Structural Graph Parameters, Fine-Grained Complexity, and Approximation

Michael Lampis

Université Paris Dauphine
PSL★

Mémoire intitulé

Structural Graph Parameters, Fine-Grained Complexity, and Approximation

présenté par

Michail Lampis

pour l'obtention du diplôme

Habilitation à Diriger des Recherches

Coordinateur:	Vangelis Th. Paschos Professeur, Université Paris-Dauphine
Rapporteurs:	Michael Fellows Professeur, Université de Bergen
	Christophe Paul Directeur de Recherche, CNRS LIRMM
	Ioan Todinca Professeur, Université d'Orléans
Examinateurs:	Éric Colin de Verdière Directeur de Recherche, CNRS LIGM
	Bruno Escoffier Professeur, Sorbonne Université

Abstract

In this work we summarize much of the research conducted by the author since his PhD defense in the area of structural parameterizations of intractable graph problems. The general objective of this research domain is the detailed investigation of the computational complexity of NP-hard graph problems using parameters that measure the structure of the given graph. This area has attracted much interest in recent years because most interesting graph problems are NP-hard, therefore there is a great need to better understand and deal with their complexity.

Our exposition begins with a summary of the scientific context and a survey of the main parameters that we use to measure graph structure, including treewidth, which is the most well-studied such parameter, and its variations. We then move on to present concrete results regarding the complexity of well-known graph problems measured as a function of such structural parameters, separating our exposition into three parts:

1. In the first part, we use the tools of parameterized complexity to understand the trade-offs between different parameters. In particular, we are interested to know which problems which are fixed-parameter tractable (FPT) for one parameter are intractable when we use a more general parameter. Among the highlights of this chapter are the discovery of the first natural problem that separates treewidth from pathwidth (Grundy Coloring); the proof that SAT remains intractable in the very restricted case where we parameterize by neighborhood diversity; and the proof that natural generalizations of Dominating Set and Independent Set are already hard parameterized by vertex cover.

2. In the second part we take another look at the same problems using the tools of fine-grained complexity theory, notably the Exponential Time Hypothesis (ETH) and its Strong variant (SETH). This allows us to obtain much more detailed estimates of the complexity of various problems. Among the main results we present are the fact that $\exists\forall$-SAT has a double-exponential complexity parameterized by treewidth under the ETH; and exact bounds on the complexity of k-Coloring and Dominating Set parameterized by clique-width, under the SETH ($(2^k - 2)^{\mathrm{cw}}$ and 4^{cw} respectively).

3. In the last part we consider how the notion of approximation interacts with structural parameters and present a general technique for obtaining FPT approximation algorithms for problems which are intractable parameterized by treewidth, as well as several example applications.

We conclude the document with directions for further work as well as a list of concrete open problems.

Résumé

Le but de cette thèse est de résumer une grande partie des travaux de recherche menés par l'auteur dans le domaine des paramétrisations structurelles de problèmes NP-difficiles dans les graphes depuis l'obtention de son doctorat. L'objectif de ce domaine de recherche est l'investigation détaillée de la complexité computationnelle de problèmes NP-difficiles dans les graphes en utilisant des paramètres qui mesurent la structure d'un graphe donné. Ce domaine a récemment attiré l'intérêt de la communauté scientifique car la grande majorité de problèmes algorithmiques dans les graphes sont NP-difficiles et par conséquent il y a un fort besoin de mieux comprendre et de faire face à leur complexité.

Notre exposition commence avec un résumé du contexte scientifique et un aperçu général des paramètres qu'on va considérer, dont notamment la largeur arborescente (treewidth), qui est le paramètre le plus étudié, et ses variations. On va alors présenter des résultats concrets concernant la complexité de plusieurs problèmes bien-connus mesurée en tant que fonction de tels paramètres structurels. Notre exposition est divisée en trois parties :

1. Dans la première partie on utilise les outils de la complexité paramétrée afin de mieux comprendre les avantages et désavantages de chaque paramètre. Plus précisément, nous allons nous intéresser à trouver quels problèmes qui sont "fixed-parameter tractable" (solubles à paramètre fixé – FPT) pour un paramètre ne sont plus solubles quand on utilise un paramètre plus général. Parmi les résultats les plus marquants de ce chapitre on a la découverte du premier problème qui peut distinguer la largeur arborescente de la largeur de chemin (pathwidth), à savoir Grundy Coloring ; la preuve que SAT est difficile quand on utilise le paramètre très restreint de la diversité de voisinage (neighborhood diversity) ; et la preuve que des généralisations naturelles des problèmes de l'ensemble dominant minimum (Dominating Set) et du stable maximum (Independent Set) sont difficiles paramétrées par la couverture par sommets (vertex cover).

2. Dans la deuxième partie on se penche sur le même type de problème mais cette fois en utilisant les outils de la complexité fine et notamment l'hypothèse du temps exponentiel (ETH) et sa variation forte (SETH). Cela nous permet d'obtenir des estimations beaucoup plus précises sur la complexité de divers problèmes. Parmi les résultats présentés sont le fait que la complexité de $\exists\forall$-SAT est doublement exponentielle quand le problème est paramétré par treewidth, sous la ETH ; et des bornes exactes sur la complexité de k-Coloration et de Dominating Set paramétrés par clique-width, sous la SETH ($(2^k-2)^{\mathrm{cw}}$ et 4^{cw} respectivement).

3. Dans la troisième partie, on considère comment les algorithmes d'approximation interagissent avec les notions des paramètres structurels. On présente une technique générale qui nous permet d'obtenir des algorithmes d'approximation FPT pour des problèmes qui sont difficiles paramétrés par treewidth, ainsi que plusieurs exemples d'applications de cette méthode.

Pour conclure, on présente quelques perspectives de recherche ainsi qu'une liste de problèmes ouverts.

Acknowledgments

First of all, I would like to thank all the members of my examination committee: Vangelis Paschos, who has been a good friend and close collaborator since I arrived in LAMSADE in 2014; the three rapporteurs Mike Fellows, Christophe Paul, and Ioan Todinca, for taking the time to carefully read my manuscript and for their kind evaluation; and Éric Colin de Verdière and Bruno Escoffier for accepting to enrich the jury with their expertise. Among them, I would like to reserve a special thank you to Mike Fellows, who has always been extremely generous and encouraging with me since we first met in 2008. Our random meeting in ISAAC 2008, where Mike put me in touch with novel research ideas that I could at the time barely understand, really changed the trajectory of my research and convinced me to work on the topics I have been working on ever since.

The work presented here was mostly done since I joined LAMSADE, where I had the good fortune to be part of a great research environment with many good collaborators – this work would not have been possible without their help. I would first like to thank Eunjung Kim and Florian Sikora, my closest collaborators in the lab and my partners in numerous failed (and a few successful) research proposals. I would also like to thank Cristina Bazgan, Laurent Gourvès (also for being a great office-mate!), Ararat Harutyunyan, Jérôme Lang, and also reserve a thought for our late friend Jérôme Monnot. Many thanks also to the "younger" members of our lab that I have had the pleasure of collaborating with: Ioannis Katsikarelis and Louis Dublois, whose theses I co-supervised with Vangelis; Édouard Bonnet (who has by now become an accomplished CNRS researcher), Mehdi Khosravian Ghadikolaei, and Nikolaos Melissinos.

I am also very grateful to Kazuhisa Makino, who was my post-doc supervisor at Kyoto University before I joined LAMSADE. Kaz was extremely generous with me during my stay in Japan, which was a very positive and memorable experience. My post-doc with him made me love Japan and I have made a sustained effort to maintain my ties with this country ever since. For that, I would like to thank Yota Otachi, who has been my partner in several fruitful bilateral research projects, as well as the members of his Japanese team: Rémy Belmonte (whom I could also have thanked in the previous paragraph, since he has now joined LAMSADE!), Hirotaka Ono, Tesshu Hanaka, and Masashi Kiyomi. Our mutual visits in the last few years have been among the highlights of my research life.

Finally, the warmest and most important thank you goes of course to the one person who has been my partner in research and in life for the past fifteen years and counting. Her presence is what has made these years fun and worthwhile. Thank you Valia, and I look forward to our new life together with our little one!

Contents

Chapter 1

Introduction

The aim of this document is to summarize a large part of the research conducted by the author since his PhD defense in 2011 and in particular the work done since joining LAMSADE – Université Paris Dauphine in September 2014. The general area which this research falls into is computational complexity theory and more specifically parameterized complexity. In a few words, the main motivating question behind this research is "what is computational intractability and how can we deal with it?". In a sense, this is the type of question that animates the whole field of computational complexity, which is interested in understanding how hard it is to solve instances of various computational problems.

The work we will present focuses on computational problems where the input is described using graphs and makes heavy use of one key insight that is as simple as it is pervasive, namely the observation that the *structure* of the input, as can be measured by various graph parameters, must have a serious impact on the computational complexity of even the most intractable computational problems. As a result, attempting to measure the complexity of a problem in terms of only the input *size*, without taking into account structural information, is bound to lead to disappointing results as it ignores too much useful information. In other words, the thesis of this work is that in order to do computational complexity the right way, we have to include in our analysis additional parameters that measure the complexity of the input, and since we will be talking about graphs, this means that the "right" way to do computational complexity must involve some structural graph parameters. We will therefore present research work whose unifying theme is exactly the description of the complexity of various graph problems in terms of structural graph parameters, such as treewidth and its many variations. This idea is not new – it is in fact a founding principle behind the whole field of parameterized complexity – and in this document we will see a few examples of how this principle has progressed recently and how it has been applied in a fruitful way to several important problems.

The goal of this document is to highlight recent works involving the author that have attempted (sometimes with success!) to describe the complexity of various computational problems on graphs as a function of the structure of the input. Our style will be more expository rather than technical: we will focus more on describing ideas and their scientific context than on providing full proofs, except where the proofs themselves may add to the reader's intuition. Nevertheless, we will in all cases give pointers to peer-reviewed publications that contain the full details of all claimed results. Before we go on to present the results that will form the core of this document, we will devote the rest of this chapter to a brief overview of the broader scientific context, including a (biased) summary of the history of the evolution of computational complexity in the last 60 years. This will help set the stage for what is to come.

Motivation and Scientific Context

The early days. Before we begin, let us take several steps back and attempt to summarize 60 years of developments in computational complexity theory in a way that explains how we arrive at

the present work. Computational complexity was founded as a sub-field of theoretical computer science in the 1960s with the goal of studying and comparing the "hardness" of solving various computational problems. The key principles, which are typically attributed to Hartmanis and Stearns [106], are as follows: the complexity of a problem is measured asymptotically as a function of n, the size of the input, and is defined as the *worst-case* complexity of the best algorithm solving the problem. Around the same time Edmonds [67] was among the first to propose that a problem should be considered tractable if it admits a polynomial-time algorithm, that is, an algorithm that solves every input of size n in time at most n^c, for some constant c. These definitions played a hugely influential role and form the bedrock of computational complexity theory to this day.

Very quickly, however, a key weakness of this framework was revealed: very few interesting problems turn out to be tractable using this definition. This intuition was confirmed by the pioneering (and Turing award-winning) work of Cook [43] and later Karp [118], which identified the key notion of NP-completeness. Although stopping short of conclusively proving that many interesting problems are intractable, this work and the large body of work that followed established that the vast majority of interesting computational problems are very unlikely to admit polynomial-time algorithms. Indeed, one could argue that one of the major advances of the field of computational complexity in the 1970s was the realization that Edmonds' definition of tractability is too optimistic and likely unachievable for most problems. This was nicely captured in Garey&Johnson's classic textbook [100] which gathered hundreds of concrete examples of problems which had been shown to be NP-complete. Although the P=NP question was not resolved (and remains open to this day) this led to a consensus in the community that polynomial-time algorithms were to be seen as the exception rather than the rule in combinatorial optimization.

This state of affairs understandably led to a minor existential crisis for the field. Clearly, if the field of computational complexity had nothing much more useful to say than "everything is hard", that would have been not only quite disappointing, but also rather contrary to real-life expectations as many problems were turning out to be more and more tractable in practice. If the definition of tractability was so hopelessly out of reach, that must have meant that the definition was flawed and some of its assumptions would need to be revisited. This led to the development of several different, sometimes competing and sometimes intersecting, lines of research.

A first solution: Restricted Inputs. One of the first possible ways to tackle this conundrum was already advocated in Garey&Johnson's textbook and consisted of considering restricted classes of inputs for the problem at hand. For example, if we are considering graph problems, the idea was that perhaps the reason that a problem turns out to be NP-hard in general is that it is possible to construct a pathological class of complicated graphs which are general enough to encode SAT. Nevertheless, it could still be hoped that such pathological instances would rarely come up in practice, and hence it makes sense to study the complexity of the problem under the promise that the input instance would be "reasonable". This gave a strong motivation to study various structured classes of graphs and examine whether standard problems become tractable when the input graph is restricted to belong to such a class. Some of the most famous such classes are acyclic graphs, planar graphs, chordal, split, and bipartite graphs to name just a few among many[1].

This line of research was quite successful and continues until today. It can be seen as a direct ancestor of the work we present here, as its key idea is in a sense that the structure of the input must be taken into account in order to determine the complexity of a problem. One drawback – and one way in which the structural parameterized complexity viewpoint we advocate for in this work is arguably superior to the study of graph classes – is that this framework is quite binary: a graph either is planar/chordal/bipartite or it is not. Hence, even if we know, say, that Dominating Set is polynomial-time solvable on interval graphs, this is of little help if the graph at hand is not an interval graph, even if we can hope that the input will somehow be *close* to being an interval graph. In other words, the positive results produced by this line of research immediately break down the moment we set foot outside of the prescribed area of inputs. As we

[1] See `https://www.graphclasses.org/` for extensive information about numerous graph classes and the complexity of standard problems restricted to them.

will see, one of the advantages of the structurally parameterized approach is exactly its flexibility that allows us to avoid such barriers.

More broadly, even if we put this lack of flexibility aside, this line of research arguably never reached the point where it could be considered a satisfactory answer to the fundamental question of dealing with NP-hardness, for the rather pedestrian reason that most of the results it produced remained rather negative. In particular, some of the most persuasive arguments to adopt this approach in the context of graph problems are that real-life graphs have low degree and often possess a simple geometric structure (planarity or low genus), hence it makes sense to study our problems restricted to planar graphs or bounded-degree graphs. Unfortunately, the vast majority of interesting problems remain NP-hard under such restrictions, so the study of restricted inputs only produces positive results if one imposes further restrictions which may or may not be realistic in practice. Therefore, even though this line of research was quite successful in identifying key structural properties of algorithmic interest and leveraging them to obtain positive results, today we would regard it as a basis and a first step towards a more complete incorporation of the notion of input structure in computational complexity theory.

Approximation: does it make things easier? Starting in the 1980s, another approach was widely considered, especially in the context of optimization problem, as a solution to the conundrum of NP-hardness. It was observed that the basic NP-completeness theory developed in the 1970s only showed that determining the *precise* optimal value of most optimization problems was hard. It could therefore still be hoped that the root of intractability was the demand for absolute precision and that one could hope that, even for an NP-hard optimization problem, it would be possible to obtain in polynomial time a solution close to the optimal. This reasoning gave rise to the field of polynomial-time approximation algorithms, which has played (and continues to play) a central role in theoretical computer science (see e.g. the standard textbooks [169, 170]).

The initial (optimistic) hope behind the development of this field was that we may be able to obtain, for many optimization problems, a type of algorithm called a polynomial-time approximation scheme (PTAS), that is, an algorithm which can approach the optimal within arbitrary accuracy $1+\epsilon$ for any $\epsilon > 0$, while still running in polynomial time. Unfortunately, outside of some sporadic exceptions (such as Knapsack-type problems), these hopes were dashed in the 1990s by the development of the celebrated PCP theorem [7, 8]. The PCP theorem (and subsequent work) showed that for most optimization problems on graphs not only is there no hope of obtaining a PTAS (unless P=NP), but often that even coming within a constant factor of the optimal solution is NP-hard (see the survey [167] for more details). Although the development of the PCP theorem and the corresponding hardness of approximation theory is one of the greatest achievements of theoretical computer science in the last few decades, this work unfortunately led to the production of mostly negative results. It is interesting, however, to note that some of the few celebrated cases where a PTAS was indeed shown to be achievable were cases which also took into account the input structure, such as the development of a PTAS for the Euclidean TSP [4, 147].

Looking back at these developments, and even though the study of polynomial-time approximation algorithms remains a vibrant field of research, it is fair to say that by the late 1990s it was more or less clear that most optimization problems on graphs which are NP-hard to solve exactly were also known to be hard to approximate (and in some cases, such as Clique and Chromatic Number were completely inapproximable [73, 108]). Hence, approximation was not and could not be a general solution for dealing with NP-hardness, except perhaps in some special cases or in conjunction with other approaches, and modern research on approximation algorithms has mostly moved away from identifying the rare problems which admit a PTAS. This is of course rather disappointing, as an approximation guarantee weaker than a PTAS is almost always too weak to be of interest in practice. However, the reader should not despair, as later on we will pick up the thread of approximation and revisit the conditions that need to be met to obtain approximation schemes in a context where input structure is also taken into account.

An organized way to deal with structure: Parameterized Complexity. The story we have told so far seems to have reached an impasse: most problems are hard, even to solve approximately, and even though restricting the input structure does lead to some positive results, the search for such structure is a bit ad-hoc and such positive algorithmic results are not too common. Let us therefore recall the idea of parameterized complexity, a different way to do computational complexity that redefined the notion of tractability, going back to the pioneering work of Downey and Fellows and their coauthors in the 1990s [59]. The presentation we will follow is heavily influenced by the eloquently named "distance from triviality" notion as advocated by Niedermeier [151]. The main ingredients are:

1. Take an NP-hard problem, for example a problem defined on graphs.
2. Find some special case of the problem which is known to be polynomial-time solvable (the "trivial" case). For example, the problem may be known to be tractable on forests (a common case), or bipartite graphs, or some other similar class. Or, the problem may be known to be tractable when the objective function has a value in a certain range.
3. Define a notion of distance from the special case, which for any given input instance defines a parameter k. Intuitively, $k = 0$ when the input instance belongs to the trivial case, and k has higher values when the input is "far" from that case.

In this framework, the complexity of a problem is measured as a function of two variables: the input size n and the parameter k which, intuitively, measures how hard an input instance is expected to be (in the sense that the "further" an instance is from the trivial case, the harder we expect it to be to solve). Notice that, since we have not really put any restrictions on what kind of trivial cases we may consider (except of course that such cases must be tractable in the classical sense), nor in the ways we may measure distance, this framework is extremely general. Furthermore, as we promised, this framework is more flexible than the graph classes framework, as inputs which are "close" to some desired class can be handled in a graceful way. The work we present in this document falls exactly in this general area of parameterized complexity: we are concerned with measuring complexity as a function of a structural graph parameter k.

Give us more variables! Before we continue our historical narrative, let us explain why we view the development of parameterized complexity theory as a turning point that forms the foundation of the current work. In order to do this, we present a short *manifesto* making the case that the correct way to develop computational complexity theory is to build it around this framework. This case has of course been made before in parameterized complexity textbooks [49] but the typical argumentation promoting the parameterized approach is the engineer's perspective: we observe that since we are talking about NP-hard problems, very often the best exact algorithm available has complexity 2^n, or perhaps c^n for some constant c. One may therefore argue that the parameterized framework, which allows us to discover algorithms with complexity, say, $2^k n$, is interesting because such algorithms may be of high practical value. In particular, if the parameter is selected judiciously, k will be much smaller than n in many practical instances, so $2^k n$ may be perfectly feasible. Hence, we get an algorithm that works well in real life, even though the problem is in general intractable.

This motivation is fine as far as it goes and is often used to lure people in parameterized complexity, but is not the true reason we adopt this approach in this work. Our true motivation is more philosophical. In a sense, the big question behind computational complexity theory is not simply establishing that some problems are harder than others, but understanding *why* that is the case. The traditional single-variable version of this theory is only able to say that, for example, Dominating Set is a hard problem because as we consider larger graphs, the time needed to solve the problem grows exponentially in the size of the input, hence this problem is harder than, say Minimum Spanning Tree. But the true question is *what aspect* of these larger inputs is it that makes Dominating Set so much harder, *why* does this problem become so much harder so fast? The main advantage of the parameterized approach is that it gives us the tools to pinpoint

specific structural aspects that have a particular impact on the problem at hand. For example, for Dominating Set we know that its complexity increases exponentially on the input graph's *treewidth* with the form $3^t n$, and this is best possible [140]. In a way this gives one possible answer to our question: it tells us that the reason Dominating Set (and many other problems!) gets harder as instances get larger is not the size of the instances per se, but the fact that it becomes harder and harder to use small separators to break them down into individual pieces. This is exemplified by the fact that if we restrict ourselves to nicely separable instances, the complexity of the problem would only be exponential in the degree of "separability" (i.e. the treewidth) and polynomial in n. Thus, if we put front and center the structure of the instance (measured by treewidth), we understand that this structural measure is a primary source of the problem's complexity, more important than the input size. Through this lens we are then not only trying to find potentially useful algorithms, but principally attempting to uncover the deeper connection between structure and complexity.

The motivation above is perhaps a long-winded way of saying something rather simple. In computational complexity we want to measure how hard it is to compute the answer to some problem, and it is natural to measure this as a function of how complicated the input is. It goes without saying that for all problems, as inputs become more complicated, it should become harder to compute a correct output! The question is, how fast does problem complexity grow as a function of *input complexity*? Traditional computational complexity correctly chose to treat this question asymptotically (i.e. as input complexity goes to infinity) but its weakness is that it uses input *size* as a higly imperfect proxy for input *complexity*. This is justified as a first step to exploring the field, but falls far short of allowing us to understand what is really going on, because obviously size and complexity are not the same thing, that is, if an input is large, it is not necessarily hard. The role of multi-variate (parameterized) complexity theory is then to allow us to replace the imperfect proxy of size by fine-tuned parameters that measure the complexity of the input in any way that we care to measure it, and hence more clearly understand the rules that govern computational complexity.

The main demand in our manifesto is therefore the following: when determining the complexity of computing something, we demand to take into account the structure of the input. This will be the guiding principle of this work, and we will explore some ways, some of them well-studied some of them newer, which allow us to measure input structure in ways that go well beyong measuring the number of bits we have been given.

Natural vs Structural Parameterized Complexity. Picking up again on our historical narrative on the development of computational complexity, it is important to note that the manifesto in favor of parameterized complexity given above is solidly grounded in the world of 2021 and based on a lot of hindsight. The foundational work of Downey and Fellows in the 1990s which established parameterized complexity as a viable field was initially primarily concerned with graph problems where k was the value of the objective function (k-Clique, k-Vertex Cover, k-Dominating Set, k-Path, and so on). All this work can be seen through the prism of the "distance from triviality" framework since these problems do become easy when $k = 0$, but this choice of k does not truly measure input structure, or it does so in a trivial way. Nevertheless, this foundational work forms the basis of the whole field, including the case we most care about here where k is a purely structural measure. As a result, the parameterization by the objective value is often nowadays called the *natural* parameter, to distinguish it from the more structural measures we will focus on in this work.

