrum and frequently requires frequency switching; therefore, the frequency switching cost is indispensable and of paramount importance. Many works in the CRNs literature focused on this frequency switching cost from an application point of view (for instance, see [40,45,48,49,121,159,230]) by analyzing its various aspects such as delay and energy consumption. Operating in a wide range of frequencies is indeed a property of not only CRNs but also other 5G technologies. Hence, applications of the reload cost concept in communication networks continuously increase. In particular, the energy consumption aspect of this switching cost is especially important in the recently active research area of green networks, which aim to tackle the increasing energy consumption of information and communication technologies [53, 79].

The concept of reload cost also finds applications in other networks such as transportation networks and energy distribution networks. For instance, a cargo transportation network uses different means of transportation. The loading and unloading of cargo at junction points is costly and this cost may even outweigh the cost of carrying the cargo from one point to another [148]. In energy distribution networks, reload costs can model the energy losses that occur at the interfaces while transferring energy from one type of carrier to another [148].

Recent works in the literature focused on numerous problems related to the reload cost concept: the minimum reload cost cycle cover problem [150], the problems of finding a path, trail or walk with minimum total reload cost between two given vertices [158], the problem of finding a spanning tree that minimizes the sum of reload costs of all paths between all pairs of vertices [151], various path, tour, and flow problems related to reload costs [43], the minimum changeover cost arborescence problem [149, 160, 162, J13], and problems related to finding a proper edge coloring of the graph so that the total reload cost is minimized [161].

The work in [241], which introduced the concept of reload cost, focused on the following problem, called Minimum Reload Cost Diameter Spanning Tree (Diameter-Tree for short), and which is the one we study in this chapter: given an undirected graph $G = (V, E)$ with a (non necessarily proper) edge-coloring $\chi : E(G) \to X$ and a reload cost function $c : X^2 \to \mathbb{N}_0$, find a spanning tree of G with minimum diameter with respect to the reload costs (see Section 8.2 for the formal definitions).

This problem has important applications in communication networks, since forming a spanning tree is crucial for broadcasting control traffic such as route update messages. For instance, in a multi-hop cognitive radio network where a frequency is assigned to each wireless link depending on availabilities of spectrum bands, delay-aware broadcasting of control traffic necessitates the forming of a spanning tree by taking the delay arising from frequency switching at every node into account. Cognitive nodes send various control information messages to each other over this spanning tree. A spanning tree with minimum reload cost diameter in this setting corresponds to a spanning tree in which the maximum frequency switching delay between any two nodes on the tree is minimized. Since control information is crucial and needs to be sent to all other nodes in a timely manner, ensuring that the maximum delay is minimum is vital in a cognitive radio network.

Wirth and Steffan [241] proved that Diameter-Tree is inapproximable within a factor better than 3 (in particular, it is NP-hard), even on graphs with maximum degree 5. They also provided a polynomial-time exact algorithm for the special case where the maximum degree is 3 and the reload costs satisfy the triangle inequality. Galbiati [148] showed stronger hardness results for this problem, by proving that even on graphs with maximum degree 4, the problem cannot be approximated within a factor better than 2 if the reload costs do not satisfy the triangle inequality, and cannot be approximated within any factor better than 5/3 if the reload costs satisfy the triangle inequality. The complexity of Diameter-Tree (in the general case) on graphs with maximum degree 3 was left open.

Our results. In this chapter we initiate a systematic study of the complexity of the Diameter-Tree problem, with special emphasis on its parameterized complexity for several choices of the parameters. Namely, we consider any combination of the parameters k (the cost of a solution), tw (the treewidth of the input graph), and Δ (the maximum degree of the input graph). We would like to note that these parameters have practical importance in communication networks. Indeed, besides the natural parameter k, whose relevance is clear, many networks that model real-life situations appear to have small treewidth [175, 192]. On the other hand, the degree of a node in a network is related to its number of transceivers, which are costly devices in many different types of networks such as optical networks [187]. For this reason, in practice the maximum degree of a network usually takes small values.

Before elaborating on our results, a summary of them can be found in Table 8.1.

We first prove, by a reduction from 3-Sat, that Diameter-Tree is NP-hard on outerplanar graphs (which have treewidth at most 2) with only one vertex of degree greater than 3, even with three different costs that satisfy the triangle inequality, and $k = 9$. Note that, in the case where the costs satisfy the triangle inequality, having only one vertex of degree greater than 3 is best possible, as if all vertices have degree at most 3, the problem

Problem	Parameterized complexity with parameter				Polynomial cases
	$k + \mathsf{tw}$	$k + \Delta$	$\mathsf{tw} + \Delta$	$k + \mathsf{tw} + \Delta$	
Diameter-Tree	NPh for $k = 9, \mathsf{tw} = 2$ (Thm 8.1)	NPh for $k = 0, \Delta = 3$ (Thm 8.2)	NPh for $\mathsf{tw} = 3, \Delta = 3$ (Thm 8.3)	FPT (Thm 8.5)	in P on cacti (Thm 8.4)
Diameter-Tree with poly costs	✓	✓	XP (Thm 8.5) W[1]-hard (Thm 8.6)	✓	✓

Table 8.1: Summary of our results, where k, tw, Δ denote the cost of the solution, the treewidth, and the maximum degree of the input graph, respectively. NPh stands for NP-hard. The symbol '✓' denotes that the result above still holds for polynomial costs.

can be solved in polynomial time [241]. Note also that the bound on the treewidth is best possible as well, since the problem is trivially solvable on graphs of treewidth 1, i.e., on forests.

Toward investigating the border of tractability of the problem with respect to treewidth, we exhibit a polynomial-time algorithm on a relevant subclass of the graphs of treewidth at most 2: cactus graphs. This algorithm is quite involved and, in a nutshell, processes in a bottom-up manner the *block tree* of the given cactus graph, and uses at each step of the processing an algorithm that solves 2-Sat as a subroutine.

Back to hardness results, we also prove, by a reduction from a restricted version of 3-Sat, that Diameter-Tree is NP-hard on graphs with $\Delta \leqslant 3$, even with only two different costs, $k = 0$, and a bounded number of colors. In particular, this settles the complexity of the problem on graphs with $\Delta \leqslant 3$ in the general case where the triangle inequality is not necessarily satisfied, which had been left open in previous work [148, 241]. Note that $\Delta \leqslant 3$ is best possible, as Diameter-Tree can be easily solved on graphs with $\Delta \leqslant 2$.

As our last NP-hardness reduction, we prove, by a reduction from Partition, that the Diameter-Tree problem is NP-hard on planar graphs with $\mathsf{tw} \leqslant 3$ and $\Delta \leqslant 3$.

The above hardness results imply that the Diameter-Tree problem is para-NP-hard for any combination of two of the three parameters k, tw, and Δ. On the positive side, we show that Diameter-Tree is FPT parameterized by the three of them, by using a (highly nontrivial) dynamic programming algorithm on a tree-decomposition of the input graph.

Since our para-NP-hardness reduction with parameter $\mathsf{tw} + \Delta$ is from Partition, which is a typical example of a *weakly* NP-complete problem [154], a natural question is whether Diameter-Tree, with parameter $\mathsf{tw} + \Delta$, is para-NP-hard, XP, W[1]-hard, or FPT when the reload costs are *polynomially bounded* by the size of the input graph. We manage to answer this question completely: we show that in this case the problem is in XP (hence *not* para-NP-hard) and W[1]-hard parameterized by $\mathsf{tw} + \Delta$. The W[1]-hardness reduction is from the Unary Bin Packing problem parameterized by the number of bins, proved to be W[1]-hard by Jansen et al. [174].

Altogether, our results provide an accurate picture of the (parameterized) complexity of the Diameter-Tree problem.

Organization of the chapter. We start in Section 8.2 with some brief preliminaries about graphs, the Diameter-Tree problem, and nice tree-decompositions that have not been already given in Section 3. In Section 8.3 we provide the para-NP-hardness results. In Section 8.4 we present the polynomial-time algorithm on cactus graphs, and in Section 8.5 we present the FPT algorithm on general graphs parameterized by $k+\mathsf{tw}+\Delta$. In Section 8.6 we focus on the case where the reload costs are polynomially bounded. Finally, we conclude the chapter in Section 8.7.

8.2 Preliminaries

Graphs and sets. Given a graph G and a set $S \subseteq V(G)$, we define $\mathsf{adj}_G(S)$ to be the set of edges of G that intersect S. For a graph G and an edge $e \in E(G)$, we let $G - e = (V(G), E(G) \setminus e)$. Given a graph G and a set $S \subseteq V(G)$, we say that S is *good for* G if each connected component of G contains at least one vertex of S. Given two integers i and j with $i \leqslant j$, we denote by $[i, j]$ the set of all integers k such that $i \leqslant k \leqslant j$. For an integer $i \geqslant 1$, we denote by $[i]$ the set of all integers k such that $1 \leqslant k \leqslant i$.

Reload costs and definition of the problem. For reload costs, we follow the notation and terminology defined by [241]. We consider edge-colored graphs $G = (V, E)$, where the colors are taken from a finite set X and the coloring function is $\chi : E(G) \to X$. The reload costs are given by a nonnegative function $c : X^2 \to \mathbb{N}_0$, which we assume for simplicity to be symmetric. The cost of traversing two incident edges e_1, e_2 is $c(e_1, e_2) := c(\chi(e_1), \chi(e_2))$. By definition, reload costs at the endpoints of a path equal zero. Consequently, the reload cost of a path with one edge also equals zero. The *reload cost* of a path P of length $\ell \geqslant 2$ with edges $e_1, e_2, \ldots, e_\ell$ is defined as $c(P) := \sum_{i=2}^{\ell} c(e_{i-1}, e_i)$. The induced reload cost distance function is given by $\mathsf{dist}^c_G(u, v) = \min\{c(P) \mid P \text{ is a path from } u \text{ to } v \text{ in } G\}$. The *diameter* of a tree T is $\mathsf{diam}(T) := \max_{u,v \in V} \mathsf{dist}^c_T(u, v)$, where for notational convenience we assume that the edge-coloring function χ and the reload cost function c are clear from the context.

The problem we study in this chapter can be formally defined as follows:

Minimum Reload Cost Diameter Spanning Tree (Diameter-Tree)
Input: A graph $G = (V, E)$ with an edge-coloring χ and a reload cost function c.
Output: A spanning tree T of G minimizing $\mathsf{diam}(T)$.

If for every three distinct edges e_1, e_2, e_3 of G incident to the same node, it holds that $c(e_1, e_3) \leqslant c(e_1, e_2) + c(e_2, e_3)$, we say that the reload cost function c satisfies the *triangle inequality*. This assumption is sometimes used in practical applications [241].

Nice tree-decompositions. In this chapter, we denote a tree-decomposition of a graph G by $\mathcal{D} = (Y, \mathcal{X})$, where Y is a tree and $\mathcal{X} = \{X_t \mid t \in V(Y)\}$ is a collection of subsets of $V(G)$. The notion of a *nice triple* defined below is essentially the same as the one of nice tree-decomposition in [96], which is the one we provided in Section 3.1.3.

Let $\mathcal{D} = (Y, \mathcal{X})$ be a tree-decomposition of G, r be a vertex of Y, and $\mathcal{G} = \{G_t \mid t \in V(Y)\}$ be a collection of subgraphs of G, indexed by the vertices of Y. We say that the triple $(\mathcal{D}, r, \mathcal{G})$ is *nice* if the following conditions hold:

- $X_r = \emptyset$ and $G_r = G$,
- each node of $\mathcal{D}$ has at most two children in Y,
- for each leaf $t \in V(Y)$, $X_t = \emptyset$ and $G_t = (\emptyset, \emptyset)$. Such a t is called a *leaf node*,
- if $t \in V(\mathcal{T})$ has exactly one child t', then either
  - $X_t = X_{t'} \cup \{v_{\text{insert}}\}$ for some $v_{\text{insert}} \notin X_{t'}$ and $G_t = (V(G_{t'}) \cup \{v_{\text{insert}}\}, E(G_{t'}))$. The node t is called *vertex-introduce node* and the vertex v_{insert} is the *insertion vertex* of X_t,
  - $X_t = X_{t'}$ and $G_t = (G_{t'}, E(G_{t'}) \cup \{e_{\text{insert}}\})$ where e_{insert} is an edge of G with endpoints in X_t. The node t is called *edge-introduce node* and the edge e_{insert} is the *insertion edge* of X_t, or
  - $X_t = X_{t'} \setminus \{v_{\text{forget}}\}$ for some $v_{\text{forget}} \in X_{t'}$ and $G_t = G_{t'}$. The node t is called *forget node* and v_{forget} is the *forget vertex* of X_t.
- if $t \in V(Y)$ has exactly two children t' and t'', then $X_t = X_{t'} = X_{t''}$, and $E(G_{t'}) \cap E(G_{t''}) = \emptyset$. The node t is called a *join node*.

As already argued in [96, 184], it is possible, given a tree-decomposition to transform it in polynomial time to a new one $\mathcal{D}$ of the same width and construct a collection $\mathcal{G}$ such that the triple $(\mathcal{D}, r, \mathcal{G})$ is nice.

Transfer triples and their fusion. Let (F, R, α) be a triple where F is a forest, $R \subseteq V(F)$, and $\alpha : R \times R^F \to [0, k] \cup \{\perp\}$, where $R^F = V(F) \cup (E(F) \setminus \mathsf{adj}_F(R))$. Keep in mind that R^F contains all vertices and edges of F except from the edges that are incident to vertices in R. We call (F, R, α) a *transfer triple* if, given a $(v, a) \in R \times R^F$, $\alpha(v, a) = \perp$ if and only if v and a belong in different connected components of F. The function α will be used for indicating for each pair (v, a) the "cost of transfering" from v to a in F (α is not necessarily a distance function).

Let (F_1, R_1, α_1) and (F_2, R_2, α_2) be two transfer triples where $R = R_1 = R_2$, $E(F_1) \cap E(F_2) = \emptyset$, and such that $F = F_1 \cup F_2$ is a forest. Let also $\beta : \mathsf{adj}_{F_1}(R) \times \mathsf{adj}_{F_2}(R) \to [0, k] \cup \{\perp\}$. We require a function $\alpha_1 \oplus_\beta \alpha_2 : R \times R^F \to [0, k] \cup \{\perp\}$ that builds the transferring costs of moving in F by taking into account the corresponding transferring costs in F_1 and F_2. The values of $\alpha_1 \oplus_\beta \alpha_2$ are defined as follows:

Let $(v, a) \in R \times R^F$. Let P be the shortest path in F containing v and a and let $V(P) = \{v_0, \ldots, v_r\}$, ordered in the way these vertices appear in P and assuming that $v_0 = v$. To simplify notation, we assume that $\{v_0, v_1\}$ is an edge of F_1 (otherwise, exchange the roles of F_1 and F_2). Given $i \in [r-1]$, we define e_i^- (resp. e_i^+) as the edge incident to v_i that appears before (resp. after) v_i when traversing P from v to a. We define the set of indices

$$I = \{i \mid e_i^- \text{ and } e_i^+ \text{ belong to different sets of } \{E(F_1), E(F_2)\}\}.$$

Let $I = \{i_1, \ldots, i_q\}$, where numbers are ordered in increasing order and we also set $i_0 = 0$. Then we set

$$\alpha_1 \oplus_\beta \alpha_2(v, a) = \sum_{h \in \left[0, \lfloor \frac{q-1}{2} \rfloor\right]} \alpha_1(v_{2i_h}, v_{2i_h+1}) + \sum_{h \in \left[0, \lfloor \frac{q-2}{2} \rfloor\right]} \alpha_2(v_{2i_h+1}, v_{2i_h+2}) + \sum_{h \in [q]} \beta(e^-_{i_h}, e^+_{i_h}) + \alpha_{(q \bmod 2)+1}(v_{i_q}, a).$$

Throughout the chapter, we let n, Δ, and tw denote the number of vertices, the maximum degree, and the treewidth of the input graph, respectively. When we consider the (parameterized) decision version of the Diameter-Tree problem, we also let k denote the desired cost of a solution.

8.3 Para-NP-hardness results

We start with the para-NP-hardness result with parameter $k + \mathsf{tw}$.