One of the main technical contributions of the work of Downey and Fellows was the development of the tools that allowed (under certain complexity hypotheses) to distinguish problems which are fixed-parameter tractable (FPT), that is, solvable in $f(k)n^{O(1)}$ time, for problems which are fixed-parameter intractable, even if they are solvable in $n^{g(k)}$ time[2]. This was based on an analogue of NP-completeness theory that defined a hierarchy of classes, the W-hierarchy, and showed that, for

[2]Problems which are NP-complete already for constant values of k, called paraNP-hard in this framework, could already be distinguished from FPT problems using the standard tools of NP-completeness.

example, while k-Vertex Cover is solvable in $2^k n^{O(1)}$, k-Clique and k-Dominating Set are complete for two higher classes (W[1] and W[2] respectively), indicating that these problems are unlikely to be fixed-parameter tractable for the natural parameter.

Even though much of the early work on parameterized complexity focused on the natural parameter, a structural measure that we have already mentioned also played a key role from the early days. We are talking of course about treewidth, a measure that quantifies how close a graph is to being a tree (and since trees are the prototypical graphs with small separators, low treewidth is connected with graph separability as hinted above). Since most problems become polynomial-time solvable on trees and forests, it is quite intuitive that a large class of problems is fixed-parameter tractable when parameterized by treewidth, as exemplified by the celebrated Courcelle's theorem [46]. Treewidth, thanks to its elegant combinatorial structure and great algorithmic properties has played a key role in the development of parameterized complexity theory, including as a tool in the study of naturally parameterized problems, and is one of the key notions that will interest us in this work. Indeed, one of the main themes that we will tackle is the comparison between treewidth and other related ways of measuring the complexity of a graph, and in particular the algorithmic trade-offs involved between generality and tractability.

Treewidth and Structural Parameters. Summarizing the role that treewidth has played in the development of parameterized complexity would be a difficult task that goes beyond the scope of this document, as treewidth-based arguments are ubiquitous in parameterized complexity (to name two examples, one of the oldest win/win FPT algorithms for k-Path used treewidth [26], while bidimensionality theory could not exist without treewidth [84]). However, since our goal here is to give a biased narrative of the events of the last few decades, let us concentrate on a specific period in time that saw the development of some ideas that prove crucial to this work.

In the late 2000s parameterized complexity was already on the way to becoming a well-established field and the general view of the community was that treewidth was well-understood and "easy". We use here the term easy in the computational complexity sense: the consensus was that, barring some obnoxious exotic problems which were already NP-hard on forests, pretty much any reasonable problem was fixed-parameter tractable parameterized by treewidth. This was of course supported by the aforementioned meta-theorem of Courcelle, which covered the very wide class of MSO-expressible problems, but also other results that established tractability for problems not expressible in this framework. From our point of view in 2021, this state of affairs showed that the integration of treewidth into the parameterized complexity framework was only half-done: treewidth was widely used as a tool to obtain FPT algorithms, but it did not yet play any role in the corresponding parameterized intractability theory.

This changed with the discovery by Fellows et al. [75] of the first natural graph problem which is solvable in $n^{O(t)}$ time on graphs of treewidth t, but is *not* FPT (in fact is W[1]-hard) parameterized by t, namely List Coloring. This discovery was of great programmatic importance, because along with other results that appeared around the same time [3] it showed convincingly that parameterized intractability theory had something interesting to say about the interplay between structure and complexity. This line of thought was later crystalized in the work of Fomin et al. [86] who showed that several classical problems which are FPT parameterized by treewidth become W-hard when parameterized by the more general notion of clique-width.

The works mentioned in the previous paragraph have a huge influence in the line of research we present here because one of the key challenges of parameterized complexity theory is the choice of the parameter k. We therefore need the tools that will allow us to understand the algorithmic trade-offs involved with each definition of distance from triviality. Transposing these results to the point of view we have presented, let us compare what we know about Dominating Set (which we recall is solvable in time $3^t n$ and this is optimal) and List Coloring. In our philosophical framework, these results tell us that treewidth is a key factor in the complexity of Dominating

[3] Dom et al. and Szeider [58, 164] gave similar results for other problems on treewidth, while [132] was the first work to do so for directed treewidth.

Set: the complexity of the problem increases smoothly[4] with t, so we can interpret this as saying that the complexity increases "because of" the increase in treewidth. In contrast, List Coloring is already a very hard problem for, say $t = 15$. We can therefore conclude that it's not treewidth that governs the complexity of List Coloring (otherwise the problem would be easy when the treewidth has moderate values) but something else. Hence, on a philosophical level, the parameterized complexity framework gives us the tools to identify the structural measures that "count" for each particular problem.

Going beyond our philosophical quests, selecting a parameter is crucial because an ideal parameter would combine generality (many instances have small k) and algorithmic tractability, two properties which are clearly in conflict with each other. It is therefore important to understand how much we sacrifice in one respect when we select a parameter that gives us more in the other. The work of Fomin et al. [86] was the first to explicitly bring this question to the table by examining clique-width, a notion that covers "more" graphs than treewidth, and explicitly asking what is the algorithmic cost of this generality, that is, which are the problems whose tractability is lost if we pick k to be the clique-width, rather than the treewidth of the input graph. Starting from this example, "price of generality" research has become a topic of interest in parameterized complexity, and nowadays many examples are known differentiating, for example, treewidth from tree-depth, or tree-depth from vertex cover, and so on, giving us a clearer sense of the algorithmic properties of standard structural parameters.

Our historical narrative has thus brought us to the first part of this work, which is devoted exactly to the study of which structural graph parameters give fixed-parameter tractability for various graph problems. The goal of this line of research is, as explained above, to attempt to understand which structural notions are "key" for each problem; and to attempt to understand how different notions of graph complexity behave algorithmically with respect to different problems, and notably when two notions have a clear hierachical relationship, how much the added generality of the more relaxed notions costs in computational complexity. In Chapter 3 we mention several works whose focus is exactly this type of study, gravitating around treewidth and several related measures which have attracted interest in the last decade. In particular, we will focus on treewidth-like measures, such as pathwidth, tree-depth, feedback vertex set, and vertex cover, as well as measures that cover dense graphs, such as clique-width, neighborhood diversity, and modular-width. The problems we study are (variations of) standard graph problems, such as Coloring, Dominating Set and others, as well as problems where graph structure is used secondarily, such as SAT. All the structural parameters studied are summarized in Figure 2.1 given in Chapter 2, where we also give precise definitions, while we defer a more detailed survey of known results to Chapter 3.

Fine-Grained Complexity and its applications. Let us now retrace a few steps in our narrative and go back to the beginning of the 2000s to revisit another line of research that will prove crucial to our objectives. It is around that time that Impagliazzo, Paturi, and Zane [110, 111] formulated the Exponential Time Hypothesis (ETH) and the Strong Exponential Time Hypothesis (SETH). Stated informally, the ETH is the hypothesis that 3-SAT cannot be solved in sub-exponential time, that is, in $2^{o(n)}$ time for n variables, while the SETH states that SAT cannot be solved in $(2-\epsilon)^n$ time. In other words, the ETH and SETH state in a stronger way a principle that stands behing the P$\neq$NP conjecture, namely that brute-force search is inevitable for NP-hard problems. Of course, these conjectures still remain open, with the ETH considered today mostly reasonable, while the SETH is seen as a bit more tenuous; nevertheless both are consistent with current algorithmic knowledge.

On the face of it, the formulation of these two conjectures doesn't seem to have much to do with our topic of structural parameterized complexity – after all, the ETH and the SETH are conjectures about the classical complexity of SAT. In an exciting twist though, the last two

[4]We use the term smoothly a bit liberally here, of course. The complexity increases exponentially in t, but of course the running time must be exponential in something, as the problem is NP-hard, so this is the smoothest behavior we can expect.

decades have shown that these two conjectures can be used to build a fine-grained understanding of parameterized complexity theory, without the need for additional complexity classes such as those given in the W-hierarchy. What we mean by fine-grained is that the goal of this theory is not the classification of problems into rough complexity classes, but the determination, in a quantitative way, of lower bounds on the running time needed to solve a problem which match as closely as possible with the known upper bounds. These research programs are often referred as the FPT-optimality and XP-optimality program, where the goal is to determine, for a problem that admits an $f(k)n^{O(1)}$ time algorithm or an $n^{g(k)}$ time algorithm, the best possible functions f, g that make this possible.

One of the points of departure of this line of research was the work of Chen et al. [41] who proved that, assuming the ETH, k-Clique cannot be solved in time $f(k)n^{o(k)}$ for any function f, that is, the correct exponent of n is linear in k. This result is of major importance because k-Clique is the prototypical W[1]-hard problem, hence reinterpreting existing FPT reductions from k-Clique under this light allowed to obtain similar (often tight) results for other parameterized intractable problems. Even though many of the results this line of work on XP-optimality has given were of a similar flavor (proving that a problem solvable in n^k cannot be solved in $n^{o(k)}$ under the ETH), more and more sophisticated reductions gradually allowed to obtain more and more sophisticated bounds, including for structural parameters. We cite as a characteristic example the recent work of Fomin et al. [87] which showed that Coloring cannot be solved in $n^{2^{o(\mathrm{cw})}}$, where cw is the input graph's clique-width (this is known to be tight by an algorithm of [125]). Notice that all these works only allow to determine the function $g(k)$ asymptotically, since they are based on the ETH which only determines the exponent of the complexity of 3-SAT asymptotically. A similar research program based on the SETH could also be built, notably Patrascu and Williams [156] showed that k-Dominating Set cannot be solved in $n^{k-\epsilon}$ under the SETH for any $\epsilon > 0$, but this approach is so far less developed.

Motivated by the success of the XP-optimality program, much effort has also been devoted in the last 15 years into a corresponding FPT optimality program which attempts to determine the best "parameter dependence", that is, the best function $f(k)$ in a running time of the form $f(k)n^{O(1)}$. The initial wave of results in this area simply confirmed that standard algorithms were in the right ballpark, for example that k-Vertex Cover cannot be solved in $2^{o(k)}n^{O(1)}$ time. This gradually led to more interesting results, proving optimality even for problems with exotic-seeming running times. A famous example is k-Edge Clique Cover, whose double-exponential dependence on k was shown optimal by Cygan et al. [52] (see [1, 126] for other double-exponential examples from the same period). Less dramatically, Lokshtanov et al. [141] set out to investigate problems whose complexity has the form $k^{O(k)}$, under the ETH. For more such results we refer the reader to [139].

Interestingly, the FPT-optimality program developed an important second branch relying on the SETH. This was jump-started by the work of Lokshtanov et al. [140] who showed that the known algorithms for several important problems parameterized by treewidth were optimal, including the constant in the base of the exponent. Notably, this included the result we have already evoked stating that Dominating Set cannot be solved in $(3-\epsilon)^t n^{O(1)}$, which was shown under the SETH. Following this ground-breaking work, much effort was devoted to discovering the "correct" constants for various other problems and parameters.

Looking back at all this work we notice that fine-grained parameterized complexity theory, based on the (S)ETH, has morphed into one of the main directions of the field going forward. Naturally, we will then devote a significant fraction of this document (all of Chapter 4) to results of this type, especially because this approach can naturally be seen as a refined version of the effort we were already making to understand the impact of structure on complexity. Indeed, using the tools given to us by the (S)ETH, we are now able to not only classify the relationship between graph and problem complexity into rough classes (FPT vs W-hard), but we can actually quantify their interdependence. Arguably, this is the research direction that corresponds more closely to our original philosophical goal of "measuring the impact" of input structure on computational complexity. Thanks to the (S)ETH we will be able to confidently predict that the complexity

of a problem is, say 4^k (and not 3.9^k or 4.1^k), or $2^{O(k^2)}$ (and not 2^k or 2^{k^3}). Crucially, this line of work further enriches our study of different parameters, as the price of generality can now be measured not only in problems "lost" (that is, transitioning from FPT to W-hard for a more general parameter), but in the precise amount of increase in complexity, even for problems that remain in the same rough complexity class. We will present several such results in Chapter 4, where we will notably show that Dominating Set and 3-Coloring, in the transition from treewidth to clique-width go from a complexity of 3^t (for both problems) to 4^{cw} for one and 6^{cw} for the other. Observe that the use of the SETH is crucial for making such fine distinctions, as both of these problems are FPT for both parameters.

The Future: the best of all worlds? So far, we seem to have narrated a mostly coherent self-contained story: computational intractability was observed as a major phenomenon in the early 1970s; input structure was initially exploited as a possible solution by considering classes of inputs; this later led to the development of structural parameterized complexity theory, which is a more flexible and general version of the same approach; and more recently, fine-grained complexity tools have allowed this theory to evolve to the point that we can precisely measure the impact of structure on computational complexity for many of the problems we most care about. Our work could just continue along this path, but no self-respecting narrative should be allowed to contain loose ends, and our biased telling of the story of computational complexity has left such a major loose end in the development of approximation algorithms. Let us then try to pick up this loose end and put everything together to arrive at a happy ending.

As a reminder, our narration on the field of approximation algorithms stopped around the late 1990s, when it became clear that most of the (graph) problems we care about in this work are NP-hard to approximate. Despite this disappointing development, this does not necessarily mean that approximation was a bad idea; rather the lesson here is that, just like Edmonds' initial definition, the concept of a PTAS as the definition of approximate tractability was simply too ambitious. Modern research in polynomial-time approximation algorithms has adjusted to this reality by lowering its expectations (a constant-factor approximation is now considered a positive result in this domain). However, another way forward is to keep the AS part of the definition of PTAS (that is, continue demanding $1 + \epsilon$ accuracy for arbitrary ϵ), while replacing the PT part with the ideas that have worked so nicely in the parameterized complexity world as alternative notions of tractability.

More concretely, the third main direction presented in this work is the study of the following type of question: given a structurally parameterized problem that is known to be fixed-parameter intractable, is it possible to obtain a $(1 + \epsilon)$-approximate FPT algorithm. In other words, we will be interested in FPT (rather than polynomial-time) approximation schemes. In a sense, this approach combines all the ideas we have touched upon so far in our overview and hence combines all the tools we have at our disposal for dealing with NP-hardness. The study of FPT approximation algorithms is a relatively new field (see the excellent recent survey of Feldmann et al. [74]), especially when it comes to structurally (rather than naturally) parameterized problems. On the positive side, this means that things are wide open, and the prospect of obtaining positive results is much less far-fetched than in the context of polynomial-time algorithms. Of course, with time a hardness theory will have to gradually be developed, but at the moment we will mostly present some positive results in this vein in Chapter 5. Notably, we will present a few applications of a generic method introduced in [129] which often leads to an FPT approximation scheme when a hard problem is parameterized by treewidth and some natural conditions are satisfied.

Organization of the rest of this document: Let us make a short recap to help the reader navigate through the rest of this document. At this point we have briefly explained why we are interested in structural parameters for graph problems; that we are going to use the tools of parameterized complexity in their traditional and fine-grained forms to study various problems under this lens; and that in the end we want to add the notion of approximation into the mix. We continue in Chapter 2 with a few definitions of standard notions. This is mostly done for

completeness and a reader familiar with the area may feel free to skip much of this chapter. However, it may still be useful to take a look at Figure 2.1, which summarizes the structural parameters we will study, as well as the corresponding definition. These definitions will be used throughout the document. Then, in Chapter 3 we present results that follow the classical pattern of parameterized complexity, that is, distinguishing fixed-parameter tractable from intractable problems, albeit always using a structural graph measure as a parameter. In other words, this is the chapter where we focus on "price of generality"-type questions regarding these structural parameters. In Chapter 4 we then move on to present more modern "fine-grained" results in this area which, assuming stronger complexity hypotheses, attempt to exactly pinpoint the complexity of various problems. In Chapter 5 we discuss how the ideas of the previous two chapters can be combined with the notion of approximation, in order to produce fixed-parameter approximation algorithms for intractable optimization problems. Finally, in Chapter 6 we give some conclusions and directions to further work.

Chapter 2

Definitions and Preliminaries

2.1 General Background and Computational Complexity

We will for the most part use standard graph-theoretic notation, see for example [57]. An undirected graph $G = (V, E)$ is a pair formed by a set of vertices V and a set of edges E, such that each edge connects a pair of distinct vertices. Unless otherwise stated, we do not allow parallel edges or self-loops. We use $N(v)$, for $v \in V$ to denote the set of neighbors of a vertex v, that is, $N(v) = \{u \mid uv \in E\}$. We use $d(v)$ to denote the degree of a vertex, that is, $d(v) = |N(v)|$, and Δ to denote the maximum degree of a graph. For a set $S \subseteq V$, the induced subgraph $G[S]$ is the subgraph of G that contains all vertices of S and all edges with both endpoints in S. We will sometimes write $G - S$ or $G - v$ to denote the graph $G[V \setminus S]$ or the graph $G[V \setminus \{v\}]$ respectively.

We assume the reader is familiar with standard concepts in computational complexity theory, such as the classes P, NP, PSPACE, and so on (see [5]). In general we will not insist on the details of a model of computation as these are not relevant unless otherwise stated. We will therefore describe algorithms in an informal way, without using pseudocode. We also assume that the reader is familiar with the concept of (many-one) polynomial-time reductions, as used to establish the NP-completeness of standard problems [100].

Let us recall some basic definitions from parameterized complexity, though we refer the reader to [49] for a complete treatment. The input to a parameterized problem is a pair (χ, k), where χ is a binary string of length n encoding the instance and k is a non-negative integer called the parameter. A problem belongs in the class FPT (fixed-parameter tractable) if it admits an algorithm running in time $f(k)n^c$, for some computable function f and some constant c that does not depend on the input. A problem belongs in the class XP if it can be solved in time $n^{g(k)}$ for some computable function g.

An FPT reduction from a problem A to a problem B is an algorithm which, given an instance (χ, k) of problem A, produces an equivalent instance (χ', k') of problem B, runs in time $f(k)|\chi|^{O(1)}$, and has $k' = g(k)$, for some functions f, g. Recall that FPT reductions preserve fixed-parameter tractability, that is, if A can be reduced to B and B belongs to the class FPT, then A also belongs to the class FPT.

Using this concept of reductions one may define the main intractability classes of this theory. W[1] can be defined as the class of problems which are FPT-reducible to k-Clique, the problem where we are given a graph G and an integer k and are asked whether G contains a clique of size k. W[2] is the class of problems FPT-reducible to k-Dominating Set, where this time we are asked if G has a dominating set of size k. It is a standard fact of parameterized complexity theory that FPT $\subseteq$ W[1] $\subseteq$ W[2], and these inclusions are widely believed to be strict. Hence, one way to show that a problem is unlikely to be FPT is to FPT-reduce k-Clique to that problem.

Let us also recall two popular complexity hypotheses, due to [110, 111], that we will often use in this work.

Definition 1 *The Exponential Time Hypothesis (ETH) states the following: there exists a constant $c > 1$ such that 3-SAT on instances with n variables and m clauses cannot be solved in time c^{n+m}.*

Definition 2 *The Strong Exponential Time Hypothesis (SETH) states the following: for every $\epsilon > 0$ there exists a constant q (depending only on ϵ) such that q-SAT on instances with n variables and m clauses cannot be solved in time $(2-\epsilon)^n(n+m)^{O(1)}$.*

We will often use some slightly weaker, but easier to state, versions of these two hypotheses. Namely, we will sometimes view the ETH as the hypothesis that 3-SAT cannot be solved in time $2^{o(n+m)}$ and the SETH as the hypothesis that SAT (with unbounded clause size) cannot be solved in time $(2-\epsilon)^n$ for any $\epsilon > 0$. The definitions we gave above imply these weaker versions, hence any time we establish a result starting from these weaker versions, the result will also follow if we assume the "official" versions above. A key fact that we will use repeatedly is that, assuming the ETH, k-Clique does not admit any algorithm running in $n^{o(k)}$ time. Using standard reductions the same follows for k-Dominating Set [41, 49].

2.2 Structural Parameters

One of the main themes of this work is that we will be interested in measuring the structural complexity of a graph and then attempting to use this information to determine the computational complexity of solving a problem on that graph. There are, however, many ways to quantify how complicated a graph is, some more sophisticated than others, and one of our main points of interest will be in evaluating and comparing different ways of measuring graph complexity.

We therefore invite the reader to take a look at Figure 2.1, which summarizes the relations between the structural graph parameters that we will be most interested in in this work. Let us first give formal definitions for all these parameters.

2.2.1 Definitions of our Parameters

Treewidth (tw). Informally, treewidth measures the tree-likeness of a graph. In order to define this measure we need the notion of a tree decomposition. A tree decomposition of a graph $G = (V, E)$ is a tree T such that each node $x \in T$ is associated with a subset $B_x \subseteq V$ and we have the following: (i) for every edge $uv \in E$ there exists $x \in T$ with $\{u, v\} \subseteq B_x$ (ii) for every $x, y \in T$ and for every $z \in T$ that is found in the (unique) path from x to y in T, we have $B_x \cap B_y \subseteq B_z$. The width of a decomposition T is defined as $\max_{x \in T} |B_x| - 1$ and the treewidth of a graph G is simply the minimum width of any tree decomposition of G.

Tree decomposition are a very well-known and widely used tool in parameterized complexity theory (we refer the reader to [49]). For the purposes of designing algorithms it is often convenient to concentrate on tree decompositions with a special form, called *nice* tree decompositions. In this case, T is a rooted binary tree whose nodes have one of four possible types: (i) Leaf nodes x, where $B_x = \emptyset$ (ii) Introduce nodes x, where x has a unique child y and $B_x = B_y \cup \{v\}$ for some $v \notin B_y$ (iii) Forget nodes x, where x has a unique child y and $B_y = B_x \cup \{v\}$ for some $v \notin B_x$ (iv) Join nodes x, where x has two children y, z and $B_x = B_y = B_z$. It is known that given any tree decomposition one can in linear time construct a nice tree decomposition of the same width.

Pathwidth (pw). This is the same definition as for treewidth, except that we now force the tree decomposition T to be a path.

Tree-depth (td). Tree-depth ([149]) can be inductively defined as follows: a singleton has tree-depth 1; the tree-depth of a disconnected graph is the maximum of the tree-depth of its components; a connected graph G has tree-depth at most k if there exists $v \in V(G)$ such that $G - v$ has tree-depth at most $k - 1$. Equivalently, G has tree-depth at most k if there exists

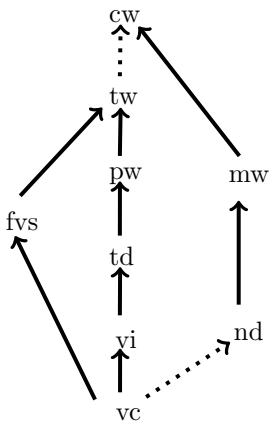


Figure 2.1: The structural parameters we focus on in this work.

a rooted tree of height at most k such that if we connect every node of the tree with all of its ancestors we obtain a super-graph of G.

Vertex Integrity (vi). A graph G has vertex integrity at most k if there exists $S \subseteq V(G)$ such that every component of $G - S$ has size at most $k - |S|$ [101].

Vertex Cover (vc). A graph G has vertex cover at most k if there exists a set $S \subseteq V(G)$, with $|S| \leq k$, such that all edges have at least one endpoint in S.

Feedback Vertex Set (fvs). A graph G has feedback vertex set at most k if there exists a set $S \subseteq V(G)$, with $|S| \leq k$, such that all cycles in G contain at least one vertex in S.

Clique-width (cw). A labeled graph G has clique-width at most w if it can be constructed using w labels and the following four basic operations: Introduce(i), for $i \in \{1, \ldots, w\}$, which constructs a single-vertex graph whose vertex has label i; Union(G_1, G_2), which constructs the disjoint union of two labeled graphs of clique-width w; Rename(i, j) which changes the label of all vertices labeled i to j; and Join(i, j) which adds all possible edges between vertices labeled i and vertices labeled j. Computing a graph's clique-width, that is, the minimum number of labels needed to construct the graph with the above operations, is NP-hard [77], and the best currently known approximation is exponential in clique-width [154]. We view a clique-width expression as a rooted binary tree, where the sub-tree rooted in each internal node represents the corresponding sub-graph of G.

Modular-width (mw). Given a graph G, a module is a set $S \subseteq V$ with the following property: every vertex $x \in V \setminus S$ is either connected to all vertices of S or to none. A graph G has modular width at most k if $|V(G)| \leq k$ or if it is possible to partition $V(G)$ into at most k sets $V_1, \ldots, V_k$, such that each V_i is a module and $G[V_i]$ has modular width at most k [93].

Neighborhood diversity (nd). Two vertices u, v of a graph G are called twins if $\{u, v\}$ is a module. The neighborhood diversity of G is the smallest k such that $V(G)$ can be partitioned into at most k sets $V_1, \ldots, V_k$, such that for each V_i, all vertices of V_i are pairwise twins [126].

2.2.2 Relations between Parameters.

Having given the formal definitions of our structural parameters, let us try to briefly explain the map given in Figure 2.1. The arrows in the figure are supposed to be read as indicating

generalization. More precisely, whenever we have a solid arrow from parameter A to parameter B it means that the class of graphs where A is bounded by a value k is contained in the class of graphs where B is bounded by $O(k)$. In this sense, B is "more general", because for (almost) the same value of the parameter it covers more graphs. Stretching this logic a bit, dashed arrows indicate the same relation with the caveat that the parameter value may increase exponentially, that is, B is bounded by $2^{O(k)}$. When we have no arrow between two parameters this indicates that they are incomparable (i.e. there exist graphs where one is constant and the other unbounded).

Before we review the known relations depicted in Figure 2.1, let us point out why this map is algorithmically interesting. All the arrows in the figure indicate a relation where we know that if one parameter is bounded by k, the other is bounded by some function $f(k)$. This indicates that, when we have an arrow from parameter A to parameter B, the identity function is a valid FPT-reduction for an problem parameterized by A to the same problem parameterized by B. As a result, if a problem is W[1]-hard parameterized by A, it is also W[1]-hard parameterized by B. Informally, negative results propagate upwards. Symmetrically, algorithmic results propagate downwards. For example, the relations between parameters tell us that if a problem is FPT by treewidth, it is automatically FPT by pathwidth, and also by tree-depth and vertex cover. Conversely, if a problem is W[1]-hard by neighborhood diversity, it must also be W[1]-hard by modular width and clique-width. In this light, the "price of generality" questions that we will be concerned with in the next chapter can be seen as attempting to trace, for each problem, the frontier where the problem transitions from being FPT to W[1]-hard.

Let us therefore review the relations between all these parameters:

1. The fact that $\mathrm{cw}(G) \leq 4 \cdot 2^{\mathrm{tw}(G)-1} + 1$ was shown in [45].

2. The relation $\mathrm{tw}(G) \leq \mathrm{pw}(G)$ is trivial, as we consider a special class of tree decompositions (paths).

3. The relation $\mathrm{tw}(G) \leq \mathrm{fvs}(G)+1$ can be seen by constructing a tree decomposition of a given graph G and its feedback vertex set S as follows: we place all vertices of S in every bag of a tree decomposition of $G - S$.

4. The relation $\mathrm{pw}(G) \leq \mathrm{td}(G)$ can be seen as follows (by induction): if G is disconnected, we simply glue together the path decompositions of each of its components; if G is connected, there exists $v \in V(G)$ such that $\mathrm{td}(G) = 1 + \mathrm{td}(G - v)$, so we add v to all bags of a path decomposition of $G - v$.

5. The relation $\mathrm{td}(G) \leq \mathrm{vi}(G)$ can be seen by induction as follows: if all components of G have size at most $\mathrm{vi}(G)$ the relation is trivial; if not, G has a separator S such that all components of $G-S$ have size at most $\mathrm{vi}(G)-|S|$. It now suffices to pick an arbitrary $v \in S$ and observe that by induction $\mathrm{td}(G-v) \leq \mathrm{vi}(G-v)$. But $\mathrm{td}(G) \leq \mathrm{td}(G-v)+1 \leq \mathrm{vi}(G-v)+1 = \mathrm{vi}(G)$.

6. The relation $\mathrm{vi}(G) \leq \mathrm{vc}(G) + 1$ follows from the fact that the vertex cover of a graph is a separator whose removal leaves a graph where all components are singletons.

7. The relation $\mathrm{fvs}(G) \leq \mathrm{vc}(G)$ is trivial.

8. The relation $\mathrm{cw}(G) \leq \mathrm{mw}(G)$ can be seen by induction as follows: suppose we have a graph G where $V(G)$ can be partitioned into k modules $V_1, \ldots, V_k$, and we have a k-clique-width expression for each $G[V_i]$. For each i, add to the root of the clique-width expression of $G[V_i]$ appropriate Rename operations so that all vertices obtain label i. We then take the Union of the k expressions and add appropriate Join operations to obtain G (observe that between two modules V_i, V_j we have either all possible edges or none).

9. The relation $\mathrm{mw}(G) \leq \mathrm{nd}(G) + 1$ is given in [93] and follows directly from the fact that a set of twins is a module that forms a clique or an independent set (and such graphs have modular width 2).

10. The relation $\mathrm{nd}(G) \leq 2^{\mathrm{vc}(G)} + \mathrm{vc}(G)$ is given in [126] and follows from the fact that, given a vertex cover S, we can partition $V \setminus S$ into at most $2^{|S|}$ sets of vertices with identical neighborhoods.

Non-equivalences Many of the measures we have presented have different variations which are known to be more-or-less equivalent. For example, branchwidth is a notion that is known to be a constant factor apart from treewidth [88] and hence every problem that is FPT for one is FPT for the other. Similarly, rank-width [153] and boolean-width [39] are measures which are FPT-equivalent to clique-width, though in this case the relationship is not as tight (we only know that one parameter is bounded by an exponential function of the other). The measures we have selected to present here, however, are picked exactly with the intent to highlight truly distinct structural aspects. Hence, it is important to argue that the arrows of Figure 2.1 are strict, and parameters that appear more general in the figure are not in fact equivalent to parameters lower in the hierarchy. Let us give specific graph constructions that prove this:

1. Treewidth cannot be bounded by a function of clique-width as K_n has constant clique-width but treewidth $n-1$.

2. Pathwidth cannot be bounded by a function of treewidth as a complete binary tree on n leaves has pathwidth $\Omega(\log n)$ [162].

3. Tree-depth cannot be bounded by a function of pathwidth as a path on n vertices P_n has pathwidth 1 but tree-depth $\Omega(\log n)$ (this can be easily shown by induction as removing any vertex of P_n will leave a component that is a path on at least $(n-1)/2$ vertices).

4. Vertex integrity cannot be bounded by a function of tree-depth. For example, if we take n copies of $K_{1,n}$ and add a universal vertex, the tree-depth of the resulting graph is 3, but the vertex integrity is $\Omega(n)$.

5. Vertex Cover cannot be bounded by a function of vertex integrity, as can be seen by subdividing every edge of a $K_{1,n}$.

6. Feedback vertex set cannot be bounded by a function of treewidth, as can be seen by a disjoint union of n copies of K_3.

7. Vertex cover cannot be bounded by a function of feedback vertex set, as can be seen by a path P_n.

8. Modular width cannot be bounded by a function of clique-width, as P_n has clique-width at most 3 but no non-trivial modules for $n \geq 4$.

9. Neighborhood diversity cannot be bounded by a function of modular width, as a disjoint union of n copies of P_4 has neighborhood diversity $\Omega(n)$ (as there are no twins in the graph) but constant modular width (we can set V_1 to be one copy of P_4 and V_2 the rest of the graph and proceed inductively).

10. Vertex cover cannot be bounded by a function of neighborhood diversity, as K_n has neighborhood diversity 1.

Finally, let us justify the missing arrows in our figure, that is, let us show that the parameters not connected by an arrow are incomparable.

1. A K_n has constant neighborhood diversity (hence modular width) but unbounded feedback vertex set. Conversely, a subdivided $K_{1,n}$ has feedback vertex set 0 but unbounded modular width (and hence neighborhood diversity). This shows that $\mathrm{fvs}(G)$ is incomparable to $\mathrm{nd}(G), \mathrm{mw}(G)$.

2. A union of n K_3s has vertex integrity (hence tree-depth and pathwidth) at most 3, but feedback vertex set n. On the other hand, a complete binary tree of height n has pathwidth (and hence tree-depth and vertex integrity) at least $n/2$ ([162]), but feedback vertex set 0. This shows that $\mathrm{fvs}(G)$ is incomparable to $\mathrm{pw}(G), \mathrm{td}(G), \mathrm{vi}(G)$.

3. A K_n has constant neighborhood diversity and modular width but treewidth (and hence pathwidth, tree-depth, and vertex integrity) at least $n-1$. On the other hand, a subdivided $K_{1,n}$ has vertex integrity at most 3 but unbounded modular width. Hence, the parameters $\mathrm{tw}(G), \mathrm{pw}(G), \mathrm{td}(G), \mathrm{vi}(G)$ are incomparable to $\mathrm{mw}(G), \mathrm{nd}(G)$.

2.2.3 Computing Structural Parameters

Let us also briefly discuss a topic that is sometimes swept under the rug in structural parameterized complexity, namely, how one is supposed to calculate the values of the structural parameters of a given graph and a fortiori the related decompositions which are necessary to obtain algorithmic results. We state for the record that, in the remainder of this work, we will in general *not* worry about this question. In particular, when we formulate an algorithm parameterized by, say, treewidth, we will always silently assume that a tree decomposition of optimal treewidth has somehow been made available. The reason we choose this approach is that what mainly interests us is to measure how useful treewidth (and pathwidth, clique-width, tree-depth, etc.) is in helping us solve other problems, that is, what is the impact of the structure measured by this parameter on a problem's complexity. Hence, we factor out the common problem of computing the value of the parameter and concentrate on using the parameter. It is important to note that, when it comes to lower bound results, this assumption actually makes all the results we present *stronger*, because it proves that the hardness comes not from the hardness of computing the parameter, but from the hardness of the problem at hand.

Nevertheless, it is of course clear that when formulating algorithmic results, the assumption that a decomposition of some form is supplied with the graph is not necessarily realistic. It therefore does become a legitimate question how one can calculate the values of all the structural parameters of Figure 2.1, at least approximately or in FPT time. We therefore summarize below some relevant facts.

- The two parameters that avoid treewidth, namely modular width and neighborhood diversity are the most well-behaved in our collection, as they can be computed in polynomial (even linear) time [126, 165].
- Treewidth and pathwidth are known to be NP-hard to compute exactly, but on the positive FPT exact algorithms are known for both parameters [27, 31]. Because the performance of these algorithms is not ideal (Bodlaender's algorithm for treewidth has parameter dependence of the form $2^{O(k^3)}$), approximation algorithms for these problems are also often considered. In polynomial time the best known achievable ratio is $O(\sqrt{\log \mathrm{tw}})$ [72], while a small constant factor approximation is achievable in time $2^{O(k)}n^{O(1)}$ [2, 28] for treewidth.
- Tree-depth is also NP-hard, but known to be FPT. The best currently known complexity for computing it is $2^{O(\mathrm{td}^2)}$, due to an algorithm by Reidl et al. [158] which relies indirectly on treewidth.
- Vertex Integrity is NP-hard but known to be solvable exactly in $\mathrm{vi}^{O(\mathrm{vi})}n^{O(1)}$ time [60].
- Vertex cover and feedback vertex set are of course standard problems in parameterized complexity for which FPT algorithms are well-known [49].
- Finally, clique-width is the toughest customer in our collection. Determining if a graph has clique-width k is not even known to be in XP. The best we can currently do is approximate clique-width in FPT time via the related measure of rank-width [154], but the approximation ratio is exponential in k.

Chapter 3

Structural Graph Parameters

In this chapter we present some results in the "price of generality" line of research, using the traditional tools of parameterized complexity. We will concentrate on the structural graph measures of Figure 2.1 and consider various interesting NP-hard problems. Our goal will be, for each such problem, to trace its border of fixed-parameter tractability, that is, to understand for which of these parameters the problem is FPT and for which it is W[1]-hard.

An astute reader will immediately notice the similarity of this line of research with the approach advocated by Garey&Johnson [100], the difference being that instead of trying to find the largest class of graphs in which an NP-hard problem becomes polynomial-time solvable, we are trying to find the least restrictive notion of structure which makes a problem fixed-parameter tractable. Indeed, in a way this line of research can be seen as the modern analogue of the graph classes approach. However, our point of view here is more parameter-centric than problem-centric. Our eventual aim is to proceed from individual data points towards an understanding of the general patterns that determine why some problem is FPT for one parameter but not for another. In other words, we want to understand what are the strengths and limits of, say, treewidth vis-à-vis clique-width by locating as many characteristic examples as possible of problems whose complexity changes from one parameter to the other.

We begin this chapter with a historical overview of this line of work (Section 3.1). As part of this overview we give a semi-exhaustive list of problems appearing in the literature which are known to be W[1]-hard by treewidth or one of the other parameters we consider. This list may be of general interest as a compendium, as such results are generally rather scattered since reductions have a tendency to start from standard problems (such as k-Clique) rather than build on each other. In the same section we also summarize some general patterns that seem to arise from the literature, that is, we attempt to derive a few intuitive reasons that explain the algorithmic differences between various parameters as indicated by the properties of specific problems.

After thus summarizing what is known, we then go on to present a series of works by the author in this area. Our presentation is guided by structure: in Section 3.2 we give results relevant to what we would call the main hierarchy of our parameters, namely vc $\to$ vi $\to$ td $\to$ pw $\to$ tw $\to$ cw. In particular, our goal here is to give some characteristic problems that trace the frontiers between these parameters. Some of the highlights of this section are the discovery of the first (natural) problem separating treewidth from pathwidth (Grundy Coloring) in Section 3.2.1 and of two natural weighted problems which are already W[1]-hard parameterized by vertex cover in Section 3.2.2. More broadly, in Section 3.2.3 we expose several results that explore the observation that treewidth becomes less efficient when the natural dynamic program is forced to store natural numbers and check under what conditions this can be worked around by using tree-depth. We also present a problem which, despite having some characteristics that would lead us to believe it would distinguish treewidth from clique-width, actually turns out to be FPT for both parameters (section 3.2.4).

We then move on to the second main hierarchy of Figure 2.1 and consider the parameters vc $\to$ nd $\to$ mw $\to$ cw. One of the main results we present here is the (surprising) W[1]-hardness

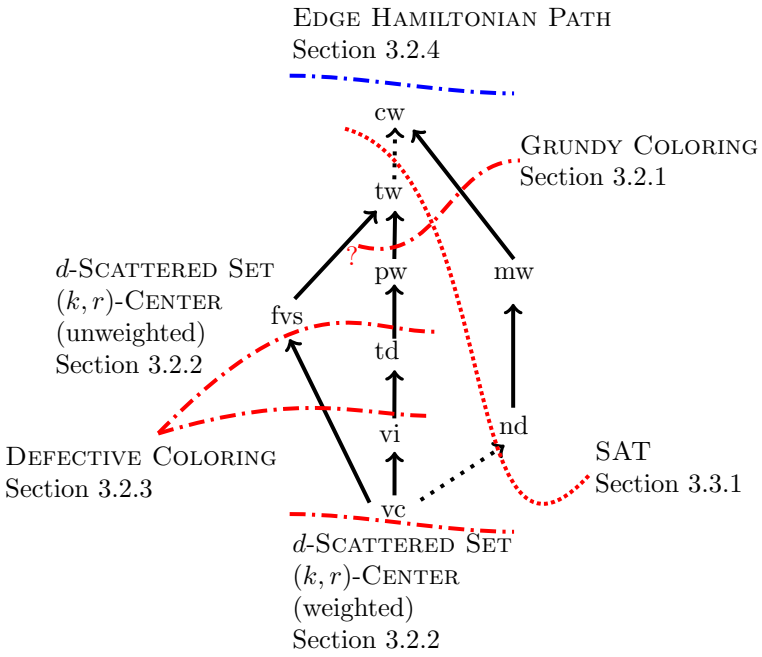


Figure 3.1: Summary of most of the results presented in this chapter. Red dotted lines indicate tractability borders, that is, the parameter transition where a problem first becomes W[1]-hard; for example SAT is FPT parameterized by treewidth (and hence vertex cover) but W[1]-hard by neighborhood diversity (and hence clique-width). Edge Hamiltonian Path is FPT for all considered parameters.

of SAT parameterized by neighborhood diversity in Section 3.3.1. We then follow this with several results tracing the difference between modular width and clique-width given in Section 3.3.2.

A summary of the most important results we present in this chapter can be seen in Figure 3.1, where we trace for each problem the point in the hierarchy where it becomes intractable. The reader can verify that we have made an effort to present a panorama of different problem complexities by selecting problems whose complexity transition takes place at different points of our hierarchy, including problems which are hard everywhere and a problem that is tractable everywhere.

3.1 Historical Overview

Let us now give a brief survey of "price of generality" results involving our considered parameters, that is, results showing that a problem is efficient for one parameter but hard for a more general one. As we explained, the overarching objective of this area is to understand which problems transition from being FPT to being W[1]-hard as we move to more general parameters (and why).

Clique-width vs Treewidth. The term "price of generality" was introduced in the very influential work of Fomin et al. [85]. This work and its follow-ups [86, 87], were the first to show that four natural graph problems (Coloring, Edge Dominating Set, Max Cut, Hamiltonicity) which are FPT for treewidth, become W[1]-hard for clique-width. In this sense, these works were the first to tackle head-on the algorithmic trade-off questions that we are concerned with in this chapter. We therefore begin our overview by revisiting problems which are known to separate treewidth from clique-width. Several other natural examples were later discovered, including: the problem of counting perfect matchings [48]; SAT parameterized by the clique-width of the incidence graph [152]; $\exists\forall$-SAT for the same parameter [135]; Orientable Deletion [104]; and

d-Regular Induced Subgraph [38] are all problems which are FPT by treewidth but become intractable for clique-width. This line of research has thus helped to illuminate the complexity border between the two most important sparse and dense parameters (treewidth and clique-width), by giving a list of *natural* problems distinguishing the two. (An artificial MSO_2-expressible such problem was already known much earlier [47, 128]).

Taking a look back, we can see that one common pattern among these problems is that they often involve objectives that have to do with selecting or deciding something about individual edges. This is quite direct for some problems (Edge Dominating Set, counting perfect matchings) and more subtle for others (Hamiltonicity can be seen as the problem of selecting a 2-regular connected spanning subgraph, which makes it akin to d-Regular Induced Subgraph). The fact that many such problems become intractable as we pass from treewidth to clique-width seems to confirm a piece of intuition that was discovered in the context of meta-theorems. Recall that Courcelle's celebrated theorem states that all MSO_2-expressible properties are FPT by treewidth [46], while its generalization to clique-width only covers MSO_1 [47]. The difference between the two logics is exactly whether one is allowed to quantify over sets of edges. Hence, individual problem examples seem to confirm the intuition that this weakness of clique-width to deal with edge properties does show up in the form of intractability. This informal observation, interesting though it may be, should not, however, be seen as a hard rule: we will present in Section 3.2.4 an example that goes against this intuition.

The hardness of SAT-related problems can also be indirectly explained by this argument, as it is probably an artifact of the application of clique-width on the incidence graph. As a reminder, the incidence graph has a vertex for each variable and each clause of a SAT instance and edges indicate which variables appear in which clauses. Unfortunately, this formulation ignores the *signs* of variables, so much useful information is lost when using clique-width as a parameter. The reason that this does not cause a problem when considering the treewidth of the incidence graph is again due to treewidth's flexibility in dealing with sets of edges (we can, for example, assume that the input edges are colored to retrieve the missing information and this does not affect treewidth but it completely ruins clique-width). A further argument in this direction is given in Section 3.3.1.

Thanks to these works, we can therefore say that the algorithmic difference between treewidth and clique-width is by now much more well-understood, with the handling of edge sets appearing as a key point. We note in passing that this has also motivated the study of other notions "between" treewidth and clique-width that fall outside the scope of this work, such as split-matching width [159].

Treewidth vs Pathwidth. Since we started our exposition by examining the contrast between treewidth and clique-width, the natural thing would be to continue towards pathwidth. However, to the best of our knowledge, until very recently no problem was known to separate treewidth and pathwidth algorithmically. We will present the first such problem in Section 3.2.1, therefore we do not have much more to say here.

Pathwidth vs Tree-depth. Let us then move on to the second most well-studied separation in our map of parameters, namely, the separation between pathwidth and tree-depth. Mixed Chinese Postman Problem was the first problem shown to separate the two parameters [103], followed by Min Bounded-Length Cut [66, 20], Integer Linear Programming (parameterized also by the maximum co-efficient) [97], Geodetic Set [122], (A, ℓ)-Path Packing [12], and (k, r)-Center and d-Scattered Set [119, 120] on unweighted graphs.

Taking a look at these results, what can we say about the algorithmic difference of pathwidth/treewidth and tree-depth? A first observation, that we will come back to later, is that problems seem to be hard for pathwidth (and treewidth) when the natural dynamic programming algorithm would need to store a natural number for each vertex in a bag. This is the case for most problems here: for example for length-bounded $s - t$ cuts we would need to store for each vertex in a bag its distance from s and t, and similar considerations go into d-Scattered Set and (k, r)-Center which are generalizations of Independent Set and Dominating Set to larger

distances.

The second observation is that in many of the problems separating pathwidth from tree-depth, the number we store is a *distance*. Intuitively, the reason such problems become FPT by tree-depth is that a graph with tree-depth d can have no path of length more than 2^d, hence bounding the tree-depth also indirectly bounds the largest distance we need to consider. As a result, when we bound tree-depth, problems for which the hardness stems from the need to store large distances "automatically" become FPT, even though we only execute the same DP algorithm that is available for treewidth.