Theorem 8.1 *The* Diameter-Tree *problem is* NP*-hard on outerplanar graphs with only one vertex of degree greater than 3, even with three different costs that satisfy the triangle inequality, and $k = 9$. Since outerplanar graphs have treewidth at most 2, in particular,* Diameter-Tree *is* para-NP*-hard parameterized by* tw *and k.*

Proof: We present a simple reduction from 3-Sat. Given a formula φ with n variables and m clauses, we create an instance (G, χ, c) of Diameter-Tree as follows. We may assume that there is no clause in φ that contains a literal and its negation. The graph G contains a distinguished vertex r and, for each clause $c_j = (\ell_1 \vee \ell_2 \vee \ell_3)$, we add a clause gadget C_j consisting of three vertices $v^j_{\ell_1}, v^j_{\ell_2}, v^j_{\ell_3}$ and five edges $\{r, v^j_{\ell_1}\}$, $\{r, v^j_{\ell_2}\}$, $\{r, v^j_{\ell_3}\}$, $\{v^j_{\ell_1}, v^j_{\ell_2}\}$, and $\{v^j_{\ell_2}, v^j_{\ell_3}\}$. This completes the construction of G. Note that G does not depend on the formula φ except for the number of clause gadgets, and that it is an outerplanar graph with only one vertex of degree greater than 3; see Figure 8.1 for an illustration.

Let us now define the coloring χ and the cost function c. For simplicity, we associate a distinct color with each edge of G, and thus, with slight abuse of notation, it is enough to describe the cost function c for every pair of incident edges of G, as we consider symmetric cost functions. We will use just three different costs: 1, 5 and 10. We set

$$c(e_1, e_2) = \begin{cases} 10 & \text{if } e_1 = \{r, v^{j_1}_{\ell_{i_1}}\}, e_2 = \{r, v^{j_2}_{\ell_{i_2}}\} \text{ and } \ell_{i_1} = \overline{\ell_{i_2}}\ , \\ 5 & \text{if } e_1 = \{r, v^{j_1}_{\ell_{i_1}}\}, e_2 = \{r, v^{j_2}_{\ell_{i_2}}\} \text{ and } \ell_{i_1} \neq \overline{\ell_{i_2}}\ , \text{ and} \\ 1 & \text{otherwise.} \end{cases}$$

Note that this cost function satisfies the triangle inequality since the reload costs between edges incident to r are 5 and 10, and the reload costs between edges incident to other vertices are 1.

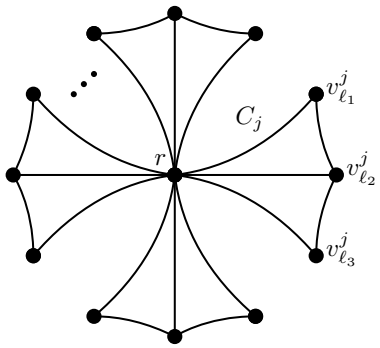

Figure 8.1: Example of the graph G built in the reduction of Theorem 8.1.

We claim that φ is satisfiable if and only if G contains a spanning tree with diameter at most 9. Since r is a cut vertex and every clause gadget is a connected component of $G-r$, in every spanning tree, the vertices of C_j together with r induce a tree with four vertices. Moreover the reload cost associated with a path from r to a leaf of this tree is always at most 2. Therefore, the diameter of any spanning tree is at most 4 plus the maximum reload cost incurred at r by a path of T.

Assume first that φ is satisfiable, fix a satisfying assignment ψ of φ, and let us construct a spanning tree T of G with diameter at most 9. For each clause c_j, the tree T^j is the tree spanning C_j and containing the edge between r and an arbitrarily chosen literal of c_j that is set to true by ψ. T is the union of all the trees T_j constructed in this way. The reload cost incurred at r by any path of T traversing it is at most 5, since we never choose a literal and its negation. Therefore, it holds that $\mathsf{diam}(T) \leqslant 9$.

Conversely, let T be a spanning tree of G with $\mathsf{diam}(T) \leqslant 9$. Then, the reload cost incurred at r by any path traversing it is at most 5 since otherwise $\mathsf{diam}(T) \geqslant 10$. For every $j \in [m]$, let T_j be the subtree of T induced by C_j and let $\{r, v^j_{\ell_{i_j}}\}$ be one of the edges incident to r in T_j. We note that for any pair of clauses c_{j_1}, c_{j_2} we have $\ell_{i_{j_1}} \neq \overline{\ell_{i_{j_2}}}$, since otherwise a path using these two edges would incur a cost of 10 at r. The variable in the literal ℓ_{i_j} is set by ψ so that ℓ_{i_j} is true. All the other variables are set to an arbitrary value by ψ. Note that ψ is well-defined, since we never encounter a literal and its negation during the assignment process. It follows that ψ is a satisfying assignment of φ. □

We proceed with the para-NP-hardness result with parameter $k + \Delta$.

Theorem 8.2 *The* Diameter-Tree *problem is* NP*-hard on graphs with* $\Delta \leqslant 3$*, even with two different costs,* $k = 0$*, and a bounded number of colors. In particular, it is* para-NP*-hard parameterized by* k *and* Δ*.*

Proof: We present a reduction from the restriction of 3-Sat to formulas where each variable occurs in at most three clauses; this problem was proved to be NP-complete by Tovey [236]. It is worth mentioning that one needs to allow for clauses of size two or three, as if all clauses have size exactly three, then it turns out that all instances are satisfiable [236].

We may assume that each variable occurs at least once positively and at least once negatively, as otherwise we may set such a variable x to the value that satisfies all clauses in which it appears, and delete x together with those clauses from the formula. We may also assume that each variable occurs *exactly* three times in the given formula φ. Indeed, let x be a variable occurring exactly two times in the formula. We create a new variable y and we add to φ two clauses $(x \vee y)$ and $(y \vee \overline{y})$. Let φ' be the new formula. Clearly φ and φ' are equivalent, and both x and y occur three times in φ'. Applying these operations exhaustively clearly results in an equivalent formula where each variable occurs exactly three times. Summarizing, we may assume the following property:

✠ *Each variable occurs exactly three times in the given formula φ of* 3-Sat. *Moreover, each variable occurs at least once positively and at least once negatively in φ.*

Given a formula φ with n variables and m clauses, we create an instance (G, χ, c) of Diameter-Tree with $\Delta(G) \leqslant 3$ as follows. Let the variables in φ be $x_1, \ldots, x_n$. For every $i \in [n]$, we add to G a *variable gadget* consisting of five vertices u_i, v_i, p_i, r_i, n_i and five edges $\{u_i, v_i\}, \{v_i, p_i\}, \{p_i, r_i\}, \{r_i, n_i\}$, and $\{n_i, v_i\}$. For every $i \in [n-1]$, we add the edge $\{u_i, u_{i+1}\}$. For every $j \in [m]$, the clause gadget in G consists of a single vertex c_j. We now proceed to explain how we connect the variable and the clause gadgets. For each variable x_i, we connect vertex p_i (resp. n_i) to one of the vertices corresponding to a clause of φ in which x_i appears positively (resp. negatively). Finally, we connect vertex r_i to the remaining clause in which x_i appears (positively or negatively). Note that these connections are well-defined because of property ✠. This completes the construction of G, and note that it indeed holds that $\Delta(G) \leqslant 3$; see Figure 8.2(a) for an example of the construction of G for a specific satisfiable formula φ with $n = 4$ and $m = 5$.

Let us now define the coloring χ and the cost function c. We use nine colors $1, 2, \ldots, 9$ associated with the edges of G as follows. For $i \in [n]$, we set $\chi(\{p_i, r_i\}) = 1$ and $\chi(\{r_i, n_i\}) = 2$, and all edges incident to u_i or v_i have color 3. Finally, for $j \in [m]$, we color the edges containing c_j with colors in $\{4, 5, 6, 7, 8, 9\}$, so that incident edges get different colors, and edges corresponding to positive (resp. negative) occurrences get colors in $\{4, 5, 6\}$ (resp. $\{7, 8, 9\}$); note that such a coloring always exists as each clause contains at most three variables; see Figure 8.2(b). We will use only two costs, namely 0 and 1, and recall that we consider only symmetric cost functions. We set $c(1, 2) = 1$, $c(1, i) = 1$ for every $i \in \{4, 5, 6\}$, $c(2, i) = 1$ for every $i \in \{7, 8, 9\}$, and $c(i, j) = 1$ for every distinct $4 \leqslant i, j \leqslant 9$. All other costs are set to 0.

We claim that φ is satisfiable if and only if G contains a spanning tree with diameter 0. Assume first that φ is satisfiable, fix a satisfying assignment ψ of φ, and let us construct a spanning tree T of G with diameter 0. For every $i \in [n]$, tree T contains all the edges containing vertex u_i or v_i. If variable x_i is set to true by ψ, we include the edge $\{r_i, n_i\}$ to T, and otherwise, that is, if x_i is set to false by ψ, we include the edge $\{p_i, r_i\}$. Finally, for $j \in [m]$, we add to T one of the edges containing c_j that corresponds to a literal satisfying that clause. It can be easily checked that T is a spanning tree of G with diameter 0; see Figure 8.2(a) for an example.

Conversely, let T be a spanning tree of G with diameter 0. Since the cost associated with any two distinct colors in $\{4, 5, 6, 7, 8, 9\}$ is 1, it follows that, for $j \in [m]$, vertex c_j has

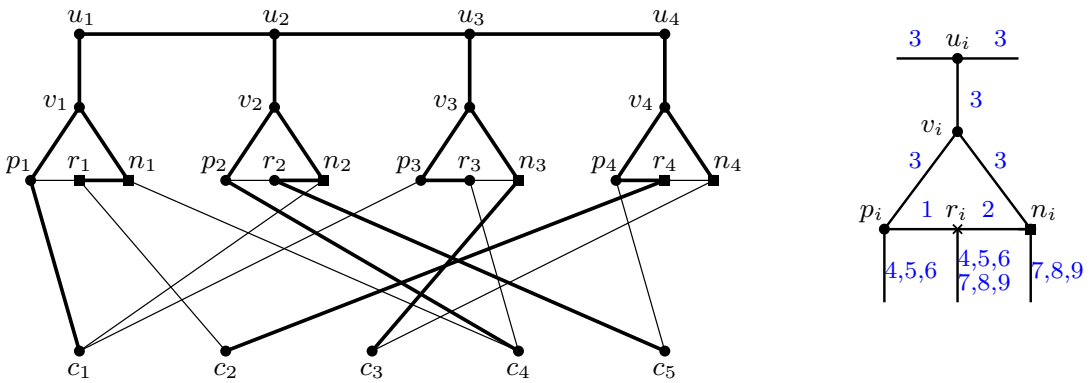

Figure 8.2: (a) Graph G described in the reduction of Theorem 8.2 for the formula $\varphi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (\overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_2 \vee x_4)$. The vertices p_i, r_i, n_i corresponding to positive (resp. negative) occurrences are depicted with circles (resp. squares). An assignment satisfying φ is given by $x_1 = x_2 = 1$ and $x_3 = x_4 = 0$, and a solution spanning tree T with diameter 0 is emphasized with thicker edges. (b) The (possible) colors associated with each edge of G are depicted in blue.

degree one in T. Therefore, the variable gadgets need to be connected in T via the vertices u_i, implying that all edges containing u_i, for $i \in [n]$, belong to T. For $i \in [n]$, in order for T to contain all four vertices v_i, p_i, r_i, n_i, by construction of G and since all clause vertices have degree one in T, tree T necessarily contains exactly three out of the four edges of the 4-cycle defined by v_i, p_i, r_i, n_i. Since $c(1,2) = 1$ and $\mathsf{diam}(T) = 0$, the missing edge is necessarily either $\{p_i, r_i\}$ or $\{r_i, n_i\}$. We define an assignment ψ of the variables $x_1, \ldots, x_n$ as follows: for $i \in [n]$, if the edge $\{r_i, n_i\}$ belongs to T, we set x_i to true; otherwise, we set x_i to false. We claim that ψ satisfies φ. Indeed, let c_j be a vertex in G corresponding to an arbitrary clause of φ. Since c_j has degree one in T, it is attached to exactly one of the vertices p_i, r_i, n_i for some $i \in [n]$. Suppose that the edge containing c_j corresponds to a positive occurrence of x_i, the other case being symmetric. Then, by construction, necessarily the edge containing c_j is either $\{c_j, p_i\}$ or $\{c_j, r_i\}$. In both cases, if the edge $\{p_i, r_i\}$ were in T, this edge together with $\{c_j, p_i\}$ or $\{c_j, r_i\}$ would incur a cost of 1 in T, contradicting the hypothesis that $\mathsf{diam}(T) = 0$. Therefore, the edge $\{p_i, r_i\}$ cannot be in T, implying that the edge $\{r_i, n_i\}$ must be in T. According to the definition of the assignment ψ, this implies that variable x_i is set to true in ψ, and therefore the clause corresponding to c_j is satisfied by variable x_i. This concludes the proof. □

Note that in the above reduction the cost function c does *not* satisfy the triangle inequality at vertices p_i or n_i for $i \in [n]$, and recall that this is unavoidable since otherwise the problem would be polynomial [241]. It is worth mentioning that using the ideas in the proof of [J13, Theorem 4] it can be proved that the Diameter-Tree problem is also NP-hard on *planar* graphs with $\Delta \leqslant 4$, $k = 0$, and a bounded number of colors; we omit the details here.

Finally, we present the para-NP-hardness result with parameter $\mathsf{tw} + \Delta$.

Theorem 8.3 *The* Diameter-Tree *problem is* NP*-hard on planar graphs with* tw $\leqslant 3$ *and* $\Delta \leqslant 3$*. In particular, it is* para-NP*-hard parameterized by* tw *and* Δ*.*

Proof: We present a reduction from the Partition problem, which is a typical example of a *weakly* NP-complete problem [154]. An instance of Partition is a multiset $S = \{a_1, a_2, \ldots, a_n\}$ of n positive integers, and the objective is to decide whether S can be partitioned into two subsets S_1 and S_2 such that $\sum_{x \in S_1} x = \sum_{x \in S_2} x = \frac{B}{2}$ where $B = \sum_{x \in S} x$.

Given an instance $S = \{a_1, a_2, \ldots, a_n\}$ of Partition, we create an instance (G, χ, c) of Diameter-Tree as follows. The graph G contains a vertex r, called the root, and for every integer a_i where $i \in [n]$, we add to G six vertices $u_i, u'_i, m_i, m'_i, d_i, d'_i$ and seven edges $\{u_i, u'_i\}$, $\{m_i, m'_i\}$, $\{d_i, d'_i\}$, $\{u_i, m_i\}$, $\{u'_i, m'_i\}$, $\{m_i, d_i\}$, and $\{m'_i, d'_i\}$. We denote by H_i the subgraph induced by these six vertices and seven edges. We add the edges $\{r, u_1\}, \{r, d_1\}$ and, for $i \in [n-1]$, we add the edges $\{u'_i, u_{i+1}\}$ and $\{d'_i, d_{i+1}\}$. Let G' be the graph constructed so far. We then define G to be the graph obtained from two disjoint copies of G' by adding an edge between both roots. Note that G is a planar graph with $\Delta(G) = 3$ and $\mathsf{tw}(G) = 3$. (The claimed bound on the treewidth can be easily seen by building a *path* decomposition of G with consecutive bags of the form $\{u'_{i-1}, d'_{i-1}, u_i, d_i\}, \{u_i, d_i, m_i, u'_i\}, \{d_i, m_i, u'_i, m'_i\}, \{d_i, u'_i, m'_i, d'_i\}, \ldots$.)

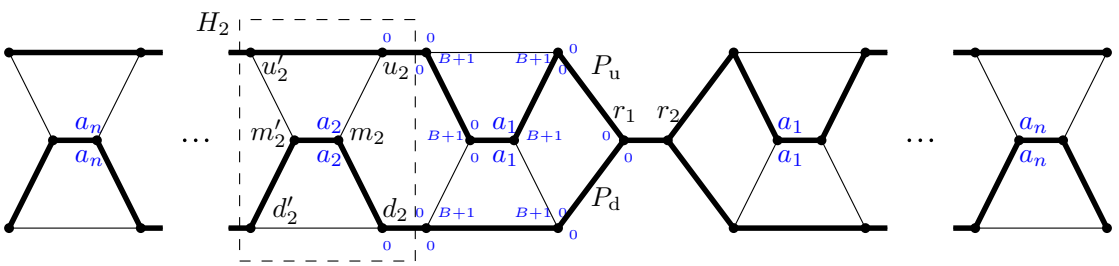

Figure 8.3: Graph G built in the reduction of Theorem 8.3, where the reload costs are depicted in blue at the angle between the two corresponding edges. For better visibility, not all costs and vertex labels are depicted. The typical shape of a solution spanning tree is highlighted with thicker edges.