Although it's nice to get fixed-parameter tractability in this way, this type of result feels a bit shallow, for two reasons: first, on a concrete level, because we are only able to bound the length of the longest path by 2^{td}, we generally tend to obtain algorithms of complexity of the form 2^{td^2}, which is rather high; second, philosophically it seems a little unsatisfying to declare a problem "easy" when we do not actually have a better algorithm for the tree-depth case, than we do for treewidth. It is therefore an interesting question whether this fixed-parameter tractability really tells us something or whether it is an artifact of the combinatorial bound on the longest path. We will investigate this in more depth in Section 4.1.2, where we confirm the intuition that this tractability for tree-depth is somewhat shallow by proving that for some problems the 2^{td^2} parameter dependence is optimal.

If we set aside the case of problems where the natural DP has to store a distance for each vertex, problems that distinguish tree-depth from pathwidth become more rare. In a sense, this seems to indicate that tree-depth and pathwidth are of similar algorithmic complexity (despite pathwidth being much more general), with the main difference stemming from the bound on the longest path evoked above.

Between Tree-depth and Vertex Cover. We now come to one of the widest gaps in our map, namely the gap between tree-depth and vertex cover. The distance between these two parameters is huge, among other reasons because for a fixed value k, the number of order n graphs with tree-depth k is exponential in n but the number of graphs of vertex cover k is only polynomial in n. In other words, graph of order n and vertex cover k can be described using $f(k) \cdot \log n$ bits only. This is a strong indication that the vast majority of problems in NP should be tractable parameterized by vertex cover, assuming of course that the input does not contain much more information in addition to a graph. This intuition is confirmed in practice and, as we see below, extremely few problems are known to be W[1]-hard parameterized by vertex cover (and these all allow some extra information in the input besides the graph). Hence, the vast majority of problems which are W[1]-hard for treewidth are lost in the transition from tree-depth to vertex cover (though this is not explicitly known in all cases, we summarize a few examples below). This situation has recently motivated the study of parameters between these two measures and hence vertex integrity is a notion that has only recently received attention [30, 65, 96, 98, 101]. Notice, however, that from an information-theoretic point of view vertex integrity is much closer to vertex cover than tree-depth, hence we would still expect the vast majority of problems to be easy for this parameter. Indeed, at the moment, few problems are known to separate vertex integrity from vertex cover. These include Steiner Forest and Graph Motif [101].

Below Vertex Cover. Finally, we reach a point in our main hierarchy where the parameter has become so restricted that we would expect the vast majority of problems to easily be FPT. This is indeed the case for vertex cover, but there do exist some notable exceptions. Of course, as we mentioned, in all W[1]-hardness results known for this parameter, the input consists of a graph supplied with some additional information. If that were not the case, it would probably be impossible (from an information-theoretic point of view) to use such problems to encode instances of k-Clique (which is an implication of W[1]-hardness).

The small list of problems in this category which are known to be W[1]-hard even for vertex cover (while being in XP for treewidth) includes: List Coloring [75] which was the first such problem to be discovered; CSP for the vertex cover of the dual graph [160]; (k, r)-Center,

d-Scattered Set for weighted graphs [119, 120]; Min Power Steiner Tree [121] also on weighted graphs; Min Maximum Out-Degree Orientation [101]; and SAT parameterized by the vertex cover of the conflict graph (the graph that has a vertex for each clause and edges represent the fact that two clauses disagree on a variable) [99]. Observe that the intuition that the input is supplied with extra information is confirmed in all cases, hence one could argue that the reason these problems turn out to be intractable for vertex cover is that the graph and its structure are not as relevant to the complexity of the problem. Be that as it may, it's still important to understand why in some rare cases parameterizing by vertex cover is not enough to lead to tractability, especially when the input is only a graph plus a little more information (for example, some weights on the edges), because vertex cover was initially studied in this context under the promise that it would make almost everything tractable [76].

3.1.1 More Problems Hard for Treewidth

Now that we have taken a first look through the hierarchy of parameters around treewidth let us cast a wider net. For the sake of completeness, we will attempt to give a semi-exhaustive list of problems which are known to be W[1]-hard for treewidth, gathering such results which are scattered in the literature and taking a look at what they imply for our purposes. One particular point of interest is that, because treewidth is by far the most famous structural parameter, many of these results are only claimed to prove hardness for treewidth, when in fact they often imply hardness for more restricted parameters. Pathwidth in particular can be verified to be a parameter that has the same complexity status as treewidth for all the following problems. We therefore make an attempt to summarize some implications of the corresponding reductions for more restricted parameters, even when this is not explicitly claimed in the corresponding publication.

A list of problems known to be W[1]-hard by treewidth thus includes the following (in addition to the problems cited in the previous section):

1. Precoloring Extension and Equitable Coloring are shown to be W[1]-hard for both tree-depth and feedback vertex set in [75] (though the result is claimed only for treewidth). This is important, because Equitable Coloring often serves as a starting point for reductions to other problems. A second hardness proof for this problem was recently given in [54]. These two problems are FPT by vertex cover [80].
2. Capacitated Dominating Set and Capacitated Vertex Cover are W[1]-hard for both tree-depth and feedback vertex set [58] (though again the result is claimed for treewidth).
3. Min Maximum Out-degree Orientation on weighted graphs was shown W[1]-hard by tree-depth and feedback vertex set [164]; this was improved to W[1]-hardness by vertex cover in [101].
4. General Factors is W[1]-hard by tree-depth and feedback vertex set [161].
5. Target Set Selection is W[1]-hard by tree-depth and feedback vertex set [19] but FPT for vertex cover [150].
6. Bounded Degree Deletion is W[1]-hard by tree-depth and feedback vertex set, but FPT for vertex cover [23, 96].
7. Fair Vertex Cover is W[1]-hard by tree-depth and feedback vertex set [124].
8. Fixing Corrupted Colorings is W[1]-hard by tree-depth and feedback vertex set [25] (reduction from Precoloring Extension).
9. Max Node Disjoint Paths is W[1]-hard by tree-depth and feedback vertex set [71, 82].
10. Defective Coloring is W[1]-hard by tree-depth and feedback vertex set [18].
11. Power Vertex Cover is W[1]-hard by tree-depth but open for feedback vertex set [3].

12. Majority CSP is W[1]-hard parameterized by the tree-depth of the incidence graph [55].
13. List Hamiltonian Path is W[1]-hard for pathwidth [145].
14. L(1,1)-Coloring is W[1]-hard for pathwidth, FPT for vertex cover [80].
15. Counting Linear Extensions of a poset is W[1]-hard (under Turing reductions) for pathwidth [68].
16. Equitable Connected Partition is W[1]-hard by pathwidth and feedback vertex set, FPT by vertex cover [70].
17. Safe Set is W[1]-hard parameterized by pathwidth, FPT by vertex cover [14].
18. Matching with Lower Quotas is W[1]-hard parameterized by pathwidth [9].
19. Subgraph Isomorphism is W[1]-hard parameterized by the pathwidth of G, even when G, H are connected planar graphs of maximum degree 3 and H is a tree [144].
20. Simple Comprehensive Activity Selection is W[1]-hard by pathwidth [69].
21. Defensive Stackelberg Game for IGL is W[1]-hard by pathwidth (reduction from Equitable Coloring) [10].
22. Directed (p, q)-Edge Dominating Set is W[1]-hard parameterized by pathwidth [13].
23. Maximum Path Coloring is W[1]-hard for pathwidth [127].
24. Unweighted k-Sparsest Cut is W[1]-hard parameterized by the three combined parameters tree-depth, feedback vertex set, and k [114].
25. Graph Modularity is W[1]-hard parameterized by pathwidth plus feedback vertex set [146].

The lesson of this list is that very often reductions which are claimed to show that a problem is hard for treewidth, actually show more: such problems are almost always also hard for pathwidth, and very often also hard for tree-depth and feedback vertex set. This is because such hardness reductions construct grid-like graphs, of dimensions $k \times n$ or $k \times k$. The pathwidth of such graphs can therefore also by bounded by a function of k (though to extend the reduction to tree-depth we must avoid $k \times n$-grid-like constructions). For the problems where hardness by tree-depth is unclear (for example because the reduction relies on the construction of a long path) it would be interesting to clarify their status with respect to this parameter.

3.2 The road through Treewidth

Having given a general taste of price-of-generality research as it pertains to treewidth-related parameters, let us now highlight a few results involving the author's work in this area. We begin by revisiting the complexity of a variation of Coloring called Grundy Coloring in Section 3.2.1; the interest of this problem is that it is the first natural problem to differentiate pathwidth from treewidth. In the following two sections we give some examples of natural problems which turn out to be intractable for treewidth because of the existence of natural numbers in any reasonable formulation of a dynamic programming algorithm. We focus mostly on two generalizations of Independent Set and Dominating Set given in Section 3.2.2, as these turn out to be hard even for vertex cover, while in Section 3.2.3 we view a few other, more tractable examples. Finally, in Section 3.2.4 we move higher in the hierarchy to revisit the separation of treewidth with clique-width. Counter-intuitively, we give a problem that does *not* separate the two parameters, despite being superficially similar to problems that do, and discuss the algorithmic tricks that lead to this and how they can be of further use.

3.2.1 Grundy Coloring – Treewidth vs Pathwidth

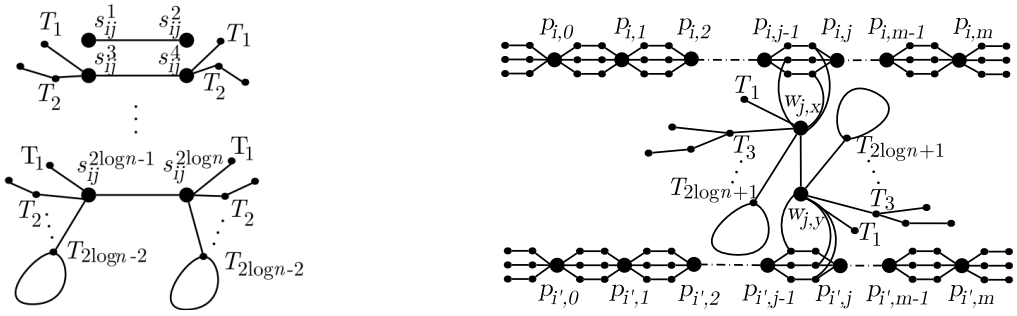
The results of this section come from the recently published work [16]. We will focus on a coloring problem called Grundy Coloring which can be defined as follows: suppose we have a graph G on which we want to apply the First-Fit coloring algorithm, that is, the algorithm which considers the vertices of $V(G)$ one by one in some order and assigns to each vertex the lowest possible color that has not already been given to one of its neighbors. The objective of Grundy Coloring is to find the *worst* coloring that may have been produced by this algorithm, or equivalently, the ordering of $V(G)$ that will lead the First-Fit algorithm to use as many colors as possible.

Even though this is a well-studied problem, one cannot help but feel that it is a little artificial. Why are we then so interested in Grundy Coloring? From the point of view of this work, the main point of interest of this problem is that it exhibits a behavior which, to the best of our knowledge, has not been witnessed in any other natural problems before. Namely, it was shown in [16] that even though Grundy Coloring is FPT by pathwidth (this follows from previous results of [64] and [166]), this fixed-parameter tractability cannot be extended to treewidth, where the problem is W[1]-hard. Hence, our interest stems from the curious complexity status of this problem and we would like to understand what is it that makes this problem behave differently for the two parameters.

Before we give more details about Grundy Coloring, let us explain why it is somewhat remarkable for such a problem to have different complexity status for the two parameters. Typical treewidth-based algorithms work using dynamic programming and the dynamic program is almost always the same whether one has at hand a tree or a path decomposition. Hence, when dealing with any problem in the context of this technique, the only extra difficulty we have when we want to parameterize by treewidth (instead of pathwidth), is that we need to somehow handle Join nodes – otherwise a tree decomposition without such nodes is just a path decomposition. But because Join nodes are nodes with two children with identical vertices, for most problems handling them can easily be done with an overhead that is at most polynomial in the size of the dynamic programming table. For this reason, for a very long time it was believed that the main drawback of treewidth over pathwidth was this extra polynomial overhead, which manifested itself in the form of worse constants: for example Dominating Set could be solved in 3^{pw} but only in 9^{tw}. Nevertheless, the discovery of fast subset convolution techniques succeeded in closing essentially all of these gaps [168]. As a result, at the moment it is not only the case that all major problems have the same complexity status with respect to these parameters (FPT vs. W-hard), but even that the best algorithms have the exact same running times. If we think about it, this is quite surprising! Pathwidth is a much more restrictive definition than treewidth, so it would be quite strange if extending the class of graphs we are willing to handle so much could cost us almost nothing in terms of complexity.

Because of this situation the existence of problems which do have different complexity for the two parameters becomes intriguing. One could of course attempt to cook up an artificial such problem: "Given a graph G, if $\mathrm{pw}(G) \leq 5$ answer Yes, otherwise solve the Halting Problem on this extra input". Obviously, making up such a problem is possible (since we know that the inclusion relation between the two parameters is strict) but would tell us little of interest. It was therefore particularly surprising that Grundy Coloring turned out to be such a problem, without having been defined for this reason (indeed Grundy Coloring has been an object of study since before treewidth and pathwidth became "structural parameters").

How does it work? Without going into technical details, let us try to give an informal overview of why Grundy Coloring has the aforementioned behavior. The work of Telle and Proskurowski supplied an algorithm for this problem with parameter dependence $2^{O(\mathrm{tw}\cdot\Gamma)}$, where Γ denotes the number of colors. Unfortunately, such a dependence is not FPT by treewidth, since Γ may increase arbitrarily with n, even for trees. However, it is also known that the number of colors can be upper-bounded by $\Gamma = O(\mathrm{tw} \cdot \log n)$ (and in particular, trees have Γ upper-bounded by $\log n$). As a result, the algorithm of Telle and Proskurowski has an XP-type running time of $n^{O(\mathrm{tw}^2)}$. As we will later show that the problem is W-hard, one could reasonably say that this is essentially



(a) Vertex Selection gadget. (b) Propagators.

Figure 3.2: The basic gadgets of the reduction for Grundy Coloring On the right the propagator gadgets from a $k \times n$ grid which allows to transmit information throughout the instance.

the best algorithm parameterized by treewidth.

An interested reader may now be expecting the description of a more clever algorithm parameterized by pathwidth – we have after all promised that the problem gets easier for this parameter. This is, however, not the case. Instead, the best algorithm known for this case is still the algorithm of Telle and Proskurowski, this time applied to a path decomposition. The missing key ingredient is an inequality, shown in [64], which states that for all graphs $\Gamma \leq 8\text{pw}$. As a result, in order to obtain an FPT algorithm parameterized by pathwidth we do not need to do anything more clever than simply running the treewidth-based algorithm for appropriate values of Γ.

We therefore reach the conclusion that a key element of the reduction establishing that Grundy Coloring is W[1]-hard parameterized by treewidth will be the use of gadgets that force Γ to values around $\log n$, because this is what constitutes the difference in complexity between the two parameters for this problem. Indeed, without going into details we invite the reader to take a look at Figure 3.2 which is taken from [16]. The idea of this construction is that the whole graph will roughly resemble a $k \times n$ grid, where in part 3.2b we depict two of the rows of this grid. Between two consecutive vertices (propagators) of this grid we place a selection gadget and the task of the propagators is to conserve the selection throughout the row.

Clearly this description is rather vague (and we invite the interested reader to take a look at the details in [16]). However, one thing that should be visible is that this construction should have both treewidth and pathwidth upper-bounded by a function of k, because a grid of size $k \times n$ has pathwidth and treewidth k. Indeed, based on this construction we can prove that a more general version of Grundy Coloring, where some vertices are pre-colored, is W[1]-hard even for pathwidth! The reason this does not in the end give W[1]-hardness for pathwidth (which would contradict the algorithmic arguments we previously cited) is the rather innocent fact that pre-coloring vertices cannot be done in the context of bounded-pathwidth graphs. Indeed, as we said, we need to ensure that some vertices end up with colors above $\log n$. In the context of bounded-pathwidth graphs, this is impossible by the result of [64]; however, in the context of bounded-treewidth graphs this is not a problem, as even a polynomially-sized tree can have $\Gamma = \log n$, and this is why we attach various trees to the gadget of 3.2a.

What does this teach us? One could say that Grundy Coloring is an interesting specimen because in a sense it is the exception that confirms the rule. In our case, the rule is that for the vast majority of problems, the complexity is the same whether we parameterize by treewidth or pathwidth, hence the best algorithm for treewidth is also the best algorithm for pathwidth. Grundy Coloring would superficially seem to contradict this, as its complexity status changes between the two parameters. But looking under the hood we realize that it does not! The status change is due to an unrelated combinatorial fact: on graphs of bounded pathwidth we get a better bound on the objective function "for free". As a result, even though we succeed with this problem to show that natural problems separating treewidth from pathwidth do exist, we do not manage

to uncover any new algorithmic properties of pathwidth. It is therefore an interesting and still open question to find a natural problem where a truly pathwidth-specific algorithmic technique would permit us to separate pathwidth from treewidth.

3.2.2 Distance-based Generalizations of Independence and Domination

In this section we present some results from [119] and [120], which also appeared in the thesis of Ioannis Katsikarelis (co-supervised with Vangelis Paschos). We will focus on (k, r)-Center and d-Scattered Set. In the former problem we are asked to select k vertices of a graph so that every other vertex is at distance at most r from one selected; in the latter problem we want to select a maximum set of vertices which are pairwise at distance greater than d. Clearly, for $r = 1$ and $d = 1$ these problems are k-Dominating Set and Independent Set respectively.

The reason we will focus on these problems here is that Independent Set and Dominating Set are some of the most basic problems one learns to solve using treewidth, so it makes sense to investigate their generalizations to see at which point treewidth-based techniques break down. As we will see, these problems continue to be solvable using treewidth, however, their complexity depends crucially on r and d. In particular, these problems witness a phenomenon that we have already talked about: the hardness of incorporating natural numbers into the standard treewidth-based dynamic programming toolbox. As such, these problems do not turn out to be special – we will see that this phenomenon is widespread and many problems become intractable if we are forced to remember a number for each vertex in a bag of a decomposition. Their interest is rather in the way that they clearly exhibit this behavior, even if we choose the much more restrictive parameterization by vertex cover.

Before we go on to present the main results of this section, let us take a moment to think what we expect to see. For Independent Set the standard DP algorithm has complexity 2^{tw}. If we generalize to distance $d \geq 2$ we would have to remember for every vertex in the bag its distance from the closest selected vertex. This has $d + 1$ choices (from 0 to d), so a natural bound would be $(d + 1)^{\mathrm{tw}}$. Similarly, for (k, r)-Center we need to remember for every vertex its distance to its closest dominating vertex, but also whether this vertex has already been dominated already or not. This would give $2(r + 1)$ choices, but if we take into account that a selected vertex is automatically dominated, this leads to a $(2r+1)^{\mathrm{tw}}$ algorithm, which is in accordance with the 3^{tw} complexity of Dominating Set. Indeed, the running times we guess with this simple reasoning are correct [37, 120], even if obtaining them is not entirely trivial (in particular, one needs to use fast subset convolution techniques to handle Join nodes).

The type of performance we thus obtain in the "obvious" way is not entirely satisfactory: the complexity of both problems seems to deteriorate rather quickly as we increase the distance parameters. The main result of this section is that this is inevitable, as both problems are W[1]-hard parameterized by treewidth. Hence, the phenomenon we are witnessing is as promised: as soon as we are forced to insert natural numbers into our dynamic program, things go wrong.

How bad are things? Let us begin by considering the two problems in their full generality: suppose that the input graph is edge-weighted (that is, every edge is assigned a positive integer weight) and distances are calculated using edge weights (that is, the length of a path is the sum of weights of its edges). In this case, things are surprisingly bad: both problems are W[1]-hard when parameterized even by vertex cover, as shown in [119] for (k, r)-Center and in [120] for d-Scattered Set.

To give the reader an idea of how these reductions work, we refer to the sketch of Figure 3.3, taken from [120]. Without going into the technical details, the main idea of the proof is that we construct k blocks of n vertices (shown as rectangles) which will represent our choice of k vertices in a given instance of k-Clique. The problem now is how this choice can be communicated to the rest of the graph via a vertex cover of size $O(k)$. This is achieved by attaching to each block two vertices (a_i, b_i in the figure) connected to the choice block in such a way that from the distance of a_i, b_i to the selected vertex we can infer which vertex was selected (more precisely, vertices which are closer to a_i are farther from b_i). In this way, the thing we need to check in the rest of the construction is whether the distances of the $2k$ vertices a_i, b_i from their closest selected vertices are

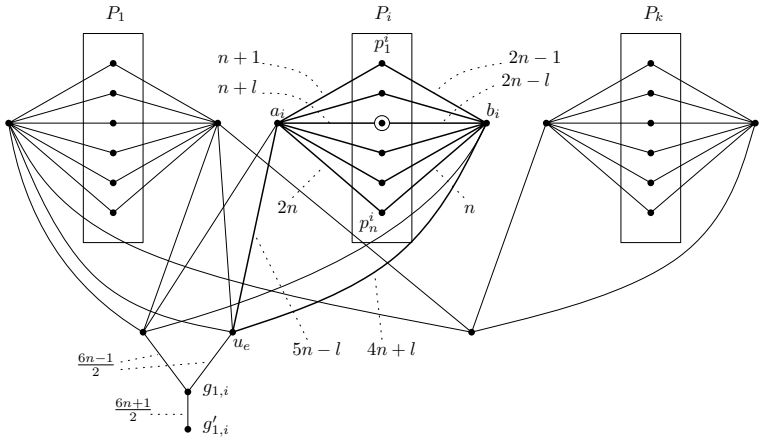


Figure 3.3: Sketch of the W[1]-hardness of d-Scattered Set parameterized by vertex cover.

correct. Even though we skip in this description the construction of the gadgets that verify these distances, this intuition captures the basic idea of the reductions for both problems and matches exactly the intuition of the treewidth-based algorithm we described above! In particular, notice that to solve this particular class of instances one would have to "guess" a number for each of the $O(k)$ vertices a_i, b_i, namely the distance of each such vertex from its closest selected vertex. The reduction proves that this is essentially best possible.

This elegant hardness argument is rather devastating, because it shows that both problems are already hard for an extremely restricted parameter. Indeed, the reduction shows that even if we restrict ourselves to graphs where deleting a small set of k vertices destroys all edges, we still have to brute-force the solution on these k vertices. However, one could argue that the reason for this hardness is that we are allowing ourselves to use weights on the edges. Do things become more reasonable for unweighted graphs? The answer is yes and no. On the one hand, we can simply replace all weighted edges with paths of appropriate lengths. This will of course increase the vertex cover, but will leave many other parameters unchanged. As a result, even the unweighted versions of this problem are W[1]-hard parameterized by both pathwidth and feedback vertex set.

On the other hand, there is a simple argument to be made (that we already alluded to in our discussion on tree-depth), which allows us to obtain tractability for a parameter more general than vertex cover. As we have mentioned, a graph of tree-depth k cannot have a path of length more than 2^k. Therefore, the reasonable values of the distance parameters d, r of our problem are bounded from above by 2^{td}. Simply plugging in this value to our algorithms gives an FPT algorithm parameterized by tree-depth, running in $2^{O(\mathrm{td}^2)}n^{O(1)}$. It is interesting to note that this situation is very reminiscent of what we saw in Section 3.2.1 regarding Grundy Coloring and pathwidth, namely, that tractability is obtained not via a better algorithm but via an unrelated combinatorial statement that allows us to bound the optimal. So, is this truly a more tractable case of the problem? The complexity status seems to suggest so, but as we will see in Section 4.1.2 the 2^{td^2} complexity is best-possible (i.e. it cannot be improved to $2^{o(\mathrm{td}^2)}$). This seems to imply that this problem is not truly easier and we were just lucky to be able to bound the value of the optimal.

What have we learned? (k, r)-Center and d-Scattered Set are two of the most natural problems that exhibit a more widely observed behavior: a natural DP involving integers which blocks us from obtaining an FPT algorithm; and a W-hardness argument showing that this is best possible. Digging deeper, these problems are interesting specimens because they are among the rare examples of problems hard for vertex cover (if we allow edge weights); and they separate pathwidth from tree-depth (for unweighted graphs). One could wonder if the particular complexities we have achieved for these problems are best possible, since the W-hardness results do not give us sufficient

granularity to answer this type of question. The reader need not worry, as we will revisit these problems in the next chapter!

3.2.3 Dynamic Programs with Numbers

In Section 3.2.2 we discussed two problems which become W-hard by treewidth because of the appearance of integers in the natural dynamic programming algorithm. We mentioned in passing that this is a generalized phenomenon: it seems quite common that whenever, for some reason or another, we need to store for each vertex of a bag a number to characterize a partial solution, this leads to a complexity of the form n^{tw}. In this section we want to further justify the claim that this is a wide-spread phenomenon and that such problems form a major class of all problems which are W-hard by treewidth. To this end we present four examples that have appeared in recent works involving the author. Our goal here is not to bore the reader with the details of various reductions but to give a flavor of what a typical problem looks like when it is W-hard parameterized by treewidth. The four problems we present will therefore highligh different "reasons" why numbers may creep into a treewidth-based algorithm, beyond the case of distances (which was covered in Section 3.2.2). The problems we present are Min Stable Cut [126], Defective Coloring [18], Safe Set [14], and Power Vertex Cover [3].

Minimum Stable Cut In this problem we are given a graph G and are asked to find a cut, that is a partition of $V(G)$ into two parts, which is stable, in the sense that every vertex has at least half of its neighbors on the other side, and has minimum size. In other words, we are looking for the smallest cut that cannot be increased by flipping a single vertex. The natural DP would store for each vertex in the bag how many neighbors this vertex currently has on its own side, giving a complexity of the form Δ^{tw}. It was shown by the author in [131] that this dependence on Δ cannot be avoided, as the problem is already W-hard parameterized by pathwidth. Hence, Minimum Stable Cut is a nice example of a problem which becomes W-hard by treewidth because we have to store *degree* information for each vertex. In Chapter 4 we will revisit this problem, which has a very interesting complexity profile when the input graph is weighted.

Defective Coloring. Consider the following generalization of Coloring: we are given a graph G, a number of colors c, and a degree bound Δ^* and are asked to partition $V(G)$ into at most c sets so that each set induces a graph of maximum degree at most Δ^*. Standard Coloring is just the special case of this problem for $\Delta^* = 0$. The reason this problem is in our list is that if we attempt to formulate a natural DP algorithm using treewidth, we would need to remember for each vertex in a bag its color (c choices), as well as its degree inside its color class (Δ^*+1 choices). Hence, the natural algorithm has parameter dependence $(c\Delta^*)^{\mathrm{tw}}$. The fact that c appears in this running time is not a problem, as we know that $c \leq \mathrm{tw}$ even for standard coloring. However, the degree bound only leads to an XP algorithm.

This problem was studied in [18] (see also [17]), where it was shown that this dependence on Δ^* is inevitable, as parameterizing the problem only by treewidth is not sufficient to give fixed-parameter tractability. More strongly, it was shown that Defective Coloring is W[1]-hard parameterized by both tree-depth and feedback vertex set, even for only 2 colors. Surprisingly, for feedback vertex set, 2-coloring is the only W-hard case, as once more colors are available the problem becomes FPT with a simple win/win argument! The same argument allows us to show fixed-parameter tractability parameterized by vertex cover: if $\Delta^* \leq \mathrm{vc}$ we can use the standard algorithm, which is FPT; while if $\Delta^* > \mathrm{vc}$ we can use one color for the vertices of the vertex cover and another for the rest of the graph.

The structural complexity of Defective Coloring is thus well-traced by [18] (and [17] which shows that the problem is NP-hard on graphs of constant modular width), though parameterizing the problem by vertex integrity is an interesting open question. Defective Coloring is therefore another nice example where the reason the problem becomes W-hard by treewidth is that we have to remember for each vertex in the bag its *degree* in the solution.

Safe Set A safe set S of a graph is defined as follows: we say that $v \in S$ is safe if the size of the component containing v in $G[S]$ is at least as large as the size of any component containing a vertex of $N(v) \setminus S$ in $G - S$. A set is safe if all its vertices are safe. In other words, the goal here is to select a set so that no component of the selected set is adjacent to a larger component made up of non-selected vertices. This problem is vaguely related to vertex integrity, since the goal is to find a small separator S which will break down the graph into even smaller parts. The natural DP algorithm for this problem using treewidth would of course remember for each vertex whether it belongs in S, but also the size of its component. As a result, at least an integer for each vertex needs to be stored, leading to a complexity of the form n^{tw}. In [14] it was shown that this is inevitable, as the problem is W[2]-hard parameterized by pathwidth. As a result, Safe Set is a nice example where the reason the problem becomes hard by treewidth is that we have to remember for each vertex the *size of its component*.

Power Vertex Cover Finally, we consider a problem where the reason we need to remember a number for each vertex in a bag comes directly from the problem definition. In Power Vertex Cover we are given a graph G and a positive integer demand w_e for each edge $e \in E(G)$. The goal is to assign a power level p_v to each vertex $v \in V(G)$ so that each edge has at least one endpoint whose power is greater or equal to the demand of the edge. Clearly, this generalizes Vertex Cover, which is the case where all demands are 1. The naive DP algorithm for this problem using treewidth would consider all possible power levels for each vertex in a bag. Unfortunately, it was shown in [3] that this is best possible (even if weights are polynomially bounded), as this problem is in fact W[1]-hard parameterized by tree-depth. It remains an interesting open problem whether this problem is FPT by feedback vertex set.

3.2.4 The Exception that Confirms the Rule – EHP

In the previous section we discussed problems which are W[1]-hard parameterized by treewidth, trying to understand why and where this hardness arises. Here, we take a different approach and consider a problem which, somewhat surprisingly, turns out to be easy for treewidth, and even for clique-width. The problem in question is Edge Hamiltonian Path and these results are taken from [133].

Given a graph $G = (V, E)$, the Edge Hamiltonian Path problem (hereafter EHP) asks whether the line graph of G is Hamiltonian, that is, whether there exists an ordering of E so that any two consecutive edges in the ordering share an endpoint. Notice that this is different from an Eulerian path, where we need to enter an edge from one endpoint and exit from the other, as it is allowed, for example, to traverse all edges incident on a vertex one after the other.

Before we sketch the results about EHP, let us take a look at what we know, to decide what we can expect for this problem. The most relevant problem seems to be Hamiltonian Path, which is known to be FPT parameterized by treewidth but W[1]-hard parameterized by clique-width [86]. One could perhaps expect EHP to have a similar status. However, a little reflection reveals the intuitive reason Hamiltonian Path is FPT for treewidth but not clique-width is that it can be recast as an edge-selection problem, rather than an ordering problem, and is hence MSO_2-expressible. The same cannot be said for EHP, where we seek to *order* the edges, not select a subset of them. Furthermore, given a graph G of treewidth k, one could try to solve EHP by computing the line graph of G and applying an algorithm for Hamiltonian Path on that graph. However, the line graph of a graph of treewidth k only has its *clique-width* (and not treewidth) bounded by a function of k (indeed, such a graph may contain large cliques), so this approach would not lead to an FPT algorithm, even for EHP parameterized by treewidth.

Given the above, it is not even clear whether the problem is FPT parameterized by treewidth, let alone clique-width. Indeed, Demaine et al. [56] explicitly posed it as an open problem whether this problem is FPT parameterized even by vertex cover! It was thus somewhat surprising that [133] managed to show that the problem is FPT for the whole hierarchy of parameters we consider.

FPT for Treewidth. First, let us see why EHP turns out to be tractable by treewidth. For this, we rely on a characterization of edge hamiltonian paths which was already established in the 1960s by Harary and Nash-Williams [105]: a graph G has an edge hamiltonian path if and only if there exists a vertex cover S of G with the property that $G[S]$ contains an Eulerian spanning subgraph, that is, a spanning subgraph that contains an Eulerian path. Recall that deciding if a graph is Eulerian (that is, whether it contains an Eulerian path) comes down to checking whether it is connected and whether the degrees of its vertices have the correct parity. Thanks to this characterization it is not too hard to arrive at the fixed-parameter tractability of EHP by treewidth: we observe that all the necessary properties are MSO_2 expressible, with the small caveat that we have to check the parity of the degree of selected vertices (but this can be handled by standard counting extensions of MSO_2). Given this knowledge, we can then come up with an explicit algorithm, solving the problem in $\mathrm{tw}^{O(\mathrm{tw})}n^{O(1)}$.

From Treewidth to Clique-width. The previous paragraph gives an intuitive understanding why EHP turned out to be FPT by treewidth. Nevertheless, one may be worried that clique-width would be a bridge too far – after all Hamiltonian Path is lost in this transition. Interestingly, this is not the case, and this was established in [133] via a structural tool due to Gurski and Wanke that may be of further interest[1]. The tool in question is a theorem that states, informally, that if a graph has small clique-width and does not contain any large complete bipartite subgraph, then the graph actually has small treewidth [102].

The main new idea of [133] was then to take in the input a graph of small clique-width and then *edit* it in a way that does not change the answer to EHP but eliminates large complete bipartite subgraphs one-by-one (of course in a way that preserves the clique-width of the graph). The way to do this was to locate such subgraphs (corresponding to the Join operations of a supposed clique-width expression) and replace them with a small gadget that preserves edge hamiltonicity. Repeating this process, which at each step reduces the size of the graph, must eventually result in a graph of small treewidth, on which the algorithm of the previous paragraph can readily be applied. Hence, the FPT algorithm for clique-width only consists of a pre-processing step which eliminates bi-cliques, followed by the previous algorithm, which can be invoked thanks to the inequality of Gurski and Wanke.

What have we learned? The lesson of this problem, which turns out to be easier than expected, is that structural combinatorial results can sometimes be very useful in translating algorithms for a restricted parameter to a more general one. The key idea here was the inequality of Gurski and Wanke which establishes that small clique-width and no complete bipartite graphs imply small treewidth. This seems to be an idea that can be further exploited, and indeed we will further exploit it in Chapter 5 to leverage (approximation) algorithms that work for treewidth into algorithms that work for clique-width.

3.3 The road around Treewidth

In the previous section we focused on the structural parameters in the hierarchy of treewidth, since these parameters have attracted the most interest in the literature so far. We now move on to take a look at an alternative, less-studied hierarchy of parameters which has the same starting and ending points: in this section we consider neighborhood diversity and modular width. In particular, in section 3.3.1 we revisit one of the problems that separate clique-width from treewidth, namely SAT parameterized by the width of the incidence graph, and show that this problem is already hard for the much more restricted parameter of neighborhood diversity. We then discuss what this means about the hardness of SAT for clique-width. Later, in Section 3.3.2 we examine modular width as a parameter which allows us to obtain tractability for some problems which are hard for

[1]Indeed, we will have the occasion to reuse this tool in Sections 5.1.2 and 5.2.

clique-width (or even for treewidth) by leveraging ideas which were originally applied to the much more restricted setting of neighborhood diversity.

3.3.1 SAT and Neighborhood Diversity

The main result of this section, which is taken from [55], will be to establish the hardness of SAT under a seemingly very restrictive parameterization. SAT is of course one of the most widely studied problems in theoretical computer science and also in parameterized complexity theory, so the application of graph structure measures to SAT-solving has long been a field of active research. In order to put the structural graph parameters we normally study into use for this problem, we of course need to define some graph structure that represents the input – recall that in SAT we are normally given as input a boolean formula in CNF. There are several ways to derive a graph structure from such a formula, but here we will focus on what is perhaps the most natural such way, which is called the incidence graph. The incidence graph of a formula ϕ contains a vertex for each variable and each clause of the formula and an edge between each clause and all the variables that appear in this clause. As a result, the incidence graph represents much of the structure of the input formula, with the minor caveat that the information of whether a variable appears positive or negative inside a clause is not represented. This seemingly minor detail will turn out to be of major interest.

Given this graph structure, it is now natural to ask if SAT becomes tractable when the incidence graph has some desirable properties, and in particular, if SAT is FPT parameterized by the treewidth or clique-width of the input graph. A simple dynamic programming algorithm can help us obtain a $2^{\text{tw}}n^{O(1)}$ algorithm for SAT[2]. On the other hand, if we parameterize by clique-width, only an XP algorithm is known [163] and it was shown in [152] that this is best possible, because SAT is W[1]-hard parameterized by the clique-width of the incidence graph. This essentially settles the problem and shows that the extra generality of graphs of bounded clique-width makes the problem intractable.

Despite this seemingly clear answer, we revisit the problem trying to understand why it becomes so much harder when transitioning from treewidth to clique-width. Our result indicates that the W[1]-hardness for clique-width is not a result of the combination of the tree-like structure of treewidth with some complete bipartite graphs; rather, the existence of some much simpler complete bipartite graphs is already sufficient to render the problem hard, even without the tree-like structure. Indeed, we show that SAT is W[1]-hard parameterized by the neighborhood diversity of the incidence graph, which in itself is rather surprising because graphs of bounded neighborhood diversity have very low information content (like graphs of vertex cover k, they can be described using $f(k) \cdot \log n$ bits). Of course, this result improves upon the hardness given in [152].

In order to gain some intuition about what is going on, let us revisit the reduction given in [55]. Suppose we are given an instance of k-Multi-Colored Independent Set, that is, a graph $G = (V, E)$, where V is given partitioned into k cliques $V_1, \ldots, V_k$ and we are asked whether an independent set of size k exists. This problem is known to be W[1]-hard [49]. Assume without loss of generality that $|V_i| = n$, the vertices of V_i are numbered $v_{i,0}, \ldots, v_{i,n-1}$, for all $i \in [k]$ and n is a power of 2. We construct a formula which uses $k \log n$ variables, partitioned into k sets, $X_1, \ldots, X_k$. For each pair $i, j \in [k]$ with $i \neq j$ we do the following: we consider each edge between V_i and V_j, say the edge $v_{i,\alpha}v_{j,\beta}$ and write α, β in binary using $\log n$ bits for each. We add to the formula a clause that contains all variables of $X_i \cup X_j$ and no other variable and will be falsified if and only if the assignment of X_i read as a binary number is equal to α and the assignment of X_j read as a binary number is equal to β (this is always possible as a clause has a unique falsifying assignment).

The reduction we sketched in the previous paragraph constructs a formula that can only be satisfied by giving to each X_i an assignment that encodes the index of the vertex of the independent

[2]Even though getting the constant in the base down to 2 requires fast subset convolution techniques, achieving a $2^{O(\text{tw})}n^{O(1)}$ complexity is an easy exercise.

set selected from V_i. The key observation about this construction now is that the neighborhood diversity of the incidence graph is only $O(k^2)$, because the clauses can be partitioned into $\binom{k}{2}$ groups, where all clauses of each group contain the same variables (all of $X_i \cup X_j$, for a pair $i, j \in [k]$), the only difference being in the sign with which variables appear in each clause. We therefore have a valid FPT reduction and we have proved that SAT is W[1]-hard not just parameterized by clique-width, but even by neighborhood diversity.

What happened? Something unusual seems to be happening with this reduction. We have constructed an instance of SAT that only has a logarithmic number of variables overall and, from the point of view of the structure of the incidence graph is extremely homogeneous. In fact, we have managed to encode a k-Clique instance into an instance of SAT where the structure of the graph can be encoded using only $O(k^2 \log n)$ bits, as is the case with graphs of neighborhood diversity k. More strongly, we can observe that the incidence graph we construct depends only on the *size* of the input graph G (more precisely, the number of edges between each pair V_i, V_j) and is completely disconnected from the structure of the input graph! Indeed, given two distinct graphs G_1, G_2 which have the same size and the same number of edges between any two sets of vertices in their partitions, our reduction would construct two distinct formulas but the same incidence graph. This is a strong indication that looking at the incidence graph, which "forgets" the sign of the appearances of variables, is a poor way to capture the structure of a formula.

In this light, it is not surprising that SAT is already hard when the unsigned incidence graph has an extremely restricted structure. In retrospect, perhaps what is a bit surprising is that it is at all possible to extract any algorithmic usefulness from the structure of this graph! In the case of treewidth, this can be understood, because if we add back the erased information to the incidence graph (and make it signed), this does not increase the treewidth. However, the (limited) tractability of SAT when parameterized by the clique-width of the unsigned incidence graph seems rather striking and fragile. This is further confirmed by the fact that for generalizations of SAT, such as notably $\exists\forall$-SAT, even this XP tractability is lost, and the problem becomes NP-hard for constant values of the parameter. Hence, perhaps the lesson to be taken here is that when the graph structure does not contain enough information to encode the whole input, structural graph parameters can mislead us into thinking that the input is simpler than it is. In particular, for dense parameters such as clique-width, one has to be extra careful with problems that "hide" information on the edges of a graph.

3.3.2 Modular Width vs Clique-width

In the last section of this chapter, let us visit a less-studied parameter that covers well-structured dense graphs: modular width. The first paper to explicitly consider modular width as a structural parameter was probably [93], whose motivation was the search for some parameter that covers dense graphs (and therefore is not dominated by treewidth) but still manages to solve some problems which are hard parameterized by clique-width. One such parameter was of course neighborhood diversity, which is restricted enough to render almost all problems tractable – indeed FPT algorithms for Edge Dominating Set, Coloring and Hamiltonicity were already given in [126]. However, neighborhood diversity can only really be thought of as a "toy" parameter for most purposes, as the structure it imposes is so strict that very few graphs have low neighborhood diversity.

Modular width was thus seen as an attractive alternative because it can be seen as the natural inductive generalization of neighborhood diversity. Indeed, a graph has neighborhood diversity k if we can partition it into k modules that induce cliques or independent sets. Modular width takes this a step further in a natural way by saying that a graph has modular width k if we can partition it into k modules, each of which has (recursively) the same structure, that is, modular width k. In a sense, one could say that the relation between modular width and neighborhood diversity is like the relationship between tree-depth and vertex integrity. Despite this, in the realm of dense graph parameters, neighborhood diversity is actually more well-studied and understood than modular width.

Main Techniques. Modular width has been considered as a structural parameter to handle various problems and, since it is incomparable to treewidth, this parameter has the advantage that it can be considered to handle even problems which are intractable for treewidth. Furthermore, a significant advantage of modular width is that this parameter can be computed in polynomial time exactly (unlike treewidth and the majority of the other measures we consider). To give the reader the flavor of this parameter, let us briefly sketch some ideas from works involving the author [15, 16, 92].

One of the most characteristic examples of using modular width can be seen in the Coloring algorithm given in [93], which is then generalized to Grundy Coloring in [16]. The idea here is the following: given a partition of G into k modules, we have two cases. Either the k modules are all independent sets or cliques, in which case, the graph actually has neighborhood diversity k and we can invoke the algorithm of [126]; or some of the modules have a more complex structure. In this case, we take such a module, say V_i, and since we know that $G[V_i]$ has modular width at most k, we apply the same algorithm to color $G[V_i]$. The key insight now is that once we know the chromatic number of $G[V_i]$ we can replace all of V_i in the graph by a clique of size equal to the chromatic number of $G[V_i]$ and this does not change the chromatic number of G. Proceeding in this way we can then eliminate all modules which are not cliques and hence always reduce to the neighborhood diversity case. This surprisingly simple idea is powerful enough to handle not only the more general Grundy Coloring problem, but also problems of higher complexity, such as the PSPACE-complete Independent Set Reconfiguration problem, which is even PSPACE-hard for constant treewidth [15].

The previous paragraph summarizes the first natural key idea which is typically used for modular width-based algorithms, namely, the idea that we can isolate modules, handle them independently, and then replace them in the original graph to obtain an easier instance [92]. However, even for problems where this is not possible (for example because coming up with an appropriate graph that would replace a processed module is not easy), modular width can be useful by leveraging a technique that has been used heavily in the context of neighborhood diversity: Integer Linear Programming (ILP) with a bounded number of variables. A key result due to Lenstra, later improved by Kannan [117, 116] states that ILPs with k variables can be solved in time that is FPT in k (and polynomial in the size of the program). This is a trick that is often used to handle problems parameterized by neighborhood diversity – indeed the Coloring algorithm of [126] we invoked in the previous paragraph uses this technique. This technique can also be used in the context of modular width, where we pre-process the modules (solving possibly a more general problem) and then use an ILP to combine the solutions of individual modules into a solution of the larger instance. This is, in particular, the technique used in [93] to solve Hamiltonicity parameterized by modular width.

Is this a good parameter? Based on the above, modular width seems like a promising parameter that captures a type of structure different from treewidth. It is quite rewarding that some problems which are hard for clique-width, or even for treewidth, do become tractable for this parameter – this is quite positive, since some of the same problems tend to resist parameterizations which are quite severe, if we remain in the hierarchy below treewidth. Nevertheless, modular width has not yet become a major player in parameterized complexity research. This is probably at least in part due to the fact that this parameter seems to be much less well-behaved (for example, deleting one vertex from a graph can have an arbitrarily large impact on its modular width). Furthermore, to the best of our knowledge, no problem is currently known to be W[1]-hard parameterized by modular width, without also being W[1]-hard for neighborhood diversity. It would therefore be interesting to further study the properties of this parameter and in particular to try to discover some natural problems which are W[1]-hard for modular width while being tractable for more restricted parameters, as this may shed further light on what this parameter is about.

Chapter 4

Fine-Grained Parameterized Complexity

The general direction behind the results we presented in Chapter 3 was structural: we wanted to understand how measuring the input structure in different ways can affect the complexity of various problems, and conversely which type of problem is amenable to parameterization with different types of structural measures. Even though this approach has been quite successful, one could argue that it contains a mild paradox. Recall that one of the motivations for looking at structural parameters in the first place was to replace the binary world of graph classes (where a graph either belongs in a class or it does not) with a more smoothed setting, where we quantify how close a graph is to having a desirable algorithmic property. Happily, this decision did indeed allow us to discover many interesting new results and to get a clearer picture of various problems' complexity. The mild paradox, however, is that while we adopted this new smoothed approach with respect to *graph* complexity, we continued using a traditional binary approach with respect to *computational* complexity. Just like traditional computational complexity theory classifies problems as polynomial or NP-hard, we continued in the same footsteps, except we were now using the notions of fixed-parameter tractability and W[1]-hardness.