Let us now define the coloring χ and the cost function c. Again, for simplicity, we associate a distinct color with each edge of G, and thus it is enough to describe the cost function c for every pair of incident edges of G. We define the costs for one of the copies of G', and the same costs apply to the other copy. For every edge e being either $\{u'_i, u_{i+1}\}$ or $\{d'_i, d_{i+1}\}$, for $1 \leqslant i \leqslant n-1$, we set $c(e, e') = 0$ for each of the four edges e' incident with e. For every edge $e = \{m_i, m'_i\}$, for $1 \leqslant i \leqslant n$, we set $c(\{u_i, m_i\}, e) = c(\{d_i, m_i\}, e) = a_i$ and $c(e, \{m'_i, u'_i\}) = c(e, \{m'_i, d'_i\}) = 0$. All costs associated with the two edges containing r in one of the copies G' are set to 0. For $e = \{r_1, r_2\}$, where r_1 and r_2 are the roots of the two copies of G', we set $c(e, e') = 0$ for each of the four edges e' incident to e. The cost associated with any other pair of edges of G is equal to $B + 1$; see Figure 8.3 for an illustration, where (some of) the reload costs are depicted in blue, and a typical solution spanning tree of G is drawn with thicker edges.

We claim that the instance S of Partition is a Yes-instance if and only if G has a spanning tree with diameter at most B.

Assume first that S is a Yes-instance of Partition, and let $S_1, S_2 \subseteq S$ be a solution. We define a spanning tree T of G with diameter B as follows. We describe the subtree of T restricted to one of the copies of G', say T'. The spanning tree T of G is defined by union of two symmetric copies of T', one in each copy of G', together with the edge $\{r_1, r_2\}$. Tree T' consists of the two edges $\{r, u_1\}, \{r, d_1\}$ and two paths $P_\mathrm{u}, P_\mathrm{d}$ (corresponding to the upper and the lower path, respectively defined as follows; see Figure 8.3). For $i \in [n-1]$, the path P_u (resp. P_d) contains the edge $\{u'_i, u_{i+1}\}$ (resp. $\{d'_i, d_{i+1}\}$), and if $a_i \in S_1$ we add the three edges $\{u_i, m_i\}, \{m_i, m'_i\}, \{m'_i, u'_i\}$ to P_u, and the edge $\{d_i, d'_i\}$ to P_d. Otherwise, if $a_i \in S_2$, we add the edge $\{u_i, u'_i\}$ to P_u and the three edges $\{d_i, m_i\}, \{m_i, m'_i\}, \{m'_i, d'_i\}$ to P_d. Since $\sum_{x \in S_1} x = \sum_{x \in S_2} x = \frac{B}{2}$, it can be easily checked that both paths P_u and P_d have diameter $\frac{B}{2}$ in each of the two copies of G', and therefore T is a spanning tree of G with diameter B.

Conversely, let T be a spanning tree of G with $\mathsf{diam}(T) \leqslant B$. Let G_1, G_2 be the two copies of G' in G, and let r_1, r_2 be their respective roots. Since the edge $\{r_1, r_2\}$ is a bridge of G, it necessarily belongs to T. By the construction of G, the choice of the reload costs, and since $\mathsf{diam}(T) \leqslant B - 1$, it can be verified that, for $j \in \{1, 2\}$, $T \cap G_j$ consists of two paths $P^j_\mathrm{u}, P^j_\mathrm{d}$ intersecting at the root r_i. Furthermore, P^j_u (resp. P^j_d) contains the edge $\{u'_i, u_{i+1}\}$ (resp. $\{d'_i, d_{i+1}\}$) of the corresponding copy of G', and the intersection of P^j_u (resp. P^j_d) with the subgraph H_i in the corresponding copy of G' is given by either the three edges $\{u_i, m_i\}, \{m_i, m'_i\}, \{m'_i, u'_i\}$ (resp. $\{d_i, m_i\}, \{m_i, m'_i\}, \{m'_i, d'_i\}$) or by the edge $\{u_i, u'_i\}$ (resp. $\{d_i, d'_i\}$). Therefore, for $j \in \{1, 2\}$ and $x \in \{\mathrm{u}, \mathrm{d}\}$, it holds that $d^j_x := \mathsf{diam}(P^j_x) = \sum_{i \in I^j_x} a_i$, where I^j_x is the set of indices $i \in \{1, \ldots, n\}$ such that the edge $\{m_i, m'_i\}$ belongs to path P^j_x. Note also that, for $j \in \{1, 2\}$, by construction we have that $d^j_\mathrm{u} + d^j_\mathrm{d} = \sum_{i=1}^n a_i$, implying in particular that $\max\{d^j_\mathrm{u}, d^j_\mathrm{d}\} \geqslant \frac{B}{2}$. On the other hand, by the structure of T it holds that

$$B \geqslant \mathsf{diam}(T) \geqslant \max\{d^1_\mathrm{u}, d^1_\mathrm{d}\} + \max\{d^2_\mathrm{u}, d^2_\mathrm{d}\} \geqslant \frac{B}{2} + \frac{B}{2} = B. \tag{8.1}$$

Equation (8.1) implies, in particular, that $d^1_\mathrm{u} = d^1_\mathrm{d} = \frac{B}{2}$. In other words, $\sum_{i \in I^1_\mathrm{u}} a_i = \sum_{i \in I^1_\mathrm{d}} a_i = \frac{B}{2}$, thus the sets $I^1_\mathrm{u}, I^1_\mathrm{d}$ define a solution of Partition. This completes the proof. □

8.4 A polynomial-time algorithm on cactus graphs

In this section we present a polynomial-time algorithm to solve the Diameter-Tree problem on cactus graphs, equivalently called *cacti*. We first need some definitions.

A *biconnected component*, or *block*, of a graph is a maximal biconnected induced subgraph of it. The *block tree* of a graph G is a tree T whose nodes are the cut vertices and the blocks of G. Every cut vertex is adjacent in T to all the blocks that contain it. Two blocks share at most one vertex. The block tree of a graph is unique and can be computed

in polynomial time [106]. A graph is a *cactus* graph if every block of it is either a cycle or a single edge. We term these blocks *cycle blocks* and *edge blocks*, respectively. It is well-known that cacti have treewidth at most 2. Given a forest F and two vertices x and y, we define $\texttt{cost}_F(x, y)$ to be $\mathsf{dist}^c_T(x, y)$ if x and y are in the same tree T of F and where c is the given reload cost function, and $\perp$ otherwise. Given a tree T and a vertex $v \in V(T)$, we define the *eccentricity* of v in T to be $\max_{v' \in V(T)} \texttt{cost}_T(v, v')$.

We present a polynomial-time algorithm that solves the decision version of the problem, which we call Diameter-Tree*: the input is an edge-colored graph G and an integer k, and the objective is to decide whether the input graph G has a spanning tree with reload cost diameter at most k. The algorithm to solve Diameter-Tree* uses dynamic programming on the block tree of the input graph.

As we aim at a *strongly* polynomial-time algorithm to solve Diameter-Tree, we *cannot* afford to solve the decision version for all values of k. To overcome this problem, we perform a double *binary search* on the possible solution values and two appropriate eccentricities, resulting (skipping many technical details) in an extra factor of $(\log \mathsf{opt})^2$ in the running time of the algorithm, where opt is the diameter of a minimum cost spanning tree. This yields a polynomial-time algorithm solving Diameter-Tree on cactus graphs.

Roughly speaking, the algorithm first fixes an arbitrary non-cut vertex r of G and the block B_r that contains it. Then it processes the block tree of G in a bottom-up manner starting from its leaves, proceeding towards B_r while maintaining partial solutions for each block. At each step of the processing, it uses an algorithm that solves an instance of the 2-Sat problem as a subroutine. The intuition behind the instances of 2-Sat created by the algorithm is the following.

Suppose that we are dealing with a cycle block B of the block tree of G (the case of an edge block being easier). Note that any spanning tree of G contains all edges of B except one. Let G_B be the graph processed so far (including B). For each potential partial solution $\mathcal{Q}$ in G_B, we associate, with each edge e of B, a variable that indicates that e is the non-picked edge by the solution in B. Now, for any *two* such variables corresponding to intersecting blocks, we add to the formula of 2-Sat essentially two types of clauses: the first set of clauses, namely ϕ_1, guarantees that the non-picked edges (corresponding to the variables set to `true` in the eventual assignment) indeed define a spanning tree of G_B, while the second one, namely ϕ_2, forces this solution to have diameter and eccentricity not exceeding the given budget k. The fact the G is a cactus allows to prove that these constraints containing only two variables are enough to compute an optimal solution in G_B.

Theorem 8.4 *The* Diameter-Tree *problem can be solved in polynomial time on cacti.*

Proof: We start with a few more definitions needed in the algorithm. Given a graph G, we denote by $\mathcal{S}(G)$ the set of spanning trees of G, and by $\mathcal{B}(G)$ the set of blocks of G. We omit G from the notation if no ambiguity arises. We assume without loss of generality that G is connected. For a block B, we denote by $\mathcal{C}(B)$ the set of blocks that are immediate descendants of B in the block tree. With a slight abuse (since we ignore the cut vertices in the block tree), we will refer to them as the *children* of B. The *parent* of a block B

is the first block after B on the path from B to B_r in the block tree. We denote by G_B the subgraph of G induced by the union of all descendants B (including B itself). The *anchor* $a(B)$ of a block B is the cut vertex separating B from its parent if $B \neq B_r$, and r if $B = B_r$.

Let B be a cycle block, $e = \{x, y\}$, and assume, without loss of generality, that $y \neq a(B)$. Clearly, the graph $G_B - e$ is connected. Moreover, $a(B)$ is a cut vertex of $G_B - e$ unless $x = a(B)$. For $z \in \{x, y\}$ we define $S_B^{z,e}$ as the set of vertices that are reachable from z in $G_B - e$ without traversing $a(B)$. See Figure 8.4 for an illustration. We denote the subgraph of $G_B - e$ induced by $S_B^{z,e}$, as $G_B^{z,e}$. Note that z and $a(B)$ are in $S_B^{z,e}$ and if $x = a(B)$ then $S_B^{x,e} = \{a(B)\}$. Since the degree of $a(B)$ in $G_B - e$ is at most two, a spanning tree T of $G_B - e$ is a union of two spanning trees, a tree $T[S_B^{x,e}]$ spanning $G_B^{x,e}$ and a tree $T[S_B^{y,e}]$ spanning $G_B^{y,e}$. Moreover, $T[S_B^{x,e}]$ and $T[S_B^{y,e}]$ intersect only at $a(B)$.

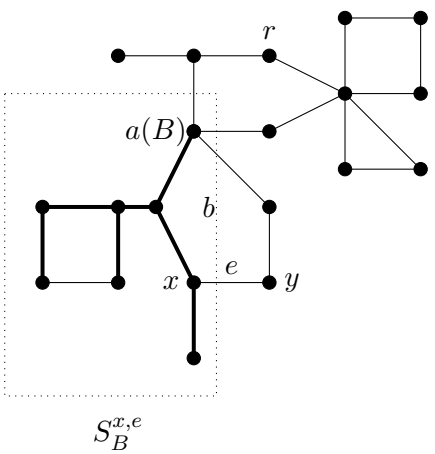

Figure 8.4: A cactus with 8 blocks, 5 cycle blocks, and 3 edge blocks. The vertices inside the dotted rectangle are the vertices of $S_B^{x,e}$ and the bold path corresponds to a possible $R_B^{x,e}$.

We proceed with the description of the algorithm. At every block B, we compute a function $\lambda_B : E(B) \to \mathcal{S}(G_B) \cup \{\perp\}$ of partial solutions. If B is an edge block consisting of the edge e, then $\lambda_B(e)$ is:

- a spanning tree of G_B,
- of diameter at most k
- that minimizes the eccentricity of $a(B)$,

if such a tree exists, and $\perp$ otherwise.

If B is a cycle block and $e = \{x, y\}$ an edge of B, then $\lambda_B(e)$ is:

- a spanning tree T of $G_B - e$,
- of diameter at most k

- that minimizes the eccentricities of $a(B)$ in both $T[S_B^{x,e}]$ and $T[S_B^{y,e}]$

if such a tree exists, and $\perp$ otherwise. Note that, as $G_B^{x,e}$ and $G_B^{y,e}$ have only the vertex $a(B)$ in common, minimizing the eccentricities of $a(B)$ in $T[S_B^{x,e}]$ and minimizing the eccentricities of $a(B)$ in $T[S_B^{y,e}]$ are two independent objectives.

If for some block B we have $\lambda_B(e) = \perp$ for every edge e of B, then G_B (and therefore G as well) does not contain a spanning tree of diameter at most k. In this case the algorithm stops and returns NO. Otherwise, the processing continues until finally B_r is processed successfully and the algorithm returns YES, since there exists $e \in E(B_r)$ such that $\lambda_{B_r}(e) \neq \perp$ which constitutes a spanning tree of G with diameter at most k.

Given a cycle block B, an edge $e = \{x, y\}$ of B, a subgraph T of G_B, and two integers i and j, we say that T satisfies the (e, i, j)-*condition* if:

- T is a tree, of diameter at most k, that does not contain e,
- the eccentricity of $a(B)$ in $T[S_B^{x,e}]$ is at most i, and
- the eccentricity of $a(B)$ in $T[S_B^{y,e}]$ is at most j.

Given an edge block B, a subgraph T of G_B, and an integer i, we say that T satisfies the (i)-*condition* if:

- T is a tree of diameter at most k and
- the eccentricity of $a(B)$ in T is at most i.

Let us fix a block B, and an edge e of $E(B)$. In the sequel our goal is to describe how to compute $\lambda_B(e)$. We can assume that for every child C of B, the function λ_C has already been computed and C contains at least one edge e' such that $\lambda_C(e') \neq \perp$, since otherwise the algorithm would have stopped.

We define T^e to be the tree obtained by taking the union of all the following:

- the graph $B - e$ if B is a cycle block,
- the graph B if B is an edge block, and
- $\lambda_C(e_C)$ for every child C of B that is an edge block containing only the edge e_C.

For every child C of B that is a cycle block, for every edge e' of C such that $\lambda_C(e') \neq \perp$ and for $x' \in e'$, the tree $R_C^{x',e'}$ is $\lambda_C(e')[S_B^{x',e'}]$. Note that, given a child C of B that is a cycle block, and three vertices v, v', v'' of $V(C)$ such that $v \neq v''$, $v' \neq a(C)$, and $\{v, v'\}$ and $\{v', v''\}$ are in $E(C)$, if $R_C^{v,\{v,v'\}}$ and $R_C^{v',\{v',v''\}}$ are defined, then $R_C^{v,\{v,v'\}}$ is a subgraph of $R_C^{v',\{v',v''\}}$. We define $\mathcal{R}_B = \{R_C^{x',e'} \mid C \in \mathcal{C}(B) \text{ is a cycle block}, e' \in E(C), \lambda_C(e') \neq \perp, x' \in e'\}$.

For $\mathcal{Q} \subseteq \mathcal{R}_B$ we denote by $T_{\mathcal{Q}}^e$ the graph obtained by taking the union of T^e and $\mathcal{Q}$. If there exists $R \in \mathcal{R}_B$ such that $\mathcal{Q} = \{R\}$, we write T_R^e instead of $T_{\mathcal{Q}}^e$. We define $\texttt{close}_{\mathcal{R}_B}(\mathcal{Q})$ to be the set of elements of $\mathcal{R}_B$ that are subgraphs of $T_{\mathcal{Q}}^e$. Note that $T_{\mathcal{Q}}^e = T_{\texttt{close}_{\mathcal{R}_B}(\mathcal{Q})}^e$.

If B is a cycle block, we define for each $i, j \in [0, k]$ the set $\mathcal{R}_B^{(e,i,j)} = \{R \in \mathcal{R}_B \mid T_R^e$ satisfies the (e, i, j)-condition$\}$. If B is an edge block, we define for each $i \in [0, k]$ the set $\mathcal{R}_B^{(i)} = \{R \in \mathcal{R}_B \mid T_R^e$ satisfies the (i)-condition$\}$.

Note that, if B is a cycle block (resp. an edge block), then for each $i, j \in [0, k]$ and for each $R_1, R_2 \in \mathcal{R}_B$ such that R_2 is a subtree of R_1, then if $R_2 \notin \mathcal{R}_B^{(e,i,j)}$ (resp. $R_2 \notin \mathcal{R}_B^{(i)}$) , we have $R_1 \notin \mathcal{R}_B^{(e,i,j)}$ (resp. $R_1 \notin \mathcal{R}_B^{(i)}$).