The results we present in this chapter follow a school of thought which can be seen as an attempt to resolve this mild paradox. In this school of thought, the reasoning is that the correct way to express the complexity of a problem is not to classify it as belonging to a class or another, but to actually attempt to measure (as precisely as possible) the resources needed to solve it. In retrospect, this line of reasoning matches perfectly with what we have been doing so far: just like we argue that the correct way to measure graph complexity is quantitative and not qualitative, the correct way to measure computational complexity must be treated in the same way. This research direction, which is not confined to parameterized complexity but has also attracted interest in the single-variable setting, is called fine-grained complexity.

Fine-grained complexity is the branch of computational complexity theory that is interested in providing precise lower bounds on the time needed to resolve a problem in the worst case. Of course, since we don't even know if P=NP, this can only be done under certain complexity assumptions which are typically stronger than (i.e. imply) the conjecture that P$\neq$NP. Probably the most well-known such assumptions are the Exponential Time Hypothesis (ETH) and the Strong Exponential Time Hypothesis (SETH) (recall Definitions 1 and 2). These two hypotheses have been shown to be of much value to parameterized complexity theory, as they have allowed us to explain why algorithmic progress in certain problems has become "stuck" at certain bounds [139]. The two main ways in which such hypotheses are relevant in this area are the XP-optimality and FPT-optimality research programs which respectively attempt to give precise lower bounds for problems in the class XP and in the class FPT. In this work we will mainly focus on results of the FPT-optimality type.

FPT-Optimality: When we can't take Yes for an answer. What do we mean by the term FPT-optimality? Recall that up to this point, when we could obtain an algorithm for a parameterized problem running in $f(k)n^{O(1)}$ we were satisfied, because such an algorithm proves that the problem is fixed-parameter tractable. A reasonable critic of this approach would be quick to point out that this definition of tractability is too relaxed because we are willing to accept *any* function f. Clearly, a problem solvable in $2^k n$ has much different complexity from one solvable in $2^{2^k} n$ and our definition of the class FPT blinds us to this distinction. The goal of the FPT-optimality program is to dig into exactly this kind of distinction and, in particular, whenever a problem appears stuck in some complexity $f(k)n^{O(1)}$ because algorithmic progress has stalled, to give us the tools to place lower bounds on the best function $f(k)$ that can be achieved. Ideally, this will eventually lead to situations where the best known algorithm has complexity that matches the best known lower bound, and we can then consider the problem settled (even if we may not be completely happy with the $f(k)$ we end up with).

Our goal in this chapter is to present some recent works by the author in the general theme of FPT-optimality for graph problems parameterized using the structural measures we have discussed (Figure 2.1). These works can be divided into two parts: fine-grained lower bounds using the ETH; and super fine-grained lower bounds using the SETH.

In the first case, we will be dealing with parameterized problems for which there exist algorithms with dependence, say k^k, or 2^{k^2}, or 2^{2^k}. Our goal will be to establish that this *type* of parameter dependence is best possible, that is, that it is impossible to solve the problem with dependence $k^{o(k)}$, or $2^{o(k^2)}$, or $2^{2^{o(k)}}$. In other words, we will be proving that, under the ETH, the known parameter dependence is essentially optimal, modulo an unknown constant (appearing in the exponent). It is natural to obtain this degree of granularity in lower bounds based on the ETH, because the ETH itself is a hypothesis which leaves room for an unknown constant (it states that 3-SAT cannot be solved in c^n, for some unspecified constant c). This type of result is the subject of Section 4.1.

The type of result mentioned in the previous paragraph is a nice way to reassure ourselves that we have an almost-optimal algorithm when the complexity we achieve is super-exponential in k. However, when a problem is solvable in $2^{O(k)}n^{O(1)}$, ETH-based lower bounds can typically only tell us that this is not improvable to $2^{o(k)}n^{O(1)}$. One could argue that, in this context, an algorithm running in 2^k is not at all similar to an algorithm running in 9^k. It therefore makes sense to go even further in our quest for granularity and attempt to distinguish between the two and for this we need to assume a stronger complexity hypothesis. This brings us to the second type of result we present in this chapther, where we show, assuming the SETH, that various problems which are solvable in $c^k n^{O(1)}$ time, cannot be solved in $(c-\epsilon)^k n^{O(1)}$ time. This type of super fine-grained result, which attempts to pinpoint exactly the parameterized complexity of a problem, is the subject of Section 4.2.

4.1 Fine-Grained Lower Bounds Using the ETH

In this section we deal with several problems which are FPT parameterized with one of our structural parameters. Our objective will be to decide what is the best function f such that our problems admit an $f(k)n^{O(1)}$-time algorithm, and for this we will rely on reductions from 3-SAT and the ETH. We will make an effort to highlight one interesting aspect of this line of research, namely, the diversity of the different functions $f(k)$ which may arise in practice for natural problems. Indeed, one striking fact of FPT optimality is that, even for exotic-seeming functions $f(k)$, sooner or later some problem will turn up whose correct complexity will end up being $f(k)$.

A main highlight in this chapter is a result stating that $\exists\forall$-SAT has double-exponential complexity parameterized by vertex cover (and hence by treewidth), given in Section 4.1.1. This is of particular interest because it tightly encapsulates a wider phenomenon (that adding quantifiers to a problem increases its complexity by treewidth by a level of exponentiation) in a reduction

that is short and simple enough to be taught in a basic parameterized complexity course[1]. As we explain, this clean and elegant result is the result of a distillation of the insights of several previous works. In addition to its value for treewidth, this adds another natural problem to the list of FPT problems for which the parameter dependence is double-exponential [1, 52].

In the following two sections we deal with problems whose complexity is more manageable, but still higher than exponential. In Section 4.1.2 we revisit (as promised) the complexity of d-Scattered Set and (k,r)-Center parameterized by tree-depth and show that the 2^{td^2} dependence cannot be improved to $2^{o(\mathrm{td}^2)}$. This is interesting both because it shows that the trivial argument we used to obtain fixed-parameter tractability in this case is optimal, and because problems whose complexity is exponential in the square of the parameter are not so common (see [157] for another natural problem whose complexity is exponential in the square of a structural parameter). We continue in Section 4.1.3 where we show that Digraph Coloring has slightly super-exponential complexity of the form $\mathrm{tw}^{\mathrm{tw}}$, which cannot be improved to $\mathrm{tw}^{o(\mathrm{tw})}$, or even $\mathrm{td}^{o(\mathrm{td})}$, intuitively because this coloring problem forces us to remember an ordering of the vertices of each bag.

We then change gears slightly and in Section 4.1.4 consider the meta-problem of deciding if a given graph G satisfies a given MSO formula ϕ. Courcelle's theorem states that this can be solved in $f(\phi,\mathrm{tw})n$, but the function f is in the worst case a tower of exponentials and this is optimal. In the case of vertex cover it is known that deciding any fixed MSO formula ϕ can be done in $2^{2^{\mathrm{vc}}}n^{O(1)}$, so things are much better than for treewidth, and this is optimal. We show that for vertex integrity things are slightly worse but still much better than for treewidth: any fixed MSO formula ϕ can be decided in $2^{2^{\mathrm{vi}^2}}n^{O(1)}$ time, and this extra quadratic dependence on vi is inevitable assuming the ETH. These results indicate that vertex integrity is a measure of truly intermediate complexity between vertex cover and tree-depth.

Finally, in Section 4.1.5 we look at a topic that can be considered a wide-open frontier in this area: dealing with multiple parameters. We revisit the Min Stable Cut problem, which we evoked in Section 3.2.3. This time we allow the input to be edge-weighted (which makes the problem much harder) but parameterize by $\mathrm{tw}+\Delta$. This leads to an FPT algorithm of dependence $2^{O(\mathrm{tw}\cdot\Delta)}$ and we consider the question of deciding whether this algorithm is best possible. As we explain, this is a complicated question, as the fine-grained complexity toolbox has not yet fully evolved to the point where we can easily handle problems with more than one parameter. Nevertheless, we present a family of reductions that does allow us to claim that our algorithm is essentially optimal.

4.1.1 Double-Exponential: SAT with Two Quantifiers

The work we present in this section is taken from [135] (see also [134] for some follow-up). The main take-away of our results is that $\exists\forall$-SAT has double-exponential complexity when parameterized by the vertex cover (and hence the treewidth) of the incidence graph of the formula. Even though in isolation this may sound like a quaint fact, this result can be seen as the culmination of a long line of research investigating the interaction between quantifier alternations and treewidth. As a nice bonus, the lower bound we present is short and self-contained. Hence, the results of this section can be seen as the "textbook" version of a line of work whose main message is that as we consider problems with more quantifier alternations, the complexity as a function of treewidth explodes super-exponentially.

Let us proceed to a more detailed presentation. We consider the standard Σ_2^p-complete problem $\exists\forall$-SAT: we are given a DNF formula ϕ with the variables partitioned into two sets X, Y, where X is the set of existential variables and Y the set of universal variables. The question is whether there exists an assignment to X so that for all assignments to Y at least one term of ϕ is set to True. The question we would like to answer is what is the complexity of solving this problem as a function of the treewidth of the primal graph of ϕ. The primal graph contains a vertex for

[1]Indeed, it would be fair to say that if there is a result in this work that deserves to be part of a future parameterized algorithms textbook, it is this one.

each variable and an edge between any two variables that appear in a common term. Note that parameterizing by the treewidth of the primal graph is more restrictive than parameterizing by the treewidth of the incidence graph, so our lower bound will also apply to the incidence graph.

The complexity of the best known algorithm for this problem is of the form $2^{2^{O(\mathrm{tw})}}n^{O(1)}$, due to Chen [40]. It is therefore a natural question to ask where this double-exponential dependence comes from and whether it can be avoided. Regarding the first question, it had previously been observed that the algorithm used in Courcelle's theorem [46] has a dependence on treewidth that is a tower of exponentials whose height grows with the number of *quantifier alternations* in the input formula. Since our problem has two quantifiers, it sounds logical that the algorithm of [40] would get a double-exponential dependence. Regarding the second question, it was later shown that the behavior of Courcelle's theorem is essentially optimal: the complexity must be a tower of exponentials whose height increases with the number of quantifier alternations, even if the input is a tree, as shown by Frick and Grohe [90]. As a result, the general feeling was that extra quantifier alternations "should" increase the number of exponentiations in the complexity dependence on treewidth; however, Frick and Grohe's work only showed this asymptotically and for an artificial MSO-expressible problem. Addressing this, Pan and Vardi [155] showed that, for every *odd* k, quantified SAT with k quantifiers had a dependence on treewidth that was at least k-fold exponential (and this is tight by the algorithm of [40]).

The general message of these works can therefore be seen as implying that each extra quantifier in the definition of a problem should increase its complexity parameterized by treewidth exponentially, and this is captured exactly by quantified SAT. Nevertheless, because the work of Pan and Vardi only covered odd numbers of quantifiers, this left the complexity of problems in the second level of the polynomial hierarchy wide open. This was explicitly addressed by Marx and Mitsou [143] who were the first to identify a natural Σ_2^p-complete graph problem, List Choosability, which they showed, through an intricate reduction, to have complexity that depends double-exponentially on treewidth.

The work we present here can therefore be seen as the culmination of this line of research. Instead of defining an arbitrary problem (as in [90]) or digging for an obscure graph problem, we study the prototypical version of SAT that contains two quantifiers and we present a reduction that is much simpler than that of [143] and implies that under the ETH the problem's complexity dependence on treewidth (and even on vertex cover) is double-exponential. Hence, the importance of this result is that we now have an example that clearly explains how and why extra quantifiers add more to the complexity of graph problems parameterized by treewidth. In other words, the result of this section can be seen as the textbook version of a series of results, all of which can be informally interpreted as saying that an extra quantifier costs an extra level of exponentiation in the complexity with respect to treewidth.

We note that this work was later followed up on by Fichte et al. [81] who gave a unified proof that filled in the remaining holes, that is, they showed that for quantified SAT with k quantifiers where k is even and at least 4, the complexity on treewidth in k-fold exponential. Hence, we now know that for all k, the version of SAT that is complete for the k-th level of the polynomial hierarchy has k-fold exponential complexity dependence on treewidth. However, in this section we will concentrate on the case of 2 quantifiers, which is probably the most interesting level of the polynomial hierarchy outside NP.

The reduction. Without further ado, let us sketch the reduction of [135]. We begin with a 3-SAT formula ϕ with n variables and m clauses, where we assume without loss of generality that m is a power of 2. Recall that, under the ETH, it should be impossible to decide ϕ in time $2^{o(n+m)}$. Suppose that the variables of ϕ are $X = \{x_1, \ldots, x_n\}$ and the clauses are numbered $c_0, \ldots, c_{m-1}$. We construct an instance ψ of $\exists\forall$-SAT as follows: the set of variables is $X \cup Y$, where Y is a set of $\log m$ fresh variables $y_0, \ldots, y_{\log m-1}$. For every clause c_i of ϕ we construct $|c_i|$ terms, one for each literal of c_i. More precisely, each such term contains a distinct literal of c_i and all of the variables of Y. We now set things up so that any two terms of ψ which were produced from the same clause c_i agree on the sign of all variables of Y; while any two terms which were produced from distinct

clauses disagree on the sign of at least one variable of Y. This can be achieved by using the signs of the variables of Y in each term to encode in binary the index of c_i; for example we could say that the terms that we produce for c_i have negations on the variables y_j such that the j-th bit of the binary representation of i is 1, while the remaining variables of Y appear positive in such terms.

This completes the construction. We have now constructed an instance ψ of $\exists\forall$-SAT with $n + \log m$ variables and at most $3m$ terms. If there is a satisfying assignment for ϕ, we use the same assignment for X and claim that for any assignment to Y at least one term of ψ is satisfied. Indeed, because m is a power of 2, for each assignment of Y there is an integer i such that if we consider the terms produced from clause c_i, all the literals from Y contained in those terms are set to True. Because we started with a satisfying assignment to ϕ, at least one of the terms will therefore evaluate to True. The converse direction is similar. Hence, deciding ψ is equivalent to deciding ϕ.

The crucial property of our construction is now that the vertex cover of the primal graph of ψ is only $\log m$, because the set Y is a vertex cover (no two variables of X appear in the same term). Hence, if there was an algorithm deciding $\exists\forall$-SAT in time $2^{o(2^{\mathrm{vc}})}|\psi|^{O(1)}$, this would give a $2^{o(m)}$ algorithm for 3-SAT, contradicting the ETH. Hence, the double-exponential dependence on vertex cover (and a fortiori treewidth) is optimal!

What have we learned? A key insight of the reduction above, which is due to [143], is that 3-SAT can be seen as a problem with an $\exists\forall$ structure: we ask whether there exists an assignment such that all clauses are satisfied. Of course, the $\forall$ part is polynomially-checkable (hence the problem is only NP-complete), but this does not preclude us from leveraging it to show a lower bound parameterized by treewidth! Indeed, the fact that the $\forall$ part of 3-SAT can be encoded using $\log m$ variables is key here and matches what we want to do exactly, as we are trying to encode the whole instance in a structure whose treewidth is logarithmic (so that we get a double-exponential lower bound on the running time). In a sense, this lower bound argument is rather devastating because it shows that in order for the double-exponential dependence on treewidth to manifest itself, we do not need a Σ_2^p-hard problem: the class of instances of $\exists\forall$-SAT we construct is clearly in NP. The lesson we learn is, therefore, that the difficulties encountered by Courcelle's theorem in handling quantifier alternations go very deep: even innocuous-seeming quantifiers may end up costing dearly in the parameter dependence on treewidth. Thankfully, this behavior does not seem to be too widespread, as most problems which are in NP turn out to be much more tractable for treewidth, even if they can be seen to contain some form of $\exists\forall$-structure (e.g. Dominating Set).

Going a bit further in our analysis, it is interesting to note that the reduction we sketched above can give various other nice implications. First, the hardness bound also applies to the incidence graph, which also has vertex cover (and treewidth) $\log m$. Second, looking at the dense parameters of our hierarchy, the reduction above has constant incidence-graph clique-width, meaning that it establishes that $\exists\forall$-SAT is a problem that goes from being FPT parameterized by treewidth to paraNP-hard parameterized by clique-width. This is another indication that incidence graph clique-width may not be a great measure of structural complexity, complementing the results of Section 3.3.1.

Finally, one may wonder what would be the effect of adding more quantifiers. As we mentioned, in the case of treewidth the answer is that each quantifier costs a level of exponentiation. However, since the reduction already applies to vertex cover it makes sense to consider this parameter. Quite nicely, it can be shown that the double-exponential dependence on vertex cover is *independent* of the number of quantifiers, that is, adding more quantifiers does not worsen the complexity with respect to vertex cover [135]. This nicely mirrors the complexity of MSO logic with respect to this parameter, which is also double-exponential independent of the number of quantifiers [126].

4.1.2 Exponential in the Square: Scattered Set Revisited

In this section we revisit the unweighted versions of d-Scattered Set and (k, r)-Center parameterized by tree-depth, presenting results from [119, 120]. Recall that we mentioned in Section

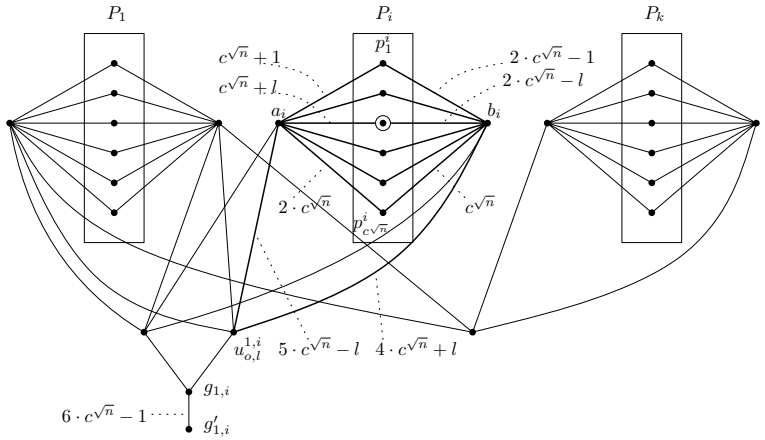


Figure 4.1: Sketch of the ETH-based lower bound for d-Scattered Set parameterized by tree-depth.

3.2.2 that these problems, which are W[1]-hard parameterized by pathwidth and feedback vertex set, are FPT parameterized by tree-depth with complexity of the form $2^{O(\mathrm{td}^2)}$. Our main goal in this section is to explain that this (curious) exponential dependence on the square of the parameter is best possible.

Before we go on to sketch the ideas of the lower bound argument, let us take another look at why we have an algorithm with this complexity. Recall that d-Scattered Set admits a $(d+1)^{\mathrm{tw}}n^{O(1)}$ algorithm and similarly (k,r)-Center admits a $(2r+1)^{\mathrm{tw}}n^{O(1)}$ algorithm. Our argument in Section 3.2.2 was that, because the diameter of a graph is always upper-bounded by 2^{td} and it never makes sense to consider the two problems for distances higher than the diameter, we automatically obtain algorithms that run in $2^{O(\mathrm{td}^2)}n^{O(1)}$ for both problems.

The reduction. Let us sketch the main idea behind the two reductions. The general strategy is very similar to the reduction sketched in Section 3.2.2; indeed, the reader is invited to compare Figure 4.1 to Figure 3.3. The key difference is in the setting of the parameters. Although the reductions in [119, 120] are formulated from 3-SAT, we can equivalently think that we are reducing from an instance of k-Multi-Colored Clique where the $k=\sqrt{n}$ and each part of the graph has size $N=2^{O(\sqrt{n})}$ (going from 3-SAT to this version of k-Clique can be done with standard techniques). This is a problem that cannot be solved in $N^{o(k)}=2^{o(n)}$ under the ETH. Given this, we can just execute the same reduction that we did in Section 3.2.2. This produces a graph where, if we remove $k=O(\sqrt{n})$ vertices, what is left is a collection of paths of length $2^{O(\sqrt{n})}$. Since a path of length 2^ℓ has tree-depth at most ℓ (this can be seen by induction by removing the middle vertex of the path), the whole graph has tree-depth $O(\sqrt{n})$. Hence, if there was an algorithm that could solve d-Scattered Set in the new graph G in time $2^{o(\mathrm{td}^2)}|G|^{O(1)}$, we would violate the ETH. In other words, this reduction, if we start from a 3-SAT instance, runs in sub-exponential time and constructs a graph of sub-exponential ($2^{O(\sqrt{n})}$ size), whose tree-depth is proportional to $\sqrt{n}$. The existence of the aforementioned algorithm shows that this seemingly random confluence of parameters is actually optimal: we need the graph to have very long paths (otherwise d-Scattered Set becomes easier), but we cannot have paths longer than $2^{\sqrt{n}}$, since we want the tree-depth of the graph to end up being at most $O(\sqrt{n})$ (to obtain the tight lower bound on the running time).

What have we learned? The reduction of this section is an excellent example of why fine-grained complexity is an indispensable tool in a research program which tries to measure the algorithmic impact of graph structure. Recall that, according to the results we presented in Section 3.2.2, the two problems we consider here are hard for pathwidth but easy for tree-depth. However,

when we dig a bit deeper, we realize that actually we do not have a better algorithm when the problem is parameterized by tree-depth! The only thing we have, that leads to fixed-parameter tractability, is a combinatorial bound relating tree-depth to a second problem parameter. It therefore seems paradoxical that our theory should classify this version of the problem as tractable for this parameter, even though no algorithmic argument specific to tree-depth appears anywhere at all! Thankfully, the fine-grained complexity argument of this section can be seen as a resolution of this paradox. One interpretation of this result is that, even though the bound given by tree-depth on the diameter allows us to bound the space of possibilies for these problems, even in this restricted space of possibilities the best algorithm possible is still the simple dynamic program that considers all possible distances for all vertices in a bag. In other words, the fine-grained lower bound result allows us to see that the problem is essentially still hard parameterized by tree-depth.

Given these observations, it would be very interesting to revisit various problems which we have identified in our price-of-generality search in Chapter 3 and see for which of these their tractability in more restricted parameters is genuine (i.e. there exists an algorithm tailored to that parameter which does not work in the more general case) and for which it is "fake" (i.e. fixed-parameter tractability only follows from a secondary combinatorial bound and an execution of a more general algorithm, and this can be shown to be optimal under the ETH). A prime suspect here is, as we have mentioned, Grundy Coloring, where we noticed that fixed-parameter tractability by pathwidth is only a side-effect of a known bound on the optimal with respect to this parameter.

4.1.3 Slightly Super-Exponential: Digraph Coloring

We now consider a problem whose complexity turns out to be slightly super-exponential, presenting results that recently appeared in [107]. We use the term slightly super-exponential to denote a parameter dependence of the form $k^{O(k)}$ (as opposed to $2^{O(k)}$). The first work to focus on such problems (and to use this term) was the work of Lokshtanov et al. [141] which showed, for example, that Chromatic Number and Disjoint Paths parameterized by treewidth cannot be solved in $\mathrm{tw}^{o(\mathrm{tw})}n^{O(1)}$ (assuming the ETH). This has recently been followed by results showing that several other natural problems have slightly super-exponential complexity parameterized by treewidth [21, 34, 35].

Where does the super-exponential parameter dependence on treewidth come from for such problems? Intuitively, the $\mathrm{tw}^{\mathrm{tw}}$ running time usually indicates that we are either trying to keep track of connectivity information or that we are trying to order the vertices of each bag. Indeed, the problem we consider in this section (Digraph Coloring) can be recast in this way, and this is where the super-exponential running time of our algorithm comes from. Nevertheless, a surprising development of the previous decade was that for many problems the natural dynamic program which keeps track of such information (and therefore has super-exponential size) is *not* optimal! This was exemplified by the Cut&Count technique, which managed to obtain single-exponential parameterized algorithms for problems such as Hamiltonicity and Steiner Tree parameterized by treewidth, for which the best known algorithm had long been super-exponential, and this had been conjectured to be optimal [51]. In this light, the fact that the "easy" algorithm for Digraph Coloring did turn out to be optimal was not a given.

The problem. In Digraph Coloring we are given as input a directed graph $G = (V, A)$ and an integer k and are asked to partition $V(G)$ into at most k sets so that each set induces an *acyclic* digraph. Notice that if we start with a graph G that is actually undirected and replace each edge by a pair of anti-parallel arcs, this problem is exactly k-Coloring (hence the name Digraph Coloring). Nevertheless, this is a much harder problem, as it becomes already NP-hard for $k = 2$.

Before we continue, let us ponder how the structural parameter framework should be applied to this problem for which the input is not a graph but a digraph. Though we have not much talked about them, directed graph analogues of our structural parameters do exist, notably directed treewidth and its many variations [22, 109, 115]. Typically, these parameters are built around the notion that the "trivial case" is a DAG, and they attempt to measure the distance of a given

digraph from being acyclic. Unfortunately, these width parameters have been much less successful than treewidth, for the simple reason that the vast majority of digraph problems, even when they are tractable on DAGs, end up being intractable by most notions of directed treewidth [94, 95, 132]. The same is true of Digraph Coloring: even though the problem is clearly polynomial-time solvable on DAGs, for every fixed $k \geq 2$ it is NP-hard to color a digraph with k colors, even if we are given in the input a directed feedback vertex set[2] of size k [107][3]. Because all known directed variants of treewidth are more general than directed feedback vertex set, this shows that obtaining fixed-parameter tractability for these parameters is, as usual, hopeless.

Because of these reasons we will continue using our undirected structural parameters, such as treewidth, to deal with this problem, where this time we consider the treewidth of the underlying undirected graph of the input. We note, however, that this is not fully satisfactory, as it does indeed fail to take into account that many dense graphs may be trivial for this problem (for example DAGs).

Algorithmic Upper Bound. Let us now briefly give a high-level intuition explaining why the complexity of Digraph Coloring is at most $tw^{O(\mathrm{tw})}n^{O(1)}$. The algorithm performs standard dynamic programming on a tree decomposition, so the question is what type of information we need to store for each bag. First, we note that we need to remember at least the color of each vertex, giving a factor of k^{tw}. We then note that, without loss of generality $k \leq \mathrm{tw}+1$, as otherwise the answer is trivially Yes and a k-coloring of the underlying undirected graph (hence a coloring of the digraph) can be found in polynomial time. Hence, $k^{\mathrm{tw}} < \mathrm{tw}^{\mathrm{tw}}$ and this is one reason we need a super-exponential complexity. However, we are not done as we also need to make sure that our solution does not construct monochromatic directed cycles. The key idea now is that in a partial solution the vertices which share a color form part of a DAG and can therefore be topologically sorted in a way that is compatible with all the arcs of the DAG. What we therefore need to remember is the topological ordering of the vertices that share a color, giving a second reason why the size of the table is of the order $\mathrm{tw}! = \mathrm{tw}^{O(\mathrm{tw})}$.

Lower Bound. Given that the problem is FPT by (undirected) treewidth, it now becomes a natural question whether a $2^{O(\mathrm{tw})}n^{O(1)}$ algorithm is possible. An immediate answer is no, because the problem, as we said, generalizes Chromatic Number, which by [141] does not admit a $\mathrm{tw}^{o(\mathrm{tw})}n^{O(1)}$ algorithm. However, this explanation is not entirely satisfying, as the lower bound of [141] relies on instances with a large number of colors; indeed, undirected 3-Coloring can of course be solved in $3^{\mathrm{tw}}n^{O(1)}$. So the more interesting question is whether the ordering part of our algorithm is optimal. To answer this question, we therefore have to focus on the case where the number of colors is small, hence the lower bound of [141] does not apply.

Without going into the technical details, this is exactly the idea of the lower bound proof given in [107], which establishes that Digraph Coloring cannot be solved in $\mathrm{td}^{o(\mathrm{td})}n^{O(1)}$, even for $k = 2$ colors. The high-level strategy of the reduction is the following: we start from a 3-SAT formula ϕ with n variables and partition the variables into $\log n$ groups of size $n/\log n$, so enumerating all assignments to a group can be done in subexponential time. The key idea now is that each group will be represented by a group of "only" $n/\log^2 n$ vertices, which are all meant to receive the same color. Indeed, the $(n/\log^2 n)!$ different orderings of these vertices will be sufficient to encode the $2^{n/\log n}$ assignments to the variables of the group. In this way we have "saved" a factor of $\log n$: we represent the assignment to $n/\log n$ variables using $n/\log^2 n$ vertices. This saved logarithmic factor in the size of the part of the graph that represents the assignments allows us to obtain a graph with tree-depth $O(n/\log n)$, proving that if one could 2-color the resulting instance with parameter dependence $\mathrm{td}^{o(\mathrm{td})}$ we would violate the ETH. Digraph Coloring is therefore a nice problem that exemplifies the slightly super-exponential complexity that sometimes arises when we have to consider all orderings of the vertices of each bag.

[2] A directed feedback vertex set is a set of vertices whose removal destroys all cycles.

[3] Notice that this is very tight, as if we are given a feedback vertex set of size $k-1$ the answer is trivially Yes.

4.1.4 Tight Meta-Theorems for Vertex Integrity

In this section we present some very recent results from [136] regarding algorithmic meta-theorems for vertex integrity. Algorithmic meta-theorems are general statements proving the tractability of a whole class of problems. In this case, we will be dealing with the (well-studied) situation where the meta-theorem involves a logic language called Monadic Second Order logic: our goal will be to prove that every problem expressible in this logic (which allows us to quantify over sets of vertices) is tractable parameterized by vertex integrity.

An astute reader will observe that this fact has long been known. Indeed, Courcelle's celebrated theorem states that all problems expressible in MSO logic are FPT by treewidth (and hence by vertex integrity) [46]. The problem is that a precise reading of Courcelle's theorem reveals that the complexity of deciding a problem described by an MSO formula ϕ is $f(\mathrm{tw}, \phi)n$, where f is a tower of exponentials whose height depends on the number of quantifier alternations of ϕ. Even worse, this cannot be avoided even for trees (unless P=NP) [90]. Things do get slightly better if we parameterize by tree-depth, as then the height of the tower of exponentials can also be upper-bounded by td [91], but the tower of exponentials is unavoidable in this case as well [128]. On the other hand, if we go to the other extreme and parameterize by vertex cover, the complexity of MSO logic is "only" double-exponential, and this is best possible under the ETH [126].

This leads us to the possibility that vertex integrity, which as a parameter is "between" vertex cover and tree-depth could have a complexity for MSO model-checking which is also between double-exponential and a tower of exponentials. This is exactly the message of the results of [136]: any MSO property which can be described by a formula ϕ can be decided in time $2^{2^{O(\mathrm{vi}^2+|\phi|\mathrm{vi})}}$, that is, double-exponential in the square of vi.

Intuitive explanation. Where does the double-exponential complexity in the square of vi come from? To explain this, one needs to recall where the double-exponential MSO model-checking complexity for vertex cover, given in [126] comes from. The idea is that vertices of a graph can be partitioned into 2^{vc} equivalence classes, in the sense that equivalent vertices are twins (they have the same neighbors). We then show that if such a class has more than 2^q elements, then we can safely delete one without affecting the truth of any MSO formula with at most q quantifiers. Hence, the strategy is one of kernelization. In the end we can simply execute the standard brute-force algorithm on the kernel (which has size $2^{\mathrm{vc}+q}$) to decide the formula.

The difference now is that the number of equivalence classes in the context of vertex integrity is not 2^{vi} but more like 2^{vi^2}. In particular, what is happening now is that we have a separator S of size at most vi, such that removing S from the graph gives a collection of components of size at most vi. These components can be distinguished from each other either because they have different internal structure (but there are only $2^{O(\mathrm{vi}^2)}$ different graphs of order vi) or because they have different neighbors in S (but again there are only at most 2^{vi^2} possibilities). Hence, in this case the strategy is to partition the components into $2^{O(\mathrm{vi}^2)}$ equivalence classes and then, if one class has too many components, to delete one. The kernelization step is once again followed by an execution of the brute-force algorithm.

Perhaps the most interesting part of this story is that the argument we sketched above is optimal: it is shown in [136] that there exists a constant-size MSO formula which cannot be decided in $2^{2^{o(\mathrm{vi}^2)}}$ under the ETH. The key intuition here is that we can take the arguments counting the number of different component types of the previous paragraph and use them in a reduction: we construct a graph with a separator S of size $\sqrt{logn}$ such that every remaining component if we delete S has size $O(\sqrt{logn})$ and all components are distinguishable by their neighborhood in S. In this way, we can construct n distinguishable components, which is sufficient to encode, for example, the structure of a graph on n vertices. We are then ready to translate any standard problem on this original graph (e.g. 3-Coloring) into an MSO property of the new graph. Since the new graph has vertex integrity $O(\sqrt{\log n})$, if we could decide any MSO property in $2^{2^{o(\mathrm{vi}^2)}}n^{O(1)}$, we would obtain a sub-exponential algorithm for 3-Coloring, violating the ETH.

What have we learned? The message of this meta-algorithmic result is a confirmation of the intuition that vertex integrity is a parameter that is truly intermediate between the extreme restriction of vertex cover and the much more general case of tree-depth. Nevertheless, it must be noted that the lower bound result only applies to an artificial problem. It would be interesting to see if more natural problems exhibit the same "double-exponential in the square" behavior for this parameter.

4.1.5 More Variables: Minimum Stable Cut

We now revisit the Min Stable Cut problem, which we already mentioned in Section 3.2.3, presenting some further results from [131]. We will stay in the general theme of ETH-based lower bounds, but the twist in this problem is that we will now be interested in proving a lower bound that involves two distinct parameters. As we explain, this significantly complicates things.

Let us recall the problem. We are given a graph $G = (V, E)$, which can now be edge-weighted, and are asked to partition V into two sets so that (i) for each vertex, at least half of its incident edge weight is cut (ii) the total weight of cut edges is minimized. In Section 3.2.3 we considered the unweighted restriction of this problem, which can easily be solved in $\Delta^{O(\mathrm{tw})}n^{O(1)}$. It is not hard to see, though, that it is much harder to apply the same techniques to the weighted version. The straightforward way to adapt the DP algorithm would remember for each vertex in a bag the total *weight* of edges currently connecting this vertex to the other side, but this would lead to a complexity of the form $(\Delta W)^{O(\mathrm{tw})}n^{O(1)}$, where W is the largest weight in the graph. This complexity is not even polynomial when $\mathrm{tw} = O(1)$, as W could be written in binary and hence be exponential in n. Indeed, it is shown in [131] that this is inevitable, as the weighted problem is weakly NP-hard on trees, hence the best we could hope for is a pseudo-polynomial time algorithm when $\mathrm{tw} = O(1)$.

The weak NP-hardness of the problem on trees motivates us to try a different approach. We observe that one way to avoid W appearing polynomially in the running time is to store for each vertex the partition of its neighborhood between the two sides. This naturally leads to an algorithm running in $2^{O(\Delta \mathrm{tw})}(n + \log W)^{O(1)}$ time, where the important factor comes from the fact that for each vertex in a bag we have 2^{Δ} possible partitions of its neighborhood. The goal of this section is to argue that this algorithm is best possible. Observe that the crucial difference with previous sections is that we now have two involved parameters: tw and Δ.

The easy way to a lower bound and why it doesn't work. What we are interested in is proving that the exponential dependence on *the product* $\Delta \cdot \mathrm{tw}$ which appears in the exponent is optimal, that is, that the problem cannot be solved in, for example $2^{\Delta+\mathrm{tw}}n^{O(1)}$. The fact that both parameters should appear in the exponent at least linearly follows from linear NP-hardness reductions for this problem (the problem is NP-hard for constant tw and also for constant Δ), but such reductions cannot rule out the $2^{\Delta+\mathrm{tw}}$ dependence speculated above. One thing we could therefore try is the following: start with a 3-SAT instance ϕ with n variables and produce an instance G of Min Stable Cut that encodes ϕ (that is, the min stable cut value of G tells us if ϕ is satisfiable), such that G satisfies the following: (i) $\Delta(G) = O(\sqrt{n})$ (ii) $\mathrm{tw}(G) = O(\sqrt{n})$ (iii) the size of G and the largest weight are both $2^{o(n)}$. If we come up with a reduction that satisfies these parameters, this proves (under the ETH) that Min Stable Cut cannot be solved in $2^{o(\Delta \mathrm{tw})}(|G| + W)^{O(1)}$. Therefore, our algorithm that has parameter dependence $2^{O(\Delta \mathrm{tw})}$ is optimal. Therefore, we are done! Or, are we?

It is true that a hypothesized reduction satisfying the parameters of the previous paragraph would indeed imply that the problem cannot be solved in $2^{o(\Delta \mathrm{tw})}$. The problem here is that this does not quite mean that our algorithm is optimal, or more precisely, it only proves that our algorithm is optimal in one particular case. To see this, we need to realize that when we are dealing with two parameters, the space of all possible running times is only *partially* ordered. When dealing with one parameter, it is clear that, say k^k is worse that 2^k, and 2^{k^2} is worse than k^k, and so on. In principle, any reasonable running time function f can be (asymptotically)

compared with any other reasonable running time function g, and we can decide whether $f = o(g)$, or $g = o(f)$, or $f = \Theta(g)$. This is not the case when several vairables are in play.

To make the above argument concrete, a reasonable question is the following: can Min Stable Cut be solved in time $2^{\Delta^2+\mathrm{tw}}n^{O(1)}$? Such an algorithm would not run afoul of any of the aforementioned lower bounds, including the hypothesized reduction that sets $\Delta = O(\sqrt{n})$. It would, however, be much faster than our algorithm whenever $\Delta < \mathrm{tw}$, which is a condition satisfied by numerous graphs. If such an algorithm exists, it would be disingenuous on our part to claim that the $2^{\Delta\mathrm{tw}}$ dependence we have achieved is best possible: our algorithm would only be best possible when $\Delta > \mathrm{tw}$. Of course, the same discussion could have taken place for several other possible running times ($2^{\Delta+\mathrm{tw}^2}$? $2^{\sqrt{\Delta}\cdot\mathrm{tw}^{3/2}+\Delta}$?). The root of our conundrum is that we are dealing with functions that cannot be totally ordered: we cannot decide whether $\Delta^2 + \mathrm{tw}$ is less than $\Delta \cdot \mathrm{tw}$ without knowing anything about the relation between tw and Δ. This is why obtaining a lower bound that is as conclusive as the lower bounds we managed to obtain for single-parameter problems is a challenging task.

Let's do it anyway! Despite these difficulties, it is in fact possible to convince ourselves that the $2^{O(\Delta\mathrm{tw})}n^{O(1)}$ algorithm is indeed optimal, without the need for any tools significantly more sophisticated than the reductions we have been using so far. Our strategy will be to use a *parametric* reduction[4] which can work for essentially any relation between Δ and pw. In particular, [131] presents a reduction from 3-Set Splitting (which, for instances of n elements cannot be solved in $2^{o(n)}$) with the following properties:

1. The reduction takes as input any desired value Δ and outputs a graph with this maximum degree.
2. The pathwidth of the produced graph in $O(n/\Delta)$.
3. The maximum weight used is $2^{O(\Delta)}$.
4. The reduction can be executed in polynomial time.

We will not describe the technical details of the reduction here, but the elements above are sufficient to make our point. Observe that a parametric reduction that satisfies the properties above would include as a special case the hypothesized reduction we previously evoked: we could simply feed the reduction the value $\Delta = \sqrt{n}$. However, what we get with this parametric approach is much stronger: because we can execute this reduction for whatever value of Δ we desire, we can use it to rule out essentially any other running time, except running times which are at least exponential in the product of $\Delta \cdot \mathrm{tw}$. For example:

- Setting $\Delta = n^{1/4}$ gives a graph with $\mathrm{tw} \leq \mathrm{pw} = O(n^{3/4})$. This rules out an algorithm of the form $2^{\Delta^2+\mathrm{tw}}$.
- Similarly, setting $\Delta = n^{0.9}$ results in $\mathrm{tw} = O(n^{0.1})$ in the produced instance. This rules out the possibility that an algorithm could solve the problem with dependence $2^{\sqrt{\Delta}\cdot\mathrm{tw}^{3/2}+\Delta}$.
- More ambitiously, could an algorithm run in, say $2^{\mathrm{tw}^{0.99}\cdot 2^{2^{\Delta}}+\mathrm{tw}}$? It suffices to execute our reduction with $\Delta = \log\log(n^{0.001}) = \log\log n - O(1)$, which results in $\mathrm{tw} = O(n/\log\log n)$. Under these parameters, the supposed algorithm would violate the ETH.

The point we are trying to make is that, no matter how one tries to rebalance the performance of a hypothetical algorithm, giving up on the complexity on one parameter to slightly improve the dependence on the other, it is always possible to cook up an appropriate value of Δ to show that this cannot be done, unless the exponent contains a term that dominates $\Delta \cdot \mathrm{tw}$. Hence, we have a convincing argument that $2^{O(\Delta\mathrm{tw})}$ is the right answer.

[4]Note that we use the word "parametric" to mean something different from "parameterized". Our reduction is parametric in the sense that we can feed into it a desired value in order to set some properties of the produced instance.

What have we learned? It is important not to forget that one of the pitfalls of complexity theory is that we always focus on worst-case analysis, that is, all our (conditional) statements are of the form "it is impossible to do better than this algorithm *in the worst case*". When dealing with problems involving multiple parameters, this can lead to an interesting temptation: when we manage to find some combination of the parameters for which a lower bound argument proves that our algorithm is best-possible, we might think that we can declare victory: no algorithm can beat our algorithm for this case, therefore no algorithm can beat our algorithm in all cases, therefore, the worst-case complexity of our algorithm seems optimal. This line of reasoning is not particularly convincing, and indeed branches of theoretical computer science where it is customary to be more precise with running times are well aware of these distinctions (e.g. the distinction between $O(n^2)$ and $O(n+m)$ algorithms in graph theory is widely considered important, even though in the worst case some graphs have $m = \Theta(n^2)$ edges).

The lesson is that in the context of problems with several parameters, whenever it is possible to avoid talking about the worst case, we should indeed try to avoid talking about the worst case, because this may implicitly insert into our reasoning some assumption about the relation between our parameters. In the particular example we presented in this section the story in the end did not change: our more sophisticated lower bound argument just confirms what we could have "proved" with a more straightforward reduction. Nevertheless, it would be surprising if this were always the case. There must exist multi-parameter problems where different combinations of parameter values should be amenable to completely different algorithms. Concretely, for our particular example of parameters, there surely exist problems for which a certain parameterized algorithm is better when $\Delta \gg \text{tw}$ and another, completely different algorithm is better when $\Delta \ll \text{tw}$. The pitfall we are worried about is therefore that one could discover the former algorithm, then construct a reduction where $\Delta \gg \text{tw}$ and this algorithm is optimal, and then call it a day and stop, thinking that the problem is "settled"! To avoid this pitfall, we have to be aware of the possibilities and either use a parametric reduction, like the one of this section, to close the problem in all cases, or continue searching for an algorithm that evades possible partial lower bounds. It must be noted that this type of investigation is still in its very early stages: multi-variate fine-grained complexity theory is an exciting new frontier and for the moment few works have carefully looked at these issues.

4.2 Super Fine-Grained Lower Bounds Using the SETH

Our goal in this section is to present some lower bound results which offer even more precision than the ETH-based lower bounds of Section 4.1. In order to do this we will inevitably have to rely on a stronger assumption, the SETH. It is important to note that, even though the ETH is a somewhat well-accepted complexity hypothesis in our community today, the SETH is considered to be on much less solid ground and it would be considered a much smaller surprise if it turned out that SAT can in fact be solved in 1.99^n, than if 3-SAT could be solved in, say $2^{n/\log n}$. Despite this, SETH-based lower bounds are still of much value, because significant effort has been expended in discovering an algorithm for SAT running in 1.99^n and so far such efforts have failed. Even if we are not confident that these efforts are doomed, it is important to discover that solving a seemingly unrelated problem would make great algorithmic progress beyond the state of the art for SAT, because such connections imply that there exist non-trivial barriers to moving forward.

An interesting phenomenon that we will encounter is that very often the lower bounds we obtain via the SETH match precisely the running times of the best known algorithms. In particular, the results we present in this section (assuming the SETH) are:

- For each fixed $k \geq 3$, the complexity of k-Coloring parameterized by clique-width is $(2^k - 2)^{\text{cw}}$, that is, the "correct" bases of the exponents for 3−,4−, and 5-Coloring parameterized by clique-width are respectively $6, 14, 30$, in the sense that we have matching algorithms and lower bounds (Section 4.2.2).
- For each fixed r, the complexity of (k,r)-Center parameterized by clique-width is $(3r+1)^{\text{cw}}$.

In particular, for $r = 1$ this implies that the correct base for Dominating Set parameterized by clique-width is 4 (Section 4.2.3).

- For several problems parameterized by treewidth or pathwidth, the natural dynamic program gives the correct base. These include d-Scattered Set, Upper Dominating Set, Mixed Dominating Set, and d-Orientable Deletion (Section 4.2.4).

A nice aspect of these results is the diversity in the bases they imply: several of them are parametric (for each fixed problem-specific parameter k we get a different base of the exponential), while the non-parametric ones, such as Mixed Dominating Set and Upper Dominating Set have random-seeming bases (5 and 6 respectively) which happen to agree with corresponding algorithms. In all cases, the hard(er) part is establishing the lower bound, which comes via a reduction from SAT. One technical difficulty of such reductions is how to transform the base of the exponential of the running time promised by the SETH (which is $2 - \epsilon$) into $5 - \epsilon$, $6 - \epsilon$, or whatever constant we need for the problem at hand. In Section 4.2.1 we present one possible technique, which is based on reductions from CSPs with appropriately chosen alphabet size. This technique allows to significantly simplify many reductions.