We associate a boolean variable $\mathbf{v}(R) = \mathbf{v}_C^{x',e'}$ with each $R = R_C^{x',e'} \in \mathcal{R}_B$. With a slight abuse of notation, we say that a set $\mathcal{Q} \subseteq \mathcal{R}_B$ satisfies a formula ϕ over these variables if ϕ is satisfied when each variable of $\{\mathbf{v}(R) \mid R \in \mathcal{Q}\}$ is set to true and each variable of $\{\mathbf{v}(R) \mid R \in \mathcal{R}_B \setminus \mathcal{Q}\}$ is set to false simultaneously. In the following we are going to build three formulas ϕ_0, ϕ_1, and ϕ_2, and if $\mathcal{Q} \subseteq \mathcal{R}_B$ satisfies $\phi_0 \wedge \phi_1 \wedge \phi_2$ then this implies that $T_{\mathcal{Q}}^e$ is a correct value for $\lambda_B(e)$.

Along with the description, the reader is referred to Figure 8.5 to get some intuition about the formulas ϕ_0, ϕ_1, and ϕ_2. In this figure, we have a cycle block B with two children C_1 and C_2. As we will see later, when computing $\lambda_B(e)$ in this example, we have:

$$\begin{aligned}\phi_0 \;=\;& (\mathbf{v}(R_{C_1}^{v',\{v',v''\}}) \Rightarrow \mathbf{v}(R_{C_1}^{v,\{v,v'\}})) \wedge (\mathbf{v}(R_{C_1}^{v'',\{v'',v\}}) \Rightarrow \mathbf{v}(R_{C_1}^{v',\{v',v''\}})) \wedge \\ & (\mathbf{v}(R_{C_1}^{v'',\{v'',v'\}}) \Rightarrow \mathbf{v}(R_{C_1}^{v,\{v,v''\}})) \wedge (\mathbf{v}(R_{C_1}^{v',\{v',v\}}) \Rightarrow \mathbf{v}(R_{C_1}^{v'',\{v'',v''\}})) \wedge \\ & (\mathbf{v}(R_{C_2}^{w',\{w',w''\}}) \Rightarrow \mathbf{v}(R_{C_2}^{w,\{w,w'\}})) \wedge (\mathbf{v}(R_{C_2}^{w'',\{w'',w\}}) \Rightarrow \mathbf{v}(R_{C_2}^{w',\{w',w''\}})) \wedge \\ & (\mathbf{v}(R_{C_2}^{w'',\{w'',w'\}}) \Rightarrow \mathbf{v}(R_{C_2}^{w,\{w,w''\}})) \wedge (\mathbf{v}(R_{C_2}^{w',\{w',w\}}) \Rightarrow \mathbf{v}(R_{C_2}^{w'',\{w'',w''\}})),\end{aligned}$$

$$\begin{aligned}\phi_1 \;=\;& (\mathbf{v}(R_{C_1}^{v',\{v',v''\}}) \vee \mathbf{v}(R_{C_1}^{v',\{v',v\}})) \wedge (\overline{\mathbf{v}(R_{C_1}^{v',\{v',v''\}})} \vee \overline{\mathbf{v}(R_{C_1}^{v',\{v',v\}})}) \wedge \\ & (\mathbf{v}(R_{C_1}^{v'',\{v'',v'\}}) \vee \mathbf{v}(R_{C_1}^{v'',\{v'',v\}})) \wedge (\overline{\mathbf{v}(R_{C_1}^{v'',\{v'',v'\}})} \vee \overline{\mathbf{v}(R_{C_1}^{v'',\{v'',v\}})}) \wedge \\ & (\mathbf{w}(R_{C_2}^{w',\{w',w''\}}) \vee \mathbf{w}(R_{C_2}^{w',\{w',w\}})) \wedge (\overline{\mathbf{w}(R_{C_2}^{w',\{w',w''\}})} \vee \overline{\mathbf{w}(R_{C_2}^{w',\{w',w\}})}) \wedge \\ & (\mathbf{w}(R_{C_2}^{w'',\{w'',w'\}}) \vee \mathbf{w}(R_{C_2}^{w'',\{w'',w\}})) \wedge (\overline{\mathbf{w}(R_{C_2}^{w'',\{w'',w'\}})} \vee \overline{\mathbf{w}(R_{C_2}^{w'',\{w'',w\}})}),\end{aligned}$$ and

for every two vertices of $V(C_1) \cup V(C_2)$, say v' and w'', the clause $\overline{\mathbf{v}(R_{C_1}^{v',\{v',v''\}})} \vee \overline{\mathbf{v}(R_{C_2}^{w'',\{w'',w'\}})}$ is a clause of ϕ_2 if and only if the path defined by v', v, w, w'' has diameter greater than k. In the general case, the clauses deal with $T^e_{\{R_{C_1}^{v',\{v',v''\}}, R_{C_2}^{w'',\{w'',w'\}}\}}$ instead of the path v', v, w, w'', but the main idea behind the clauses is the same.

We construct a 2-Sat formula ϕ_0 such that for each $R_1, R_2 \in \mathcal{R}_B$ where R_2 is a subgraph of R_1, ϕ_0 contains the clause $\mathbf{v}(R_1) \Rightarrow \mathbf{v}(R_2)$. It is easy to show that given $\mathcal{Q} \subseteq \mathcal{R}_B$, $\mathcal{Q}$ satisfies ϕ_0 if and only if $\mathcal{Q} = \texttt{close}_{\mathcal{R}_B}(\mathcal{Q})$.

We construct a 2-Sat formula ϕ_1 as follows. For every child C of B that is a cycle block, and every two consecutive edges $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_2, v_3\}$ of C such that

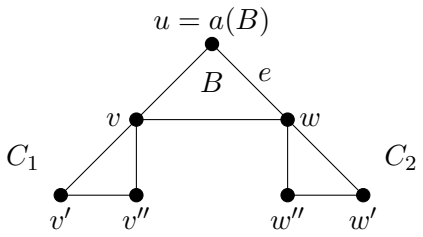

Figure 8.5: Example of a cycle block B with two children C_1 and C_2.

$a(C) \notin \{v_1, v_2\}$ and $R_C^{v_2,e_1}, R_C^{v_2,e_2} \in \mathcal{R}_B$, we add to ϕ_1 two clauses $\mathbf{v}_C^{v_2,e_1} \vee \mathbf{v}_C^{v_2,e_2}$ and $\overline{\mathbf{v}_C^{v_2,e_1}} \vee \overline{\mathbf{v}_C^{v_2,e_2}}$. With this definition of ϕ_1, we now state the following lemma.

Lemma 8.1 *Let $\mathcal{Q}$ be a subset of $\mathcal{R}_B$ such that $\mathcal{Q}$ satisfies ϕ_0. $\mathcal{Q}$ satisfies ϕ_1 if and only if $T^e_{\mathcal{Q}}$ is a spanning tree of G_B.*

Proof: Let $\mathcal{Q} \subseteq \mathcal{R}_B$. First assume that $T^e_{\mathcal{Q}}$ is a spanning tree of G_B. Let $C \in \mathcal{C}(B)$ be a cycle block, and let $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_2, v_3\}$ be two consecutive edges of C such that $e_1 \neq e_2$, $v_1 \neq a(C)$, and $v_2 \neq a(C)$. As $T^e_{\mathcal{Q}}$ is connected, the clause $\mathbf{v}_C^{v_2,e_1} \vee \mathbf{v}_C^{v_2,e_2}$ is satisfied. As $T^e_{\mathcal{Q}}$ does not contain any cycle, the clause $\overline{\mathbf{v}_C^{v_2,e_1}} \vee \overline{\mathbf{v}_C^{v_2,e_2}}$ is satisfied.

Assume now that $\mathcal{Q}$ satisfies ϕ_1, and let z be a vertex of G_B. If $z \in V(B)$, then there is a path from z to $a(B)$ in T^e and hence also in $T^e_{\mathcal{Q}}$. Otherwise, let $C_z \in \mathcal{C}(B)$ be the block such that $z \in V(G_{C_z})$, and let $s(z)$ be the only vertex of $V(C_z)$ such that $s(z) \neq a(C_z)$ and each path in G_B from $a(B)$ to z contains $s(z)$. Note that if $z \in V(C_z)$, then $s(z) = z$. If C_z is an edge block, then $z \in V(T^e)$; therefore, there is a path from $a(B)$ to z in $T^e_{\mathcal{Q}}$. Otherwise, if C_z is a cycle block, then the condition $\mathbf{v}_C^{v_2,e_1} \vee \mathbf{v}_C^{v_2,e_2}$, with $v_2 = s(z)$, ensures that $z \in T^e_{\mathcal{Q}}$ and that there is a path in $T^e_{\mathcal{Q}}$ from $a(B)$ to z. Thus, $T^e_{\mathcal{Q}}$ is connected and $V(T^e_{\mathcal{Q}}) = V(G_B)$. We need to show that $T^e_{\mathcal{Q}}$ does not contain any cycle. By construction of $T^e_{\mathcal{Q}}$, if it contains a cycle, this cycle should be C where $C \in \mathcal{C}(B)$. The condition $\overline{\mathbf{v}_C^{v_2,e_1}} \vee \overline{\mathbf{v}_C^{v_2,e_2}}$ ensures that C is not a subgraph of $T^e_{\mathcal{Q}}$. □

We build a formula ϕ_2 over the variables $\{\mathbf{v}(R) \mid R \in \mathcal{R}_B^{(e,k,k)}\}$. For each $R_1, R_2 \in \mathcal{R}_B^{(e,k,k)}$, $R_1 \neq R_2$, if $T^e_{\{R_1,R_2\}}$ has diameter greater than k, then we add the clause $\overline{\mathbf{v}(R_1)} \vee \overline{\mathbf{v}(R_2)}$ to ϕ_2. With this definition of ϕ_2, we now state the following lemma.

Lemma 8.2 *Let B be a cycle block (resp. an edge block), i and j be two integers of $[0, k]$, and $\mathcal{Q}$ be a subset of $\mathcal{R}_B^{(e,i,j)}$ (resp. of $\mathcal{R}_B^{(i)}$) such that $\mathcal{Q}$ satisfies ϕ_0 and ϕ_1. $\mathcal{Q}$ satisfies ϕ_2 and T^e satisfies the (e,i,j)-condition (resp. the (i)-condition) if and only if $T^e_{\mathcal{Q}}$ is a spanning tree of G_B that satisfies the (e,i,j)-condition (resp. the (i)-condition).*

Proof: Assume that B is a cycle block. Let i, j be two integers in $[0, k]$ and let $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i,j)}$.

First assume that $T^e_{\mathcal{Q}}$ is a spanning tree of G_B that satisfies the (e,i,j)-condition. This directly implies that T^e also satisfies the (e,i,j)-condition. It remains to show that $\mathcal{Q}$ satisfies ϕ_2. For this, assume that there exist R_1 and R_2 in $\mathcal{Q}$ such that $T^e_{\{R_1,R_2\}}$ has

diameter more than k. Since $T^e_{\{R_1,R_2\}}$ is a subtree of $T^e_{\mathcal{Q}}$, this implies that $T^e_{\mathcal{Q}}$ also has diameter more than k, which is a contradiction because $T^e_{\mathcal{Q}}$ satisfies the (e,i,j)-condition.

Assume now that $\mathcal{Q}$ satisfies ϕ_2 and T^e satisfies the (e,i,j)-condition. As $\mathcal{Q}$ satisfies ϕ_1, we know by Lemma 8.1 that $T^e_{\mathcal{Q}}$ is a spanning tree of G_B. As $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i,j)}$ and T^e satisfies the (e,i,j)-condition, then the eccentricity of $a(B)$ in $T^e_{\mathcal{Q}}[S_B^{x,e}]$ is at most i, and the eccentricity of $a(B)$ in $T^e_{\mathcal{Q}}[S_B^{y,e}]$ is at most j. Indeed, let $z \in S_B^{x,e}$. If $z \in V(B)$ then, as T^e satisfies the (e,i,j)-condition, we have that $\texttt{cost}_{T^e_{\mathcal{Q}}}(a(b),z) \leqslant i$ If $z \notin V(B)$ then, as $T^e_{\mathcal{Q}}$ is a spanning tree of G_B, we have that there exists $R \in \mathcal{Q}$ such that $z \in V(R)$. By definition of $\mathcal{R}_B^{(e,i,j)}$, we obtain that $\texttt{cost}_{T^e_{\mathcal{Q}}}(a(b),z) \leqslant i$ The same argument applies if $z \in S_B^{y,e}$. It remains to show that $T^e_{\mathcal{Q}}$ is of diameter at most k. Let z and z' be two vertices of $T^e_{\mathcal{Q}}$. If both z and z' are in $V(B)$, then as T^e satisfies the (e,i,j)-condition, this implies that $\texttt{cost}_{T^e_{\mathcal{Q}}}(z,z') \leqslant k$. If $z \in V(B)$ and $z' \notin V(B)$ then, as $T^e_{\mathcal{Q}}$ is a spanning tree of G_B, there exists $R' \in \mathcal{Q}$ such that $z' \in V(R')$. As $R' \in \mathcal{R}_B^{(e,i,j)}$, $\texttt{cost}_{T^e_{\mathcal{Q}}}(z,z') \leqslant k$. Otherwise, if both z and z' are not in $V(B)$, since $T^e_{\mathcal{Q}}$ is a spanning tree of G_B, there exist $R, R' \in \mathcal{Q}$ such that $z \in V(R)$ and $z' \in V(R')$. If $R = R'$, then as $R' \in \mathcal{R}_B^{(e,i,j)}$, $\texttt{cost}_{T^e_{\mathcal{Q}}}(z,z') \leqslant k$. Otherwise, as $\mathcal{Q}$ satisfies ϕ_2, then $T^e_{\{R,R'\}}$ has diameter at most k; therefore, $\texttt{cost}_{T^e_{\mathcal{Q}}}(z,z') \leqslant k$.

The same arguments apply if B is an edge block. □

Lemma 8.3 *If B is a cycle block (resp. an edge block) and if there exists a spanning tree $\hat{T}_B$ of that satisfies the (e,i,j)-condition (resp. (i)-condition) for some $i,j \in [0,k]$, then there exists $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i,j)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2.*

Proof: For readability, we consider the case where B is an edge block. Let $x = a(B)$. Assume that there exists $\hat{T}_B$, a spanning tree of G_B, that satisfies the (i)-condition for some $i \in [0,k]$. We define $\mathcal{Q} = \texttt{close}_{\mathcal{R}_B}(\{R_C^{x',e'} \mid C \in \mathcal{C}(B), C \text{ is a cycle block}, e' \in E(C), e' \notin E(\hat{T}_B[V(C)]), x' \in e'\})$ and we claim that $\mathcal{Q}$ satisfies ϕ_0, ϕ_1, and ϕ_2. By definition of $\texttt{close}_{\mathcal{R}_B}$, $\mathcal{Q}$ satisfies ϕ_0. It is not difficult to see that $T^e_{\mathcal{Q}}$ is a spanning tree and so, by Lemma 8.1, $\mathcal{Q}$ satisfies ϕ_1. Let z be a vertex of $V(G_B)\setminus V(B)$, and let $C \in \mathcal{C}(B)$ such that $z \in V(G_C)$. The path in $\hat{T}_B$ and the path in $T^e_{\mathcal{Q}}$ from $a(B)$ to z use exactly the same edges of C. This implies that $\texttt{cost}_{T^e_{\mathcal{Q}}}(a(B),z) \leqslant i$. Otherwise $\hat{T}_B[V(G_C)]$ would have been a better value for $\lambda_C(e')$ for the only compatible edge $e' \in E(C)$. With the same arguments we show that $T^e_{\mathcal{Q}}$ has diameter at most k. This implies that $T^e_{\mathcal{Q}}$ satisfies the (i)-condition, and so $\mathcal{Q} \subseteq \mathcal{R}_B^{(i)}$ and $\mathcal{Q}$ satisfies ϕ_2.

The same arguments also work if B is a cycle block but we should take care about the part that is in $S_B^{x,e}$ and the part that is in $S_B^{y,e}$ separately. □

We now have all the elements to compute the value $\lambda_B(e)$. We assume that B is a cycle block (resp. an edge block). If there is no $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,k,k)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(k)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2, or T^e does not satisfy the (e,k,k)-condition (resp. (k)-condition), then we set $\lambda_B(e) = \perp$. Otherwise, we aim at computing two integers i_0 and j_0 that are the smallest i and j in $[0,k]$ such that there exists $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i,j)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2 and such that T^e satisfies the (e,i,j)-condition (resp. (i)-condition). In order to

compute i_0 and j_0, we first fix j to be k and do a binary search on i, between 0 and k, to find the smallest value i_0 such that there exists $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i_0,k)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i_0)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2 and such that T^e satisfies the (e, i_0, k)-condition (resp. (i_0)-condition). We fix this value of i_0 and we do a second binary search, this time on j, between 0 and k, to find the smallest value j_0 such that there exists $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i_0,j_0)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i_0)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2 and such that T^e satisfies the (e, i_0, j_0)-condition (resp. (i_0)-condition). We fix this value of j_0 and we also fix $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i_0,j_0)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i_0)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2. We set $\lambda_B(e) = T^e_{\mathcal{Q}}$. Using Lemma 8.1 and Lemma 8.2, we know that the graph $\lambda_B(e)$ is a spanning tree of G_B that satisfies the (e, i_0, j_0)-condition (resp. (i_0)-condition). Moreover, using Lemma 8.3, we know that there is no spanning subtree of G_B that satisfies the (e, i_1, j_1)-condition (resp. (i_1)-condition) with $i_1 < i_0$ or $j_1 < j_0$ (resp. $i_1 < i_0$). This finishes the description of the algorithm.

Let us now discuss about the running time of the algorithm. At each step, given a cycle block (resp. an edge block) B and $e \in E(B)$, for each $i, j \in [0, k]$, we can check if T^e satisfies the (e, i, j)-condition (resp. the (i)-condition) in time $\mathcal{O}(n^2)$. Moreover, the number of elements in $\mathcal{R}_B$ is linear in n and for each $i, j \in [0, k]$, $\mathcal{R}_B^{(e,i,j)}$ can be computed in time $\mathcal{O}(n^2)$. As $\mathcal{R}_B$ contains at most $\mathcal{O}(n)$ elements, then the 2-Sat formulas ϕ_0, ϕ_1, and ϕ_2 contain at most $\mathcal{O}(n^2)$ clauses. We can check for each of the $\mathcal{O}(n^2)$ possible clauses if it is in ϕ_0, ϕ_1, or ϕ_2 in time $\mathcal{O}(n)$. Hence, we can compute ϕ_0, ϕ_1 and ϕ_2 in time $\mathcal{O}(n^3)$. As they contain at most $\mathcal{O}(n^2)$ clauses, we can solve them in time $\mathcal{O}(n^2)$. Since for each block B and each edge $e \in E(B)$, we perform at most two (independent) binary searches to find i_0 and j_0, we can compute $\lambda_B(e)$ in time $\mathcal{O}(n^3 \cdot \log k)$. Because there is a linear number of values $\lambda_B(e)$ to compute, we obtain an algorithm that solves Diameter-Tree* in time $\mathcal{O}(n^4 \cdot \log k)$. Using again a binary search on k between 0 and $2^{\lceil \log \mathbf{opt} \rceil}$, and the previous algorithm that solves Diameter-Tree* as a subroutine, we obtain an algorithm that solves Diameter-Tree in time $\mathcal{O}(n^4 \cdot (\log \mathbf{opt})^2)$ where **opt** is the diameter of the solution. □

8.5 FPT algorithm parameterized by $k + \mathsf{tw} + \Delta$

In this section we prove that the Diameter-Tree problem is FPT on general graphs parameterized by k, tw, and Δ. The proof is based on standard, but nontrivial, dynamic programming on graphs of bounded treewidth. It should be mentioned that we can assume that a tree-decomposition of the input graph G of width $\mathcal{O}(\mathsf{tw})$ is given together with the input. Indeed, by using for instance the algorithm of Bodlaender et al. [62], we can compute in time $2^{\mathcal{O}(\mathsf{tw})} \cdot n$ a tree-decomposition of G of width at most 5tw. Note that this running time is clearly dominated by the running time stated in Theorem 8.5. Recall also that, by [96, 184], it is possible, given a tree-decomposition to transform it in polynomial time to a new one $\mathcal{D}$ of the same width and construct a collection $\mathcal{G}$ such that the triple $(\mathcal{D}, r, \mathcal{G})$ is nice.

Theorem 8.5 *The* Diameter-Tree *problem can be solved in time* $(k^{\Delta \cdot \mathsf{tw}} \cdot \Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. *In particular, it is* FPT *parameterized by* k, tw, *and* Δ.

Proof: Before we start the description of the dynamic programming, we need some definitions. Let F be a forest and let S be a set of vertices in F that is good for G. We define $\mathsf{Reduce}(F, S)$ as the forest F' that is obtained from F by repetitively applying the following operations to vertices that are not in $N_F[S]$ as long as this is possible:

- removing a vertex of degree 1 and
- dissolving a vertex of degree 2.

Suppose now that $\mathsf{Reduce}(F, S) = F'$. We define the associated *reduce function* $\varphi : V(F) \to V(F') \cup E(F')$ as follows. For every vertex $z \in V(F)$, we define K_z to be the set of vertices x of $V(F')$ such that there exists a path in F from z to x that does not use any vertex of $V(F') \setminus \{x\}$. If K_z contains only one element x, then we define $\varphi(z) = x$, otherwise we define $\varphi(z) = K_z$. To show that φ is well-defined, we claim that $1 \leqslant |K_z| \leqslant 2$ and if $|K_z| = 2$ then $K_z \in E(F')$. Indeed, since each connected component of F contains an element of S, we have that $|K_z| \geqslant 1$. Assume that K_z contains two distinct vertices x_1 and x_2. By definition, we know that x_1 and x_2 are in the same connected component of F and also of F'. Let P_i be the path from z to x_i, $i \in \{1, 2\}$, in F and let P be the path from x_1 to x_2 in $F[V(P_1) \cup V(P_2)]$. By definition of x_1 and x_2, $V(P) \cap V(F') = \{x_1, x_2\}$. Moreover, since F is a forest, then P is the unique path from x_1 to x_2 in F. Let assume that $\{x_1, x_2\}$ is not an edge of F' and let x_3 be a vertex of F' on the path from x_1 to x_2 in F'. Then x_3 should be in P. This contradicts the fact that $V(P) \cap V(F') = \{x_1, x_2\}$. As F' is a forest, this also implies that $|K_z| \leqslant 2$.

We now proceed with the dynamic programming algorithm that solves Diameter-Tree*, the decision version of Diameter-Tree. Let (G, χ, c, k) be an instance of Diameter-Tree*. Consider a nice triple $(\mathcal{D}, r, \mathcal{G})$ where $\mathcal{D}$ is a tree-decomposition $D = (Y, \mathcal{X} = \{X_t \mid t \in V(Y)\})$ of G with width at most tw and $\mathcal{G} = \{G_t \mid t \in V(Y)\}$. For each $t \in V(Y)$ we set $w_t = |X_t|$ and $V_t = V(G_t)$. We also refer to the vertices of X_t as *t-terminals* and to the edges that are incident to vertices in X_t as *t-terminal edges*. We provide a table $\mathcal{R}_t$ that the dynamic programming algorithm computes for each node of $\mathcal{D}$. For this, we need first the notion of a *t-pair*, that is a pair (F, α) where:

- F is a forest such that
  1. X_t is good for F,
  2. $X_t \subseteq V(F)$,
  3. $N_F(X_t) \subseteq N_G(X_t)$,
  4. $|V(F) \setminus N_F[X_t]| \leqslant w_t - 2$, and
  5. $|\{e \in E(F) \mid e \cap X_t = \emptyset\}| \leqslant 2w_t - 3$,
- $\alpha : X_t \times X_t^F \to [0, k] \cup \{\perp\}$,

We call the vertices in $V(F) \setminus N_F[X_t]$ *external* vertices of F and the edges of $\{e \in E(F) \mid e \cap X_t = \emptyset\}$ *external* edges of F.

We need the function $\beta_t : \binom{\mathsf{adj}_G(X_t)}{2} \to [0,k] \cup \{\perp\}$ so that, for each $e_1, e_2 \in \mathsf{adj}_G(X_t)$, if there exists $x \in X_t$ such that $e_1 \cap e_2 = \{x\}$, then $\beta_t(e_1, e_2) = c(e_1, e_2)$, otherwise $\beta_t(e_1, e_2) = \perp$.

Let (F, α) be a t-pair. Recall that X_t^F contains all t-terminals and all non-t-terminal edges of F. Given a t-pair (F, α) as above we say that it is *admissible* if for every $(a, a') \in X_t^F \times X_t^F$ one of the following holds:

- there is no path between a and a' in F containing a vertex in X_t,
- one, say a, of a, a'' is a vertex in X_t and $\alpha(a, a') \leqslant k$,
- some internal vertex b of the path P between a and a' in F belongs in X_t and $\alpha_t(b, a) + \beta_t(e^-, e^+) + \alpha_t(b, a') \leqslant k$, where e^+, e^- are the two edges in P that are incident to b.

Intuitively, the admissibility of a t-pair (F, α) assures that the transferring cost, indicated by α, between any two external elements is bounded by k.

It is now time to give the precise definition of the table $\mathcal{R}_t$ of our dynamic programming algorithm. A pair (F, α) belongs in $\mathcal{R}_t$ if G contains a spanning tree $\hat{T}$ where $\mathsf{diam}(T) \leqslant k$ and the forest $\hat{F} = \hat{T}[V_t]$ (i.e. the restriction of $\hat{T}$ to the part of the graph that has been processed so far) satisfies the following properties:

- $\mathsf{Reduce}(\hat{F}, X_t) = F$, with the reduce function φ,
- for each $x \in X_t$ and $y \in X_t^F$, $\alpha(x, y) = \perp$ if and only if x and y are in two different connected components in F and if $\alpha(x, y) \neq \perp$, then for each $z \in \varphi^{-1}(y)$, $\mathtt{cost}_{\hat{F}}(x, z) \neq \perp$ and $\alpha(x, y) \geqslant \mathtt{cost}_{\hat{F}}(x, z)$.

Notice that each (F, α) as above is a t-pair. Indeed, Conditions 1–3 follow by the fact that $\hat{T}$ is a spanning tree of G and therefore $\hat{F}$ is a spanning forest of G_T. Conditions 4 and 5 follow by the fact that the internal vertices (resp. edges) of a tree with no vertices of degree 2 are at most two less than the number of its leaves (resp. at most twice the number of its leaves minus three). Moreover, the values of α are bounded by k because the diameter of $\hat{T}$ is at most k and therefore the same holds for all the connected components of $\hat{F}$. Notice that, for the same reason, all pairs in $\mathcal{R}_t$ must be admissible.

In the above definition, the external vertices and edges of F correspond to the parts of $\hat{F}$ that have been "compressed" during the reduction operation and the function α stores the transfer costs between those parts and the terminals. In this way, the trees in the t-pairs in $\mathcal{R}_t$ "represent" the restriction of all possible solutions in G_t. Moreover, the values of α indicate how these partial solutions interact with the t-terminals.

Our next concern is to bound the size of $\mathcal{R}_t$.

Claim 8.1 *For every $t \in V(Y)$, it holds that $|\mathcal{R}_t| \leqslant k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$.*

Proof: As we impose $N[X_t] \subseteq V(F)$, we have at most $2^{\Delta \cdot \mathsf{tw}}$ choices for the set $\{e \in E(F) \mid e \cap X_t \neq \emptyset\}$ and at most $(\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$ choices for the other edges or vertices. So the number of forest we take into consideration in $\mathcal{R}_t$ is at most $2^{\Delta \cdot \mathsf{tw}} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$. As the number of vertices and the number of edges of F is upper bounded by $\mathcal{O}(\Delta \cdot \mathsf{tw})$, the number of function α is at most $k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)}$. So $|\mathcal{R}_t| \leqslant k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$ and the claim holds. □

Clearly, (G, χ, c, k) is a YES-instance if and only if $\mathcal{R}_r \neq \emptyset$. We now proceed with the description of how to compute the set $\mathcal{R}_t$ for every node $t \in \mathcal{T}$. For this, we will assume inductively that, for every descendent t' of t, the set $\mathcal{R}_{t'}$ has already been computed. We distinguish several cases depending on the type of node t:

- If t is a *leaf node*. Then $G_t = \{\emptyset, \emptyset\}$ and $\mathcal{R}_t = \{((\emptyset, \emptyset), \varnothing)\}$.
- If t is an *vertex-introduce node*. Let v be the insertion vertex of X_t and let t' be the child of t. Then

$$R_t = \big\{\big((V(F') \cup \{v\}, E(F')), \alpha\big) \mid \exists (F', \alpha') \in R_{t'} : \alpha = \alpha' \cup \big\{\big((v, v), 0\big)\big\} \cup \big\{\big((v, a), \perp\big) \mid a \in X_t^{F'} \setminus \{v\}\big\}.$$

  Notice that at this point v is just an isolated vertex of G_t. This vertex is added in F and α is updated with the corresponding "void" transfer costs.

- If t is an *edge-introduce node*. Let $e = \{x, y\}$ be the insertion edge of X_t and let t' be the child of t. We define $F'' = (X_t, \{e\})$ and we set up $\alpha'' : X_t \times X_t^{F''} \to [0, k] \cup \{\perp\}$ (notice that $X_t^{F''} = X_t$) so that $\alpha''(x, y) = \alpha''(y, x) = 0$ and is $\perp$ for all other pairs of $X_t \times X_t$. Then

$$R_t = R_{t'} \cup \{(F, \alpha) \mid (F, \alpha) \text{ is admissible, } F \text{ is a forest, and there exists a pair } (F', \alpha') \in R_{t'} \text{ such that } F = F' \cup F'' \text{ and } \alpha = \alpha' \oplus_{\beta_t} \alpha''\}.$$

  In the above case, the single edge graph F'' is defined and the F of each new t-pair is its union with F'. Similarly, the function α'' encodes the trivial transfer costs in F''. Also, α is updated so to include the fusion of the transfer costs of α and α''.

- If t is an *forget node*. Let v be the forget vertex and let t' be the child of t. Then R_t contains every t-pair (F, α) such that there exists $(F', \alpha') \in \mathcal{R}_{t'}$ where:
  - if t is not the root of Y, then the connected component of F' containing v also contains an other element $v' \in X_t$ (this is necessary as X_t should always be good for F),
  - $F = \mathsf{Reduce}(F', X_t)$, with associated reduce function φ,
  - we denote by Z the set of every edge and every vertex that is in F' but not in F. Moreover, if $\varphi(v)$ is a vertex, then we further set $Z \leftarrow Z \cup \{\varphi(v)\}$. Notice also that if $z \in Z$, then $\varphi(z) = \varphi(v)$. Then $\alpha = \alpha'|_{X_t \times (X_t^F \setminus \{\varphi(v)\})} \cup \big\{\big((x, \varphi(v)), \max_{y \in Z} \alpha'(x, y)\big) \mid x \in X_t\big\}$.

Notice that F is further reduced because v has been "forgotten" in X_t. This may change the status of v as follows: either v is not any more in F or v is still in F but it is not a t-terminal. In the first case $\varphi(v)$ is either a vertex or an edge of F and in the second $\varphi(v) = v$. In any case we should update the values of $\alpha(x, \phi(v))$ for every $x \in X_t$ to the maximum transition cost (with respect to α') from x to some element of Z.

- If t is an *join node*. Let t' and t'' be the children of t. We define

$$R_t = R_{t'} \cup \{(F, \alpha) \mid (F, \alpha) \text{ is admissible, } F \text{ is a forest, and there exist two pairs } (F', \alpha') \in R_{t'} \text{ and } (F'', \alpha'') \in R_{t''} \text{ such that } F = F' \cup F'' \text{ and } \alpha = \alpha' \oplus_{\beta_t} \alpha''\}.$$

The above case is very similar to the case of the edge-introduce node. The only difference is that now F'' is now taken from $\mathcal{R}_{t''}$.

Taking into account Claim 8.1 on the bound of the size of $\mathcal{R}_t$, it is easy to verify that, in each of the above cases, R_t can be computed in $k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$ steps. So we can solve our problem in time $k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})} \cdot n$, and the theorem follows. □

8.6 Polynomially bounded costs

So far, we have completely characterized the parameterized complexity of the Diameter-Tree problem for any combination of the three parameters k, tw, and Δ. In this section we focus on the special case when the maximum cost value is polynomially bounded by n. The following corollary is an immediate consequence of Theorem 8.5.

Corollary 8.1 *If the maximum cost value is polynomially bounded by n, the* Diameter-Tree *problem is in* XP *parameterized by* tw *and* Δ.

From Corollary 8.1, a natural question is whether the Diameter-Tree problem is FPT or W[1]-hard parameterized by tw and Δ, in the case where the maximum cost value is polynomially bounded by n. The next theorem provides an answer to this question.

Theorem 8.6 *When the maximum cost value is polynomially bounded by n, the* Diameter-Tree *problem is* W[1]*-hard parameterized by* tw *and* Δ.

Proof: We present a parameterized reduction from the Bin Packing problem parameterized by the number of bins. In Bin Packing, we are given n integer item sizes $a_1, \ldots, a_n$ and an integer capacity B, and the objective is to partition the items into a minimum number of bins with capacity B. Jansen et al. [174] proved that Bin Packing is W[1]-hard parameterized by the number of bins in the solution, even when all item sizes are bounded by a polynomial of the input size. Equivalently, this version of the problem corresponds to the case where the item sizes are given in *unary* encoding; this is why it is called Unary Bin Packing in [174].

Given an instance $(\{a_1, a_2, \ldots, a_n\}, B, k)$ of Unary Bin Packing, where k is the number of bins in the solution and where we can assume that $k \geqslant 2$, we create an instance (G, χ, c) of Diameter-Tree as follows. The graph G contains a vertex r and, for $i \in [n]$ and $j \in [k]$, we add to G vertices v_i, ℓ_j^i, r_j^i and edges $\{r, \ell_j^1\}$, $\{v_i, \ell_j^i\}$, $\{v_i, r_j^i\}$, and $\{\ell_j^i, r_j^i\}$. Finally, for $i \in [n-1]$ and $j \in [k]$, we add the edge $\{r_j^i, \ell_j^{i+1}\}$. Let G' be the graph constructed so far; see Figure 8.6 for an illustration.

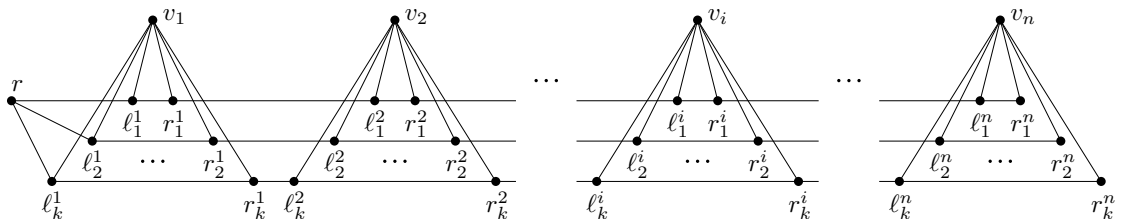

Figure 8.6: Graph G' built in the reduction of Theorem 8.6. Reload costs are not depicted.

Similarly to the proof of Theorem 8.3, we define G to be the graph obtained by taking two disjoint copies of G' and identifying vertex r of both copies. Note that G can be clearly built in polynomial time, and that $\mathsf{tw}(G) \leqslant k+1$ and $\Delta(G) = 2k$ (since we assume $k \geqslant 2$). Therefore, $\mathsf{tw}(G) + \Delta(G)$ is indeed bounded by a function of k, as required. (Again, the claimed bound on the treewidth can be easily seen by building a *path* decomposition of G with consecutive bags of the form $\{v_i, \ell_1^i, \ell_2^i, \ldots, \ell_k^i, r_1^i\}, \{v_i, \ell_1^i, \ell_2^i, \ldots, \ell_{k-1}^i, r_1^i, r_2^i\}, \{v_i, \ell_1^i, \ell_2^i, \ldots, \ell_{k-2}^i, r_1^i, r_2^i, r_3^i\}, \ldots.$)

Let us now define the coloring χ and the cost function c. Once more, for simplicity, we associate a distinct color with each edge of G, and thus it is enough to describe the cost function c for every pair of incident edges of G. The cost function is symmetric for both copies of G', so we just focus on one copy. For $i \in [n]$, let e_1, e_2 be two distinct edges containing vertex v_i. We set $c(e_1, e_2) = 2B + 1$ unless $e_1 = \{v_i, \ell_j^i\}$ and $e_2 = \{v_i, r_j^i\}$ for some $j \in [k]$, in which case we set $c(e_1, e_2) = a_i$. The cost associated with any other pair of edges of G is set to 0. Note that, as $(\{a_1, a_2, \ldots, a_n\}, B, k)$ is an instance of Unary Bin Packing, the reload costs of the instance (G, χ, c) of Diameter-Tree are polynomially bounded by $|V(G)|$.

We claim that $(\{a_1, a_2, \ldots, a_n\}, B, k)$ is a Yes-instance of Unary Bin Packing if and only if G has a spanning tree with diameter at most $2B$.

Assume first that $(\{a_1, a_2, \ldots, a_n\}, B, k)$ is a Yes-instance of Unary Bin Packing, and let $S_1, \ldots, S_k$ be the k subsets of $\{1, \ldots, n\}$ defining the k bins in the solution. We define a spanning tree T of G with $\mathsf{diam}(T) \leqslant 2B$ as follows. For each of the two copies of G', tree T contains, for $i \in [n-1]$ and $j \in [k]$, edges $\{r, \ell_j^1\}$ and $\{r_j^i, \ell_j^{i+1}\}$. For $i \in [n-1]$, if the item a_i belongs to the set S_j, we add to T the two edges $\{v_i, \ell_j^i\}$ and $\{v_i, r_j^i\}$; otherwise we add to T the edge $\{\ell_j^i, r_j^i\}$. Since the total item size of each bin in the solution of Unary Bin Packing is at most B, it can be easily checked that T is a spanning tree of G with $\mathsf{diam}(T) \leqslant 2B$.

Conversely, let T be a spanning tree of G with $\mathsf{diam}(T) \leqslant 2B$, and we proceed to define a

solution $S_1, \ldots, S_k$ of Unary Bin Packing. Let T_1 and T_2 be the restriction of T to the two copies of G'. By the choice of the reload costs and since $\mathsf{diam}(T) \leqslant 2B$, for every $i \in [n]$ and every $x \in \{1, 2\}$, tree T_x contains the two edges $\{v_i, \ell_j^i\}$ and $\{v_i, r_j^i\}$ for some $j \in [k]$, and none of the other edges incident with vertex v_i. Therefore, for every $x \in \{1, 2\}$, tree T_x consists of k paths sharing vertex r. This implies that $\mathsf{diam}(T) \geqslant \frac{1}{2}\mathsf{diam}(T_1) + \frac{1}{2}\mathsf{diam}(T_2)$, and since $\mathsf{diam}(T) \leqslant 2B$, it follows that there exists $x \in \{1, 2\}$ such that $\mathsf{diam}(T_x) \leqslant B$. Assume without loss of generality that $x = 1$, i.e., that $\mathsf{diam}(T_1) \leqslant B$. We define the bins $S_1, \ldots, S_k$ as follows. For every $i \in [n]$, if T_1 contains the two edges $\{v_i, \ell_j^i\}$ and $\{v_i, r_j^i\}$, we add item a_i to the bin S_j. Let us verify that this defines a solution of Unary Bin Packing. Indeed, assume for contradiction that for some $j \in [k]$, the total item size in bin S_j exceeds B. As bin S_j corresponds to one of the k paths in tree T_1, the diameter of this path would also exceed B, contradicting the fact that $\mathsf{diam}(T_1) \leqslant B$. The theorem follows. □

8.7 Concluding remarks

We provided an accurate picture of the (parameterized) complexity of the Diameter-Tree problem for any combination of the parameters k, tw, and Δ, distinguishing whether the reload costs are polynomial or not. Some questions still remain open. First of all, in the hardness result of Theorem 8.3, we already mentioned that the bound $\Delta \leqslant 3$ is tight, but the bound $\mathsf{tw} \leqslant 3$ might be improved to $\mathsf{tw} \leqslant 2$. A relevant question is whether the problem admits polynomial kernels parameterized by $k + \mathsf{tw} + \Delta$ (recall that it is FPT by Theorem 8.5). Theorem 8.6 motivates the following question: when all reload costs are bounded by a *constant*, is the Diameter-Tree problem FPT parameterized by $\mathsf{tw} + \Delta$? It also makes sense to consider the *color-degree* as a parameter (cf. [160]). Finally, we may consider other relevant width parameters, such as pathwidth (note that the hardness results of Theorems 8.1, 8.3, and 8.6 also hold for pathwidth), cliquewidth, treedepth, or tree-cutwidth.

Chapter 9

Further research

In a near future, I plan to continue working on parameterized complexity, with special emphasis on efficient FPT algorithms and kernelization. I would also like to pursue with the combinatorial problems on which I have been working recently.

At the end of chapters 5-6-7-8 a number of open problems related to the topic of each chapter have been already stated; see sections 5.7-6.7-7.4-8.7, respectively.

These are some other specific open problems or research avenues that I encountered during the projects in which I have been working in the last years, and that I would really like to solve, or at least to study in detail:

1. In [J6] we investigated the role of planarity in connectivity problems parameterized by treewidth. One notorious problem remains open: can Planar Disjoint Paths be solved in single-exponential time parameterized by treewidth? On general graphs, we know [228] that the problem can be solved in time $2^{\mathcal{O}(\mathsf{tw}\log\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, and that this running time is tight under the ETH [198]. Nevertheless, its time complexity on *planar* graphs is still unknown.

2. In [C27] we studied the complexity of the $\mathcal{F}$-Deletion problem parameterized by treewidth. Determining the tight complexity for all families $\mathcal{F}$ is an ambitious and fascinating research project. The following two questions may be easier to solve, but still quite challenging:
   - We do not know whether there exists some family $\mathcal{F}$ for which $\mathcal{F}$-Deletion *cannot* be solved, under the ETH, in time $2^{\mathrm{poly}(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. The recent results of Kociumaka and Pilipczuk [185] may shed some light in this direction, probably toward a negative answer.
   - On the other hand, the case where $\mathcal{F}$ contains a planar graph is extremely interesting in its own. We know by [C27] that, in that case, the problem can always be solved in time $2^{\mathcal{O}(\mathsf{tw}\log\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, and in some cases it can be solved in time $2^{\mathcal{O}(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. We suspect that this latter case holds if only if $\mathcal{F}$ contains one of the graphs P_2, P_3, P_4, C_3, C_4, or $K_{1,3}$. We are currently working on trying to prove this conjecture, which does not look like an easy task.

3. In one of my recent articles [C28], we studied how one relevant structural graph parameters, namely the *treedepth*, can be used to obtain polynomial kernels on dense graphs for classical graph problems such as Vertex Cover or Dominating Set. Two natural research directions appear from this work:

- Are there natural width parameters that allow for *meta-kernelization* results on dense graphs? Here, meta-kernelization refers to a result that guarantees the existence of particular types of kernels for a whole class of problems satisfying certain properties.
- In the particular case of treedepth, and as a follow-up of [C28], which problems admit polynomial kernels parameterized by the size of a *treedepth modulator*? Recall that a treedepth modulator of a graph is vertex set whose removal results in a graph of bounded treedepth.

4. The following question is simple to state, but probably difficult to answer, as it has been left as an open problem in [89, C34]: is the Minimum Colored Cut problem NP-complete? In this problem, we are given an edge-colored graph, and the objective is to find an edge cut using the minimum number of colors. Without colors, the problem is well-known to be solvable in polynomial time by the maximum flow-minimum cut duality [106], but its complexity in edge-colored graphs remains open.

5. Recall that a graph is (r, ℓ) if it admits an (r, ℓ)-partition, that is, a partition of its vertex set into r independent sets and ℓ cliques. Also, a graph is *well-covered* if every maximal independent set is also maximum. A graph is (r, ℓ)-*well-covered* if it is both (r, ℓ) and well-covered. In [C22] we managed to classify the complexity of deciding whether a graph given with an (r, ℓ)-partition is well-covered for all values of r and ℓ, except for $\ell = 0$ and $r \geqslant 3$. In particular, given a tripartite graph G together with a tripartition of $V(G)$, is it NP-complete to determine whether G is well-covered?

6. Switching to combinatorial questions, we proved in [C25] that the number $T_{n,k}$ of labeled graphs on n vertices and treewidth at most k satisfies

$$\left(c \cdot \frac{k\ 2^k n}{\log k}\right)^n 2^{-\frac{k(k+3)}{2}} k^{-2k-2} \leqslant T_{n,k} \leqslant \left(k\ 2^k n\right)^n 2^{-\frac{k(k+1)}{2}} k^{-k},$$

for $k > 1$ and some explicit absolute constant $c > 0$. Closing this gap remains open.

We strongly believe, but have not been able to prove yet, that there exist a constant d and a function f, with $k^{-2k-2} \leqslant f(k) \leqslant k^{-k}$ for every positive integer k, such that

$$T_{n,k} \geqslant (d \cdot k \cdot 2^k \cdot n)^n \cdot 2^{-\frac{k(k+1)}{2}} \cdot f(k).$$

7. Finally, concerning the Erdős-Pósa property for graph minors, we know that when H is a planar graph, the gap for the Erdős-Pósa property for H-models satisfies $f(k) = \mathcal{O}(k \cdot \log^{\mathcal{O}(1)} k)$ [80] and that, if H contains a cycle, it holds that $f(k) = \Omega(k \cdot \log k)$ [128]. We recently proved in [S35] that $f(k) = \Theta(k \cdot \log k)$ when H is a wheel, and we conjecture that this is the right gap for every planar graph H. Some new techniques and ideas seem to be needed in order to solve this conjecture.

Bibliography

Personal Bibliography

International Journals

[J1] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos. Faster parameterized algorithms for minor containment. *Theoretical Computer Science*, 412(50):7018–7028, 2011.

[J2] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos. Fast minor testing in planar graphs. *Algorithmica*, 64(1):69–84, 2012.

[J3] J. Baste, F. Beggas, H. Kheddouci, and I. Sau. On the parameterized complexity of the edge monitoring problem. *Information Processing Letters*, 121:39–44, 2017.

[J4] J. Baste, L. Faria, S. Klein, and I. Sau. Parameterized Complexity Dichotomy for (r, ℓ)-Vertex Deletion. *Theory of Computing Systems*, 61(3):777–794, 2017.

[J5] J. Baste, C. Paul, I. Sau, and C. Scornavacca. Efficient FPT algorithms for (strict) compatibility of unrooted phylogenetic trees. *Bulletin of Mathematical Biology*, 79(4):920–938, 2017.

[J6] J. Baste and I. Sau. The role of planarity in connectivity problems parameterized by treewidth. *Theoretical Computer Science*, 570:1–14, 2015.

[J7] N. Bousquet, D. Gonçalves, G. B. Mertzios, C. Paul, I. Sau, and S. Thomassé. Parameterized domination in circle graphs. *Theory of Computing Systems*, 54(1):45–72, 2014.

[J8] D. Chatzidimitriou, J. Raymond, I. Sau, and D. M. Thilikos. Minors in graphs of large θ_r-girth. *European Journal of Combinatorics*, 65:106–121, 2017.

[J9] N. Cohen, D. Gonçalves, E. J. Kim, C. Paul, I. Sau, D. M. Thilikos, and M. Weller. A polynomial-time algorithm for outerplanar diameter improvement. *Journal of Computer and System Sciences*, 89:315–327, 2017.

[J10] L. Faria, S. Klein, I. Sau, and R. Sucupira. Improved kernels for Signed Max Cut parameterized above lower bound on (r, ℓ)-graphs. *Discrete Mathematics & Theoretical Computer Science*, 19(1), 2017.

[J11] V. Garnero, C. Paul, I. Sau, and D. M. Thilikos. Explicit linear kernels via dynamic programming. *SIAM Journal on Discrete Mathematics*, 29(4):1864–1894, 2015.

[J12] V. Garnero, I. Sau, and D. M. Thilikos. A linear kernel for planar red-blue dominating set. *Discrete Applied Mathematics*, 217:536–547, 2017.

[J13] D. Gözüpek, S. Özkan, C. Paul, I. Sau, and M. Shalom. Parameterized complexity of the MINCCA problem on graphs of bounded decomposability. *Theoretical Computer*

Science, 690:91–103, 2017.

[J14] G. Joret, C. Paul, I. Sau, S. Saurabh, and S. Thomassé. Hitting and harvesting pumpkins. *SIAM Journal on Discrete Mathematics*, 28(3):1363–1390, 2014.

[J15] E. J. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, and S. Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Transactions on Algorithms*, 12(2):21:1–21:41, 2016.

[J16] E. J. Kim, C. Paul, I. Sau, and D. M. Thilikos. Parameterized algorithms for min-max multiway cut and list digraph homomorphism. *Journal of Computer and System Sciences*, 86:191–206, 2017.

[J17] G. B. Mertzios, I. Sau, M. Shalom, and S. Zaks. Placing regenerators in optical networks to satisfy multiple sets of requests. *IEEE/ACM Transactions on Networking*, 20(6):1870–1879, 2012.

[J18] L. P. Montejano and I. Sau. On the complexity of computing the k-restricted edge-connectivity of a graph. *Theoretical Computer Science*, 662:31–39, 2017.

[J19] J. Raymond, I. Sau, and D. M. Thilikos. An edge variant of the Erdős-Pósa property. *Discrete Mathematics*, 339(8):2027–2035, 2016.

[J20] J. Rué, I. Sau, and D. M. Thilikos. Asymptotic enumeration of non-crossing partitions on surfaces. *Discrete Mathematics*, 313(5-6):635–649, 2013.

[J21] J. Rué, I. Sau, and D. M. Thilikos. Dynamic programming for graphs on surfaces. *ACM Transactions on Algorithms*, 10(2):8:1–8:26, 2014.

International Conferences (not yet appeared in journal)

[C22] S. R. Alves, K. K. Dabrowski, L. Faria, S. Klein, I. Sau, and U. dos Santos Souza. On the (parameterized) complexity of recognizing well-covered (r, ℓ)-graphs. In *Proc. of the 10th International Conference on Combinatorial Optimization and Applications (COCOA)*, volume 10043 of *LNCS*, pages 423–437, 2016.

[C23] J. Araújo, J. Baste, and I. Sau. Ruling out FPT algorithms for Weighted Coloring on forests. In *Proc. of the IX Latin and American Algorithms, Graphs and Optimization Symposium (LAGOS)*, volume 62 of *ENDM*, pages 195–200, 2017.

[C24] J. Baste, D. Gözüpek, C. Paul, I. Sau, M. Shalom, and D. M. Thilikos. Parameterized complexity of finding a spanning tree with minimum reload cost diameter. In *Proc. of the 12th International Symposium on Parameterized and Exact Computation (IPEC)*, 2017. To appear in LIPIcs.

[C25] J. Baste, M. Noy, and I. Sau. On the number of labeled graphs of bounded treewidth. In *Proc. of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 10520 of *LNCS*, pages 88–99, 2017.

[C26] J. Baste, D. Rautenbach, and I. Sau. Uniquely restricted matchings and edge colorings. In *Proc. of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 10520 of *LNCS*, pages 100–112, 2017.

[C27] J. Baste, I. Sau, and D. M. Thilikos. Optimal algorithms for hitting (topological) minors on graphs of bounded treewidth. In *Proc. of the 12th International Symposium on Parameterized and Exact Computation (IPEC)*, 2017. To appear in LIPIcs.

[C28] M. Bougeret and I. Sau. How much does a treedepth modulator help to obtain polynomial kernels beyond sparse graphs? In *Proc. of the 12th International Symposium on Parameterized and Exact Computation (IPEC)*, 2017. To appear in LIPIcs.

[C29] D. Chatzidimitriou, J. Raymond, I. Sau, and D. M. Thilikos. An $O(\log \mathrm{OPT})$-approximation for covering/packing minor models of θ_r. In *Proc. of the 13th International Workshop on Approximation and Online Algorithms (WAOA)*, volume 9499 of *LNCS*, pages 122–132, 2015.

[C30] N. Cohen, F. Havet, D. Mazauric, I. Sau, and R. Watrigant. Complexity dichotomies for the Minimum $\mathcal{F}$-Overlay problem. In *Proc. of the of the 28th International Workshop on Combinatorial Algorithms (IWOCA)*, 2017. To appear in LNCS.

[C31] E. Kim, S. Oum, C. Paul, I. Sau, and D. M. Thilikos. An FPT 2-approximation for tree-cut decomposition. In *Proc. of the 13th International Workshop on Approximation and Online Algorithms (WAOA)*, volume 9499 of *LNCS*, pages 35–46, 2015.

[C32] H. Perret du Cray and I. Sau. Improved FPT algorithms for weighted independent set in bull-free graphs. In *Proc. of the 9th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 8894 of *LNCS*, pages 282–293, 2014.

[C33] J. Rué, I. Sau, and D. M. Thilikos. Dynamic Programming for H-minor-free Graphs (Extended Abstract). In *Proc. of the 18th Annual International Computing and Combinatorics Conference (COCOON)*, volume 7434 of *LNCS*, pages 86–97, 2012.

[C34] R. Sucupira, L. Faria, S. Klein, I. Sau, and U. S. Souza. Maximum cuts in edge-colored graphs. In *Proc. of the IX Latin and American Algorithms, Graphs and Optimization Symposium (LAGOS)*, volume 62 of *ENDM*, pages 87–92, 2017.

Submitted for Publication

[S35] P. Aboulker, S. Fiorini, T. Huynh, G. Joret, J.-F. Raymond, and I. Sau. A tight Erdős-Pósa function for wheel minors. *CoRR*, abs/1710.06282, 2017. Manuscript submitted for publication.

[S36] J. Araújo, V. A. Campos, A. K. Maia, I. Sau, and A. Silva. On the complexity of finding internally vertex-disjoint long directed paths. *CoRR*, abs/1706.09066, 2017. Manuscript submitted for publication.

[S37] V. Garnero, C. Paul, I. Sau, and D. M. Thilikos. Explicit linear kernels for packing problems. *CoRR*, abs/1610.06131, 2016. Manuscript submitted for publication.

[S38] V. Garnero and I. Sau. A linear kernel for planar total dominating set. *CoRR*, abs/1211.0978, 2012. Manuscript submitted for publication.

General Bibliography

[39] K. R. Abrahamson and M. R. Fellows. Finite automata, bounded treewidth, and well-quasiordering. In *Proc. of Graph Structure Theory, Contemporary Mathematics 147*, pages 539–564. American Mathematical Society, 1991.

[40] S. Agarwal and S. De. Dynamic spectrum access for energy-constrained CR: single channel versus switched multichannel. *IET Communications*, 10(7):761–769, 2016.

[41] J. Alber, M. Fellows, and R. Niedermeier. Polynomial-Time Data Reduction for Dominating Set. *Journal of the ACM*, 51(3):363–384, 2004.

[42] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.

[43] E. Amaldi, G. Galbiati, and F. Maffioli. On minimum reload cost paths, tours, and flows. *Networks*, 57(3):254–260, 2011.

[44] J. Araújo, N. Nisse, and S. Pérennes. Weighted coloring in trees. *SIAM Journal on Discrete Mathematics*, 28(4):2029–2041, 2014.

[45] S. Arkoulis, E. Anifantis, V. Karyotis, S. Papavassiliou, and N. Mitrou. On the optimal, fair and channel-aware cognitive radio network reconfiguration. *Computer Networks*, 57(8):1739–1757, 2013.

[46] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *Journal of the ACM*, 40(5):1134–1164, 1993.

[47] J. Battle, F. Harary, Y. Kadama, and J. Youngs. Additivity of the genus of a graph. *Bulletin of the American Mathematical Society*, 68:565–568, 1962.

[48] S. Bayhan and F. Alagoz. Scheduling in centralized cognitive radio networks for energy efficiency. *IEEE Transactions on Vehicular Technology*, 62(2):582–595, 2013.

[49] S. Bayhan, S. Eryigit, F. Alagoz, and T. Tugcu. Low complexity uplink schedulers for energy-efficient cognitive radio networks. *IEEE Wireless Communications Letters*, 2(3):363–366, 2013.

[50] L. W. Beineke and R. E. Pippert. The number of labeled k-dimensional trees. *Journal of Combinatorial Theory*, 6(2):200–205, 1969.

[51] A. Benhocine and A. P. Wojda. On the existence of specified cycles in a tournament. *Journal of Graph Theory*, 7(4):469–473, 1983.

[52] U. Bertelé and F. Brioschi. *Nonserial Dynamic Programming.* Academic Press, ISBN 0-12-093450-7, pp. 37–38, 1972.

[53] A. P. Bianzino, C. Chaudet, D. Rossi, and J.-L. Rougier. A survey of green networking research. *IEEE Communications Surveys & Tutorials*, 14(1):3–20, 2012.

[54] M. Bodirsky, O. Giménez, M. Kang, and M. Noy. Enumeration and limit laws for series-parallel graphs. *European Journal of Combinatorics*, 28(8):2091–2105, 2007.

[55] H. Bodlaender and T. Kloks. Only few graphs have bounded treewidth. Technical

Report RUU-CS-92-35, Utrecht University. Department of Computer Science, 1992.

[56] H. L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In *Proc. of the 15th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 317 of *LNCS*, pages 105–118, 1988.

[57] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.

[58] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[59] H. L. Bodlaender. Kernelization: New Upper and Lower Bound Techniques. In *Proc. of the 4th International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 5917 of *LNCS*, pages 17–37, 2009.

[60] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015.

[61] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.

[62] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.

[63] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) Kernelization. *Journal of the ACM*, 63(5):44:1–44:69, 2016.

[64] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014.

[65] H. L. Bodlaender and J. Nederlof. Subexponential time algorithms for finding small tree and path decompositions. In *Proc. of the 23rd Annual European Symposium on Algorithms (ESA)*, volume 9294 of *LNCS*, pages 179–190, 2015.

[66] H. L. Bodlaender, J. Nederlof, and T. C. van der Zanden. Subexponential Time Algorithms for Embedding H-Minor Free Graphs. In *Proc. of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55 of *LIPIcs*, pages 9:1–9:14, 2016.

[67] H. L. Bodlaender and E. Penninkx. A linear kernel for Planar Feedback Vertex Set. In *Proc. of the 3rd International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 5018 of *LNCS*, pages 160–171, 2008.

[68] H. L. Bodlaender, E. Penninkx, and R. B. Tan. A linear kernel for the k-Disjoint Cycle problem on planar graphs. In *Proc. of the 19th International Symposium on Algorithms and Computation (ISAAC)*, volume 5369 of *LNCS*, pages 306–317, 2008.

[69] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *Proc. of the 17th Annual European Symposium on Algorithms*

(ESA), volume 5757 of *LNCS*, pages 635–646, 2009.

[70] H. L. Bodlaender and B. van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Information and Computation*, 167(2):86–119, 2001.

[71] B. Bollobás and A. Thomason. Proof of a conjecture of Mader, Erdős and Hajnal on topological complete subgraphs. *European Journal of Combinatorics*, 19(8):883–887, 1998.

[72] R. B. Borie, R. G. Parker, and C. A. Tovey. Automatic Generation of Linear-Time Algorithms from Predicate Calculus Descriptions of Problems on Recursively Constructed Graph Families. *Algorithmica*, 7(1):555–581, 1992.

[73] A. Brandstädt. Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics*, 152(1-3):47–54, 1996.

[74] J. R. Büchi. On a Decision Method in Restricted Second-Order Arithmetic. In *Proc. of International Congress on Logic, Methodology, and Philosophy of Science*, pages 1–11, 1962.

[75] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58:171–176, 1996.

[76] L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997.

[77] Y. Cao. Linear recognition of almost interval graphs. In *Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1096–1115, 2016.

[78] Y. Cao and D. Marx. Interval Deletion Is Fixed-Parameter Tractable. *ACM Transactions on Algorithms*, 11(3):1–35, 2015.

[79] A. Celik and A. E. Kamal. Green cooperative spectrum sensing and scheduling in heterogeneous cognitive radio networks. *IEEE Transactions on Cognitive Communications and Networking*, 2(3):238–248, 2016.

[80] C. Chekuri and J. Chuzhoy. Large-treewidth graph decompositions and applications. In *Proc. of the 45th annual ACM Symposium on Theory of Computing*, pages 291–300, 2013.

[81] C. Chekuri and J. Chuzhoy. Polynomial bounds for the grid-minor theorem. *Journal of the ACM*, 63(5):40:1–40:65, 2016.

[82] J. Chen, H. Fernau, I. Kanj, and G. Xia. Parametric duality and kernelization: lower bounds and upper bounds on kernel size. *SIAM Journal on Computing*, 37:1077–1106, 2007.

[83] J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188–1198, 2008.

[84] J. Chen, I. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

[85] R. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, and M. Pilipczuk. Designing

FPT algorithms for cut problems using randomized contractions. *SIAM Journal on Computing*, 45(4):1171–1229, 2016.

[86] M. Chudnovsky. The structure of bull-free graphs I - Three-edge-paths with centers and anticenters. *Journal of Combinatorial Theory, Series B*, 102(1):233–251, 2012.

[87] M. Chudnovsky. The structure of bull-free graphs II and III - A summary. *Journal of Combinatorial Theory, Series B*, 102(1):252–282, 2012.

[88] N. Cohen, F. Havet, W. Lochet, and N. Nisse. Subdivisions of oriented cycles in digraphs with large chromatic number. *CoRR*, abs/1605.07762, 2016.

[89] D. Coudert, P. Datta, S. Pérennes, H. Rivano, and M. Voge. Shared risk resource group complexity and approximability issues. *Parallel Processing Letters*, 17(2):169–184, 2007.

[90] B. Courcelle. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Information and Computation*, 85(1):12–75, 1990.

[91] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language Theoretic Approach*. Number 138 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012.

[92] R. Crowston, G. Gutin, M. Jones, and G. Muciaccia. Maximum balanced subgraph problem parameterized above lower bound. *Theoretical Computer Science*, 513:53–64, 2013.

[93] R. Crowston, M. Jones, and M. Mnich. Max-Cut Parameterized Above the Edwards-Erdős Bound. *Algorithmica*, 72(3):734–757, 2015.

[94] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[95] M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Minimum bisection is fixed parameter tractable. In *Proc. of the 46th ACM Symposium on Theory of Computing (STOC)*, pages 323–332, 2014.

[96] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proc. of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 150–159, 2011.

[97] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. An improved FPT algorithm and a quadratic kernel for pathwidth one vertex deletion. *Algorithmica*, 64(1):170–188, 2012.

[98] A. Dawar, M. Grohe, and S. Kreutzer. Locally excluding a minor. In *Proc. of the 22nd IEEE Symposium on Logic in Computer Science (LICS)*, pages 270–279, 2007.

[99] B. de Fluiter. *Algorithms for Graphs of Small Treewidth*. PhD thesis, Utrecht University, 1997.

[100] F. K. H. A. Dehne, M. R. Fellows, M. A. Langston, F. A. Rosamond, and K. Stevens. An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. *The-*

ory of Computing Systems, 41(3):479–492, 2007.

[101] H. Dell and D. van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *Journal of the ACM*, 61(4):23:1–23:27, 2014.

[102] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. *ACM Transactions on Algorithms*, 1(1):33–47, 2005.

[103] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005.

[104] E. D. Demaine and M. Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008.

[105] C. Desset, N. Ahmed, and A. Dejonghe. Energy savings for wireless terminals through smart vertical handover. In *Proc. of IEEE International Conference on Communications*, pages 1–5, 2009.

[106] R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[107] M. J. Dinneen. Too many minor order obstructions. *Journal of Universal Computer Science*, 3(11):1199–1206, 1997.

[108] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and ids. In *Proc. of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5555 of *LNCS*, pages 378–389, 2009.

[109] D. Dong, X. Liao, Y. Liu, C. Shen, and X. Wang. Edge self-monitoring for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(3):514–527, 2011.

[110] F. Dorn. Planar subgraph isomorphism revisited. In *Proc. of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 5 of *LIPIcs*, pages 263–274, 2010.

[111] F. Dorn, F. V. Fomin, and D. M. Thilikos. Catalan structures and dynamic programming in H-minor-free graphs. *Journal of Computer and System Sciences*, 78(5):1606–1622, 2012.

[112] F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010.

[113] R. G. Downey and M. R. Fellows. Fixed-Parameter Tractability and Completeness I: Basic Results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[114] R. G. Downey and M. R. Fellows. Fixed-Parameter Tractability and Completeness II: On Completeness for W[1]. *Theoretical Computer Science*, 141(1&2):109–131, 1995.

[115] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[116] R. G. Downey, M. R. Fellows, and U. Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In *Proc. of a DIMACS Workshop*, volume 49 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 49–100, 1999.

[117] M. Drmota and E. Y. Jin. An asymptotic analysis of labeled and unlabeled k-trees. *Algorithmica*, 75(4):579–605, 2016.

[118] C. S. Edwards. Some extremal properties of bipartite subgraphs. *Canadian Journal of Mathematics*, 25:475–485, 1973.

[119] C. S. Edwards. An improved lower bound for the number of edges in a largest bipartite subgraph. *Recent Advances in Graph Theory*, pages 167–181, 1975.

[120] P. Erdős and L. Pósa. On independent circuits contained in a graph. *Canadian Journal of Mathematics*, 17:347–352, 1965.

[121] S. Eryigit, S. Bayhan, and T. Tugcu. Channel switching cost aware and energy-efficient cooperative sensing scheduling for cognitive radio networks. In *Proc. of IEEE International Conference on Communications (ICC)*, pages 2633–2638, 2013.

[122] U. Feige, M. Hajiaghayi, and J. R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008.

[123] M. R. Fellows. Surfing with rod. In *Computability and Complexity - Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, volume 10010 of *LNCS*, pages 9–18, 2017.

[124] M. R. Fellows and M. A. Langston. Nonconstructive tools for proving polynomial-time decidability. *Journal of the ACM*, 35:727–739, 1988.

[125] M. R. Fellows and M. A. Langston. An Analogue of the Myhill-Nerode Theorem and Its Use in Computing Finite-Basis Characterizations (Extended Abstract). In *Proc. of the 30th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 520–525, 1989.

[126] M. R. Fellows and M. A. Langston. On search, decision, and the efficiency of polynomial-time algorithms. *Journal of Computer and System Sciences*, 49(3):769–779, 1994.

[127] S. Fiorini and A. Herinckx. A tighter Erdős-Pósa function for long cycles. *Journal of Graph Theory*, 77(2):111–116, 2014.

[128] S. Fiorini, G. Joret, and D. R. Wood. Excluded forest minors and the Erdős-Pósa property. *Combinatorics, Probability and Computing*, 22(5):700–721, 2013.

[129] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[130] M. Flammini, A. Marchetti-Spaccamela, G. Monaco, L. Moscardelli, and S. Zaks.

On the complexity of the regenerator placement problem in optical networks. *IEEE/ACM Transactions on Networking*, 19(2):498–511, 2011.

[131] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. Springer, 2006.

[132] D. Foata. Enumerating k-trees. *Discrete Mathematics*, 1(2):181–186, 1971.

[133] F. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Linear kernels for (connected) dominating set on graphs with excluded topological subgraphs. In *Proc. of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 20 of *LIPIcs*, pages 92–103, 2013.

[134] F. V. Fomin, P. A. Golovach, and D. M. Thilikos. Contraction obstructions for treewidth. *Journal of Combinatorial Theory, Series B*, 101(5):302–314, 2011.

[135] F. V. Fomin, D. Lokshtanov, N. Misra, G. Philip, and S. Saurabh. Hitting forbidden minors: Approximation and kernelization. *SIAM Journal on Discrete Mathematics*, 30(1):383–410, 2016.

[136] F. V. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh. Nearly optimal FPT algorithms for Planar-$\mathcal{F}$-Deletion. Unpublished manuscript, 2011.

[137] F. V. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh. Planar $\mathcal{F}$-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *Proc. of the 53rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 470–479, 2012.

[138] F. V. Fomin, D. Lokshtanov, F. Panolan, and S. Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *Journal of the ACM*, 63(4):29:1–29:60, 2016.

[139] F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Bidimensionality and EPTAS. In *Proc. of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 748–759, 2011.

[140] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *Proc. of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 503–510, 2010.

[141] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Linear kernels for (connected) dominating set on H-minor-free graphs. In *Proc. of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 82–93, 2012.

[142] F. V. Fomin, S. Oum, and D. M. Thilikos. Rank-width and tree-width of H-minor-free graphs. *European Journal of Combinatorics*, 31(7):1617–1628, 2010.

[143] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *Journal of Computer and System Sciences*, 77(1):91–106, 2011.

[144] M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130(1–3):3–31, 2004.

[145] A. Gainer-Dewar. Γ-species and the enumeration of k-trees. *Electronic Journal of Combinatorics*, 19(4):P45, 2012.

[146] A. Gainer-Dewar and I. M. Gessel. Counting unlabeled k-trees. *Journal of Combinatorial Theory, Series A*, 126:177–193, 2014.

[147] J. Gajarský, P. Hlinený, J. Obdrzálek, S. Ordyniak, F. Reidl, P. Rossmanith, F. S. Villaamil, and S. Sikdar. Kernelization using structural parameters on sparse graph classes. *Journal of Computer and System Sciences*, 84:219–242, 2017.

[148] G. Galbiati. The complexity of a minimum reload cost diameter problem. *Discrete Applied Mathematics*, 156(18):3494–3497, 2008.

[149] G. Galbiati, S. Gualandi, and F. Maffioli. On minimum changeover cost arborescences. In *Proc. of the 10th International Symposium on Experimental Algorithms (SEA)*, volume 6630 of *LNCS*, pages 112–123, 2011.

[150] G. Galbiati, S. Gualandi, and F. Maffioli. On minimum reload cost cycle cover. *Discrete Applied Mathematics*, 164:112–120, 2014.

[151] I. Gamvros, L. Gouveia, and S. Raghavan. Reload cost trees and network design. *Networks*, 59(4):365–379, 2012.

[152] R. Ganian, F. Slivovsky, and S. Szeider. Meta-kernelization with structural parameters. *Journal of Computer and System Sciences*, 82(2):333–346, 2016.

[153] Y. Gao. Treewidth of Erdős-Rényi random graphs, random intersection graphs, and scale-free random graphs. *Discrete Applied Mathematics*, 160(4-5):566–578, 2012.

[154] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco, 1979.

[155] E. Ghosh, S. Kolay, M. Kumar, P. Misra, F. Panolan, A. Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015.

[156] A. C. Giannopoulou, B. M. P. Jansen, D. Lokshtanov, and S. Saurabh. Uniform kernelization complexity of hitting forbidden minors. *ACM Transaction on Algorithms*, 13(3):35:1–35:35, 2017.

[157] M. C. Golumbic, T. Hirst, and M. Lewenstein. Uniquely restricted matchings. *Algorithmica*, 31:139–154, 2001.

[158] L. Gourvès, A. Lyra, C. Martinhon, and J. Monnot. The minimum reload $s-t$ path, trail and walk problems. *Discrete Applied Mathematics*, 158(13):1404–1417, 2010.

[159] D. Gözüpek, S. Buhari, and F. Alagöz. A spectrum switching delay-aware scheduling algorithm for centralized cognitive radio networks. *IEEE Transactions on Mobile Computing*, 12(7):1270–1280, 2013.

[160] D. Gözüpek, H. Shachnai, M. Shalom, and S. Zaks. Constructing minimum changeover cost arborescenses in bounded treewidth graphs. *Theoretical Computer Science*, 621:22–36, 2016.

[161] D. Gözüpek and M. Shalom. Edge coloring with minimum reload/changeover costs. In *Proc. of the 13th Cologne Twente Workshop on Graphs and Combinatorial Optimization (CTW)*, pages 205–208, 2015.

[162] D. Gözüpek, M. Shalom, A. Voloshin, and S. Zaks. On the complexity of constructing minimum changeover cost arborescences. *Theoretical Computer Science*, 540:40–52, 2014.

[163] M. Grohe and D. Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *SIAM Journal on Computing*, 44(1):114–159, 2015.

[164] D. J. Guan and X. Zhu. A coloring problem for weighted graphs. *Information Processing Letters*, 61(2):77–81, 1997.

[165] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computing and System Sciences*, 72(8):1386–1396, 2006.

[166] J. Guo and R. Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In *Proc. of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *LNCS*, pages 375–386, 2007.

[167] J. Guo, R. Niedermeier, and S. Wernicke. Fixed-parameter tractability results for full-degree spanning tree and its dual. *Networks*, 56(2):116–130, 2010.

[168] R. Halin. s-functions for graphs. *Journal of Geometry*, 8:171–186, 1976.

[169] I. V. Hicks. Branch decompositions and minor containment. *Networks*, 43(1):1–9, 2004.

[170] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[171] B. M. P. Jansen and H. L. Bodlaender. Vertex cover kernelization revisited - upper and lower bounds for a refined parameter. *Theory of Computing Systems*, 53(2):263–299, 2013.

[172] B. M. P. Jansen, D. Lokshtanov, and S. Saurabh. A near-optimal planarization algorithm. In *Proc. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1802–1811, 2014.

[173] B. M. P. Jansen and J. J. H. M. Wulms. Lower bounds for protrusion replacement by counting equivalence classes. In *Proc. of the 11th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 63 of *LIPIcs*, pages 17:1–17:12, 2016.

[174] K. Jansen, S. Kratsch, D. Marx, and I. Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.

[175] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[176] I. A. Kanj, M. J. Pelsmajer, M. Schaefer, and G. Xia. On the Induced Matching problem. *Journal of Computer and System Sciences*, 77(6):1058–1070, 2011.

[177] K. Kawarabayashi. Planarity allowing few error vertices in linear time. In *Proc. of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 639–648, 2009.

[178] K. Kawarabayashi and Y. Kobayashi. Linear min-max relation between the treewidth

of H-minor-free graphs and its largest grid. In *Proc. of the 29th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 14 of *LIPIcs*, pages 278–289, 2012.

[179] K. Kawarabayashi, Y. Kobayashi, and B. A. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012.

[180] K. Kawarabayashi and M. Thorup. The Minimum k-way Cut of Bounded Size is Fixed-Parameter Tractable. In *Proc. of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–169, 2011.

[181] J. M. Keil. The complexity of domination problems in circle graphs. *Discrete Applied Mathematics*, 42(1):51–63, 1993.

[182] E. J. Kim, C. Paul, and G. Philip. A single-exponential FPT algorithm for the K_4-minor cover problem. *Journal of Computer and System Sciences*, 81(1):186–207, 2015.

[183] R. Kim, S.-J. Kim, J. Ma, and B. Park. Cycles with two blocks in k-chromatic digraphs. *CoRR*, abs/1610.05839, 2016.

[184] T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994.

[185] T. Kociumaka and M. Pilipczuk. Deleting vertices to graphs of bounded genus. *CoRR*, abs/1706.04065, 2017.

[186] J. Komlós and E. Szemerédi. Topological cliques in graphs II. *Combinatorics, Probability and Computing*, 5(01):79–90, 1996.

[187] V. R. Konda and T. Y. Chow. Algorithm for traffic grooming in optical networks to minimize the number of transceivers. In *Proc. of IEEE Workshop on High Performance Switching and Routing*, pages 218–221, 2001.

[188] S. Kreutzer. Algorithmic meta-theorems. *Electronic Colloquium on Computational Complexity*, 16:147, 2009.

[189] S. Kreutzer and S. Tazari. On brambles, grid-like minors, and parameterized intractability of monadic second-order logic. In *Proc. of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 354–364, 2010.

[190] D. Kühn and D. Osthus. Minors in graphs of large girth. *Random Structutres & Algorithms*, 22(2):213–225, 2003.

[191] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Matematicae*, 15(1):271–283, 1930.

[192] S. J. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *The Journal of the Royal Statistical Society. Series B (Methodological)*, 50:157–224, 1988.

[193] A. Leaf and P. D. Seymour. Tree-width and planar minors. *Journal of Combinatorial Theory, Series B*, 111:38–53, 2015.

[194] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties

is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[195] D. Lokshtanov. Wheel-free deletion is $W[2]$-hard. In *Proc. of the 3rd International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 141–147, 2008.

[196] D. Lokshtanov, D. Marx, and S. Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. In *Proc. of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 777–789, 2011.

[197] D. Lokshtanov, D. Marx, and S. Saurabh. Slightly superexponential parameterized problems. In *Proc. of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 760–776, 2011.

[198] D. Lokshtanov, D. Marx, S. Saurabh, et al. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, (105):41–72, 2011.

[199] D. Lokshtanov, M. Mnich, and S. Saurabh. A linear kernel for planar connected dominating set. *Theoretical Computer Science*, 23(412):2536–2543, 2011.

[200] D. Lokshtanov, M. Mnich, and S. Saurabh. A linear kernel for Planar Connected Dominating Set. *Theoretical Computer Science*, 412(23):2536–2543, 2011.

[201] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms*, 11(2):15, 2014.

[202] D. Lokshtanov, S. Saurabh, and S. Sikdar. Simpler parameterized algorithm for OCT. In *Proc. of the 20th International Workshop on Combinatorial Algorithms (IWOCA)*, volume 5874 of *LNCS*, pages 380–384, 2009.

[203] D. Marx. Chordal Deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010.

[204] D. Marx, B. O'sullivan, and I. Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30:1–30:35, 2013.

[205] D. Marx and I. Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3–4):807–822, 2012.

[206] S. Mishra. On the maximum uniquely restricted matching for bipartite graphs. *Electronic Notes in Discrete Mathematics*, 37:345–350, 2011.

[207] D. Mitsche and G. Perarnau. On the treewidth and related parameters of random geometric graphs. In *Proc. of the 29th International Symposium on Theoretical Aspects of Computer Science, (STACS)*, volume 14 of *LIPIcs*, pages 408–419, 2012.

[208] B. Mohar. An obstruction to embedding graphs on surfaces. *Discrete Mathematics*, 78(1–2):135–142, 1989.

[209] J. W. Moon. The number of labeled k-trees. *Journal of Combinatorial Theory*, 6(2):196–199, 1969.

[210] H. Moser and S. Sikdar. The parameterized complexity of the Induced Matching problem. *Discrete Applied Mathematics*, 157(4):715–727, 2009.

[211] A. Nerode. Linear Automaton Transformations. *Proceedings of the American Mathematical Society*, 9:541–544, 1958.

[212] J. Nesetril and P. Ossona De Mendez. *Sparsity: Graphs, Structures, and Algorithms.* Springer, 2012.

[213] J. Nešetřil and P. O. de Mendez. On nowhere dense graphs. *European Journal of Combinatorics*, 32(4):600–617, 2011.

[214] R. Niedermeier. *Invitation to fixed parameter algorithms*, volume 31. Oxford University Press, 2006.

[215] J. Nešětril and P. Ossona de Mendez. Grad and classes with bounded expansion I. Decompositions. *European Journal of Combinatorics*, 29(3), 2008.

[216] D. Osthus, H. J. Prömel, and A. Taraz. On random planar graphs, the number of planar graphs and their triangulations. *Journal of Combinatorial Theory, Series B*, 88(1):119–134, 2003.

[217] G. Philip, V. Raman, and Y. Villanger. A quartic kernel for Pathwidth-One Vertex Deletion. In *Proc. of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 6410 of *LNCS*, pages 196–207, 2010.

[218] M. Pilipczuk. A tight lower bound for vertex planarization on graphs of bounded treewidth. *Discrete Applied Mathematics*, 231:211–216, 2017.

[219] A. Rafiey. Single Exponential FPT Algorithm for Interval Vertex Deletion and Interval Completion Problem. *CoRR*, abs/1211.4629, 2012.

[220] A. Rai, M. S. Ramanujan, and S. Saurabh. A parameterized algorithm for mixed-cut. In *Proc. of the 12th Latin American Symposium on Theoretical Informatics (LATIN)*, volume 9644 of *LNCS*, pages 672–685, 2016.

[221] B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[222] N. Robertson and P. D. Seymour. Graph Minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[223] N. Robertson and P. D. Seymour. Graph Minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41:92–114, 1986.

[224] N. Robertson and P. D. Seymour. Graph Minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52:153–190, 1991.

[225] N. Robertson and P. D. Seymour. Graph Minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1994.

[226] N. Robertson and P. D. Seymour. Graph Minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63:65–110, 1995.

[227] N. Robertson and P. D. Seymour. Graph Minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92:325–357, 2004.

[228] P. Scheffler. A practical linear time algorithm for disjoint paths in graphs with

bounded tree-width. Fachbereich 3 Mathematik, Tech. Report 396/1994, FU Berlin, 1994.

[229] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[230] N. Shami and M. Rasti. A joint multi-channel assignment and power control scheme for energy efficiency in cognitive radio networks. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2016.

[231] L. Takács. On the number of distinct forests. *SIAM Journal on Discrete Mathematics*, 3(4):574–581, 1990.

[232] A. Takahashi, S. Ueno, and Y. Kajitani. Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Discrete Mathematics*, 127(1-3):293–304, 1994.

[233] A. Takahashi, S. Ueno, and Y. Kajitani. Mixed searching and proper-path-width. *Theoretical Computer Science*, 137(2):253–268, 1995.

[234] A. Thomason. The extremal function for complete minors. *Journal of Combinatorial Theory, Series B*, 81(2):318–338, 2001.

[235] S. Thomassé, N. Trotignon, and K. Vuskovic. A polynomial Turing-kernel for weighted independent set in bull-free graphs. *Algorithmica*, 77(3):619–641, 2017.

[236] C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8:85–89, 1984.

[237] B. van Antwerpen-de Fluiter. *Algorithms for graphs of small treewidth.* PhD thesis, Utrecht University, 1997.

[238] R. van Bevern, R. G. Downey, M. R. Fellows, S. Gaspers, and F. A. Rosamond. Myhill-nerode methods for hypergraphs. *Algorithmica*, 73(4):696–729, 2015.

[239] V. V. Vazirani. *Approximation algorithms.* Springer, 2001.

[240] K. Wagner. Graphentheorie. In *B. J. Hochschultaschenbucher, Mannheim*, volume 248/248a, 1970.

[241] H.-C. Wirth and J. Steffan. Reload cost problems: minimum diameter spanning tree. *Discrete Applied Mathematics*, 113(1):73–85, 2001.

[242] P. Wollan. The structure of graphs not admitting a fixed immersion. *Journal of Combinatorial Theory Series B*, 110:47–66, 2015.

[243] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM Journal of Applied Mathematics*, 38(3):364–372, 1980.

[244] C.-K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983.