The bigger picture. The highlight of this section is probably the results of Sections 4.2.2 and 4.2.3 which settle the complexity of two standard problems (k-Coloring and Dominating Set) parameterized by clique-width. However, if we take a look at the bigger picture, what is most striking is for how many problems the SETH implies a tight lower bound when parameterizing by treewidth or clique-width. The collection of such results was started by [140], who showed for instance that the correct complexity for Dominating Set is 3^{tw}. Already, it's surprising that the SETH gives the correct constant here, but the story goes on: there was a priori no good reason why connected versions of standard problems would add $+1$ to the base of the exponent, but the Cut&Count method does exactly that and is optimal under the SETH [51]; this phenomenon continues to hold with the same tightness for domination at distance r [37], which has complexity $(2r+1)^{\mathrm{tw}}$ in the plain version and $(2r+2)^{\mathrm{tw}}$ in the connected version and again the SETH-based bound falls in exactly the right place; similarly, the reasons the algorithm of [119] for distance r domination has complexity $(3r+1)^{\mathrm{cw}}$ seem kind of random, but the SETH-based bound matches it exactly; most striking of all, Cygan et al. [50] give an algorithm for Hamiltonicity running in $(2+\sqrt{2})^{\mathrm{pw}}$ and the seemingly random constant $2+\sqrt{2}$ turns out to be optimal under the SETH!

We could go on listing more such results, but perhaps what is most surprising about this line of research is how well everything seems to fall into place. Obviously, this makes it even more desirable for the SETH to be true. But in a sense this situation is also reminiscent of the theoretical computer science community's fascination with the Unique Games Conjecture in the mid 2000's. One of the most striking points of that story was the discovery that the UGC implied that the seemingly arbitrary irrational approximation ratio of the famous Goemans-Williamson algorithm for Max Cut is optimal [123]. Just like with the UGC back then, the fact that the SETH seems to always be exactly right seems to indicate, if not necessarily that the conjecture is true[5], at least that the conjecture allows us to put our fingers on a real complexity phenomenon. In this light, even though we continue this line of research mainly because we want to use the SETH to understand our structurally parameterized problems, this body of work has grown to the point where one could argue that these results also give some insight about the SETH itself.

4.2.1 The SETH and CSPs with Larger Alphabet

The SETH states, informally, that as SAT clauses become larger, eventually the best algorithm for SAT is simply to try out all possible assignments to all variables. In this section we present a result from [130] showing that the same is essentially true for CSP with a larger, non-binary alphabet. The interest in presenting such a result is that very often we seek to prove a SETH-based lower bound showing that a problem does not admit an algorithm running in c^w, for some constant

[5]That would be proof by wishful thinking!

c and width parameter w (such as treewidth or clique-width). This becomes complicated when we reduce directly from SAT if c is not a power of 2 as one cannot make a one-to-one correspondence between binary SAT variables and "units of width" in the new instance, which are intended to encode c choices. As a result, essentially all known SETH lower bounds of this form published before [130] include as part of their construction a group gadget, which maps every t variables of the original SAT instance to p elements of the new problem, for appropriately chosen integers p, t (see e.g. [37, 51, 112, 140]). Such gadgets are, however, often cumbersome to design, because they require a problem-specific trick that expresses a mapping of assignments from a binary to a non-binary domain. We therefore prefer to construct a custom-made CSP with a convenient running time bound. Such a CSP allows us to reduce directly to the problem we are interested in, in a way that maps exactly one variable to one unit of width. Thanks to this intermediate problem the SETH-based bound of [130] (and several later works [62, 63, 83]) were significantly simplified, as it was no longer necessary to worry about a discrepancy between the bases of the exponentials.

We state the following theorem without a proof. The proof can be found in [130] and follows ideas which were already present in [140]. The problem q-CSP-B is defined as follows: we are given n variables which take values over an alphabet of size B, and a collection of m constraints over these variables, where each constraint has arity at most q. Since we are interested in the case where q, B are absolute constants, the encoding of the constraints is not crucial; we can consider for example that for each constraint we are given a list of all the assignments to its variables that satisfy the constraint. The objective is to find an assignment to the variables that satisfies all the constraints or conclude that no such assignment exists.

Theorem 1 *For any $B \geq 2$, $\epsilon > 0$ we have the following: if the SETH is true, then there exists a q such that n-variable q-CSP-B cannot be solved in time $O^*\left((B-\epsilon)^n\right)$.*

Note that q-SAT is of course a special case of q-CSP-2. Essentially, what Theorem 1 is saying is that if brute-force is the best possible algorithm when variables have two possible values, then it is also the best algorithm when variables have more values. This will allow us to start our reductions from problems for which the running time lower bound has a more convenient form.

4.2.2 Coloring parameterized by Clique-width

In this section we present some results from [130] which imply that, under the SETH, the complexity of k-Coloring parameterized by clique-width is precisely $(2^k-2)^{\mathrm{cw}} n^{O(1)}$, for any fixed $k \geq 3$.

Background. k-Coloring is of course one of the most well-studied problems in theoretical computer science. With respect to treewidth, it has long been known that k-Coloring is FPT, because we can assume without loss of generality that $k \leq \mathrm{tw}$ (otherwise the answer is trivially Yes). We remark that this argument is akin to the arguments of Section 4.1.2 (for d-Scattered Set parameterized by tree-depth) and 3.2.1 (for Grundy Coloring parameterized by path-width), in the sense that the standard k-Coloring algorithm does not a priori seem to be FPT: we need to consider all k-colorings of the vertices of the bag, and in general k could be as large as $\Theta(n)$. Fixed-parameter tractability is therefore obtained via a (somewhat unrelated) combinatorial argument, which allows us to bound the optimal, without supplying a truly superior algorithm. In this light, it is not surprising that generalizations such as List Coloring are W[1]-hard by treewidth [75], or that the fine-grained analysis of this problem reveals that, despite its fixed-parameter tractability, the problem is essentially still hard. In particular, [141] shows that, under the ETH, Coloring cannot be solved in $\mathrm{tw}^{o(\mathrm{tw})} n^{O(1)}$, while [140] shows that under the SETH k-Coloring cannot be solved in $(k-\epsilon)^{\mathrm{tw}}$ for any $k \geq 3, \epsilon > 0$.

The results above seem to indicate that k-Coloring is a problem general enough that we would expect the "easy" algorithm to be optimal in most cases. How does this intuition translate to clique-width? First, we observe that in the context of clique-width we lose the combinatorial

bound on k, since already cliques have unbounded chromatic number. As expected, this translates into a loss of fixed-parameter tractability and Coloring was one of the first problems to be shown W[1]-hard by clique-width. In fact, it was recently shown more strongly that, under the ETH, Coloring cannot be solved in $n^{2^{o(\mathrm{cw})}}$, so the best XP algorithm is double-exponential in the parameter, and this is tight [87]. It therefore arguably makes more sense to focus on the case where k is fixed. Here, k-Coloring was known to be FPT via an algorithm of Kobler and Rotics running in $4^{k\cdot\mathrm{cw}}n^{O(1)}$ [125]. The contribution of the results of this section is to improve this algorithm to $(2^k-2)^{\mathrm{cw}}n^{O(1)}$ and then, more importantly, to show that this improvement is best possible under the SETH.

What is the correct base? As we already explained, one of the most striking aspects of this line of research is that SETH-based lower bounds have a tendency to always land on exactly the right spot. But, looking at this problem for this parameter, what would we expect to be the right base of the exponential? To answer this, we should take another look at the algorithm of [125]. The strategy of this algorithm is, as usual, dynamic programming. In this case, for each label in our clique-width expression we need to store *a subset* of $\{1,\ldots,k\}$, signifying which colors are used at least once in the vertices of that label in the current partial solution. This leads to a table size of $2^{k\cdot\mathrm{cw}}$. The running time then deteriorates to $4^{k\cdot\mathrm{cw}}$ because in Join nodes we consider every combination between every partial solution of one graph with every partial solution of another.

Looking at this we first realize that the algorithm of [125] predates the discovery of fast subset convolution techniques and their applicability to such problems [168]. The goal of such techniques was exactly to deal with problems where some parts of the dynamic programming algorithm took quadratic time in the size of the table (because we had to try all elements of one table in combination with all elements of another) and speed up such algorithms to running time quasi-linear in the size of the table. It stands to reason that applying such techniques to our problem would speed up the algorithm to $2^{k\cdot\mathrm{cw}}$ and indeed, this is part of the improvement given in [130]. However, 2^k is also not the right answer. To see this, note that an easy way to tighten the analysis of our algorithm is to observe that the set of colors used for a non-empty set of vertices cannot be empty in any valid solution. Hence, instead of 2^k sets, we may consider "only" 2^k-1 (which makes a non-trivial difference for small values of k). Somewhat surprisingly though, 2^k-1 is also not the right answer. Continuing this line of reasoning we can also observe that a set of vertices that uses all k colors cannot acquire any new neighbors (via a Join operation). Hence, by scanning forward in the clique-width expression to determine if a label set has further neighbors, our algorithm can further discount one possible set. This leads to a base of 2^k-2, and an algorithm with complexity $(2^k-2)^{\mathrm{cw}}n^{O(1)}$ is given in [130] using these simple observations and fast subset convolution techniques.

The lower bound. The more interesting part is now that this base of 2^k-2 turns out to be optimal for all $k\geq 3$ (assuming the SETH). This is established via a reduction from q-CSP-B with n variables. Without going into the details of the gadgets, the general strategy is to construct a graph (and a clique-width expression) where the main part contains vertices partitioned into n label sets. The *set* of colors we use for each such set will encode the assignment for a variable of the CSP instance. For example, let us consider $k=4$, and in this case we are reducing from q-CSP-14, because $2^k-2=14$. We set up a global equivalence relation between values in our CSP alphabet and subsets of $\{1,\ldots,4\}$ (excluding the sets $\emptyset$ and $\{1,2,3,4\}$). For example, we could say that the value 7 is represented by the set $\{1,3\}$. We start our construction with n independent sets of size 4, with the intended meaning that the colors used for each set I_i, $i\in\{1,\ldots,n\}$ will encode the assignment to the variable x_i of the CSP. We then add a (constant clique-width) gadget for each constraint with the following property: we are forced to give a special color to a vertex of the gadget, and the vertex which receives this color encodes the selection of the satisfying assignment to this constraint. Suppose that x_i appears in this constraint, and that the particular assignment sets $x_i=7$. Then, the special color forces us to color two particular vertices with $\{1,3\}$, and two others with $\{2,4\}$. We then perform a Rename operation to place the first two vertices into I_i

(if the coloring of I_i does indeed encode $x_i = 7$ this does not change the colors used in I_i) and a Join operation between the other two vertices and I_i (to ensure that I_i is not using other colors). Repeating this for all constraints allow us to check that the coloring of the initial independent set encodes a satisfying assignment.

Even though the details of the construction are somewhat complicated (and we have not at all touched on the description of the specific gadgets of [130]), the fundamental idea turns out to be exactly the idea of the algorithm. We need to make sure that we have a collection of n label sets, such that the best we can hope to do is consider all possible colorings of these sets, i.e., all sets of colors that could be used inside them. This confirms the intuition that k-Coloring is a problem sufficiently hard that we are forced to try essentially all possibilities.

4.2.3 Dominating Set Parameterized by Clique-width

In this section we revisit the results of [119] on (k, r)-Center. Recall that in this problem we are given a graph and are asked to select k vertices so that all other vertices are at distance at most r from a selected vertex. We will consider k as part of the input and focus on two parameters: r and cw. More precisely, our main result determines precisely the complexity of this problem parameterized by cw for every fixed value $r \geq 1$ and shows that it is $(3r + 1)^{\text{cw}}$, assuming the SETH.

The reason we devote a section to this result is that for $r = 1$, our result includes as a special case a tight bound on the complexity of Dominating Set parameterized by clique-width, and Dominating Set is a standard problem. Before [119] the best lower bound for Dominating Set was due to [140], who showed that the problem does not admit a $(3 - \epsilon)^{\text{pw}} n^{O(1)}$ time algorithm; since cw $\leq$ pw for all graphs, this also implies that the problem does not admit a $(3 - \epsilon)^{\text{cw}} n^{O(1)}$ time algorithm. On the other hand, the best known algorithm has complexity $4^{\text{cw}} n^{O(1)}$, due to Bodlaender et al. [33]. Hence, for $r = 1$ our results close the gap between 3 and 4 and confirm that the algorithm of [33] is optimal. Nevertheless, for the sake of completeness, we will sketch the main ideas of our results for general r, and the application to Dominating Set will follow simply by setting $r = 1$.

Algorithmic Intuition. The main algorithmic result of [119] with respect to clique-width is that (k, r)-Center can be solved in $(3r + 1)^{\text{cw}} n^{O(1)}$. Here, we give some intuition why the size of the dynamic programming table is $(3r + 1)^{\text{cw}}$, without focusing on how to make the algorithm quasi-linear in the size of the table (this can be done using fast subset convolution techniques). The first step is to recast the problem as a distance-labeling problem: we want to assign to each vertex of the graph a number from $\{0, \ldots, r\}$, so that (i) at most k vertices are assigned 0 (ii) each vertex assigned $i > 0$ has a neighbor assigned at most $i - 1$. Note that this recasting simply says that we label every vertex with its distance from the closest selected vertex, but it allows us to verify a solution locally (i.e. checking only the neighborhood of each vertex).

Given this recasting, the $(2r + 1)^{\text{tw}}$ complexity obtained by Borradaile and Le [37] is quite natural: for each vertex in a bag we have to keep track of its distance label ($r + 1$ choices); furthermore, if this label is positive, we have to keep track of whether we have already seen the neighbor who has lower distance (another r choices), resulting in $2r + 1$ possible states per vertex. The situation for clique-width is a bit more complicated.

Recall that in clique-width we are working with a partition of the graph into cw sets of vertices. We could decide to remember for each set which distance labels appear in this set in the current partial solution, but this would end up around $2^{r \cdot \text{cw}}$ (this is similar to the Coloring algorithm of Section 4.2.2). A more clever observation is that what is important in each set is the *smallest* distance value and the *smallest* currently unsatisfied distance value. This is because, if when we perform a Join operation we manage to give the missing neighbor to a vertex of distance label i, we will automatically satisfy vertices with higher labels as well. This naturally leads to a DP table of size around $(r + 1)^{2\text{cw}}$. We are clearly not done yet.

For the next improvement, we consider a set of vertices of the clique-width expression, and say that the smallest distance label that appears in this set is i and the smallest unsatisfied distance

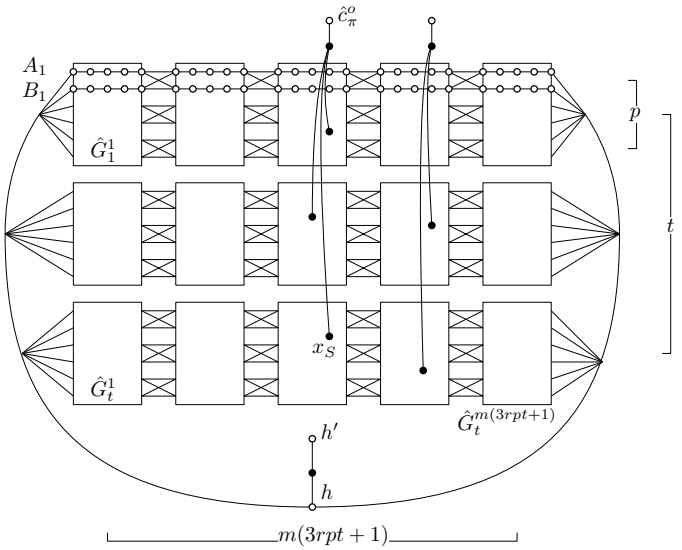


Figure 4.2: Overview of the construction of [119] showing that (k, r)-CENTER cannot be solved in time less than $(3r + 1)^{\text{cw}} n^{O(1)}$ under the SETH.

label is $j \geq i$. We now observe that if $j \geq i + 2$, we have two cases: either a Join operation on this set will be executed at some point later, or not. In the former case, all currently unsatisfied labels of this set will then become satisfied, so we don't have to worry about them; in the latter, the currently unsatisfied vertices will remain unsatisfied in the final graph, so we can discard this partial solution. Note that what we are doing here is the somewhat unusual step of looking *later* in the clique-width expression to allow us to decrease the size of the table[6]. What this gives us is that actually we have much fewer than $(r + 1)^2$ states for each step. More carefully, we observe that we have $r + 1$ choices for the smallest distance label i that appears in the set; and then the smallest unsatisfied label can only be i, $i + 1$, or everything is satisfied. Hence, we have at most $3r + 3$ states per set, leading to a table size of $(3r + 3)^{\text{cw}}$.

Finally, let's put the finishing touches on our argument. If the smallest distance label of a set is 0, the smallest unsatisfied distance label cannot also be 0, since vertices with label 0 are automatically satisfied; we thus save one case. Furthermore, if the smallest label is r, the smallest unsatisfied label cannot be $r + 1$, as we only use labels up to r; this saves one more case. We thus arrive at $3r + 1$ states per set of vertices, which explains the complexity of our algorithm.

The Lower Bound. Reviewing the arguments of the previous paragraph, it doesn't necessarily seem obvious that $3r + 1$ should be the right answer. Indeed, many of the arguments we used in our algorithm seem a bit ad-hoc, so one may be tempted to think that more such arguments may allow us to shave a bit more off the complexity (perhaps $(3r)^{\text{cw}}$ would seem more natural as a bound). It is therefore a bit surprising that we are able to come up with a tight lower bound, proving that under the SETH this random-seeming collection of ad-hoc arguments is best possible.

How does the lower bound work? A very high level view of the construction is given in Figure 4.2. The key idea is that the construction is made up of a collection of many parallel paths. This is a common pattern in such constructions: indeed, the key idea in the $(2r + 1)^{\text{tw}}$ lower bound of [37] is to construct parallel paths where the intended solution is to select one out of every $2r + 1$ consecutive vertices. This gives us $2r + 1$ choices per path (allowing us to encode, say, the assignment for a q-CSP-$(2r + 1)$ variable), and this choice is propagated throughout the graph because the most efficient way to dominate a path at distance r is indeed to take one out of every $2r + 1$ vertices.

[6]This is similar to the trick we use in Section 4.2.2 to discard solutions which use all k colors.

The idea we use here is the same, except we have to compress $3r+1$ choices per unit of clique-width. Therefore, we construct more paths, but for every *pair* of paths, we periodically (every $2r+1$ vertices) perform a Join operation that connects the two paths with cross edges (see the paths A_1, B_1 in Figure 4.2). This has two effects. One, it allows us to decrease the clique-width by a factor of 2: if we intended to construct, say $2n$ paths, building them in an independent "parallel" way, as was done in the construction of [37] would probably lead to a clique-width of $2n$. Here, we can use the same clique-width label for each pair of paths, so the clique-width is only n. This is good, because it allows us to make the reduction more efficient. Second, on the other hand connecting the paths in this way makes the reduction less "reliable" in the sense that information about our choice on one path leaks out onto its sibling path. In particular, if we look at the paths A_1, B_1, if the two paths were parallel we would be looking at $(2r+1)^2$ reasonable solutions that the two paths may reliably represent. Here, the space of reliably representable solutions diminishes, because solutions which were previously distinct (e.g. taking the third vertex of each block of A_1 and the fifth of each block of B_1, or doing the opposite) now become equivalent (that is, the solution may alternate between the two indefinitely without increasing the cost).

The hope of this construction is then that, even if gluing two paths together in this way may not give us the capacity to squeeze $(2r+1)^2$ choices into one unit of width (one clique-width label), it may still give us the capacity to squeeze more than $2r+1$ choices into a unit of width. We will not go through the detailed case analysis which is performed on this topic in [119], but it turns out that it is possible to write down all the $(2r+1)^2$ choices one can make them for the two paths and partition them into exactly $3r+1$ classes, which can be totally ordered in a way that, if in one section the solution has made choice α, in the following section the solution has made choice β, and $\beta \prec \alpha$ in the ordering, then the solution is invalid. In other words, we can partition all possible types of solutions into $3r+1$ classes, in a way that if the solution is not consistent at some point, it must switch to a type that comes later in our ordering. Since our ordering does not have cycles, the solution must eventually stabilize into something consistent, allowing the reduction to go through.

Bottom line. Looking back at our arguments for the algorithm and the lower bound, perhaps what is a bit striking is that the two end up giving bounds that match. On the algorithmic side, we rely on several ad-hoc observations; on the reduction side, we rely on a case-by-case analysis of the reduction. It must also be observed that one could a priori have hoped to make the reduction more efficient (e.g. why don't we glue together three paths instead of two? or we could remove one of the cross edges connecting two paths without increasing the clique-width). Surprisingly, whether one would like to dig a bit deeper on the algorithmic side, or try a few more tricks to improve the reduction, it just so happens that this would be impossible without falsifying the SETH! It therefore seems that $3r+1$ is the right bound for this problem parameterized by clique-width, whether this was clear from the beginning or not, at least for people who espouse the point of view of this work that the SETH is probably true.

4.2.4 Other Tight bounds Parameterized by Treewidth/Pathwidth

In the previous two sections we chose to make a more detailed exposition of results on k-Coloring and (k,r)-Center (and its special case Dominating Set) because of the importance of these standard problems. Nevertheless, as we have mentioned, one striking aspect of SETH-based lower bounds is that so far they have seemed to always fall into exactly the right place, even when it comes to dealing with more exotic problems. In this section we briefly review a few more results of this type, where for four different, less well-known problems we show that the SETH correctly predicts the precise complexity of their parameterization by treewidth or pathwidth. Rather than giving technical details, we simply informally describe the problems and try to give a basic intuition why the bounds end up being what they are.

Scattered Set. As we have mentioned, this problem was considered in [120]. In this problem, which is a natural generalization of Independent Set, we are given a graph and an integer $d \geq 1$

and are asked to select a set of vertices of maximum size such that any two vertices of the set are at distance strictly greater than d [7]. In [120] the parameter dependence of this problem was shown to be $(d+1)^{\text{tw}}$ and $(d+1)^{\text{pw}}$, under the SETH. To give some intuition where this running time comes from, recall how we recast (k,r)-Center as a distance labeling problem. We can do the same here and equivalently assign a label from $\{0,\ldots,d\}$ to each vertex so that (i) the set of vertices with label 0 is maximized (ii) for every vertex v with label $i>0$, there is no neighbor of v with label $\leq i-2$. Again, we are labeling the vertices according to their distance from the closest selected vertex. Since there are $d+1$ possible labels, it is natural to obtain a DP of size $(d+1)^{\text{tw}}$ and, as usual, to make the running time quasi-linear in the table size we rely on fast subset convolution techniques. For the lower bound, we follow the same general technique as [37] did for (k,r)-Center, namely, we construct n parallel paths, and from each path we are supposed to select one out of each $d+1$ consecutive vertices. A solution that packs as many vertices in its selection as possible will maintain the same choice throughout for each path, hence this allows us to encode an assignment to q-CSP-$(d+1)$[8]. It therefore seems that, assuming the SETH, this problem behaves exactly as expected parameterized by treewidth and pathwidth.

Mixed Dominating Set. This problem was considered in [62]. In this problem we are given a graph and are asked to select a "mixed" set, containing both vertices and edges, that dominates all vertices and all edges. Equivalently, we are asked to find a $(k,2)$-Center in the graph obtained if we subdivide every edge once. In a previous paper [113] it had been shown that this problem can be solved in $6^{\text{tw}}n^{O(1)}$ and $5^{\text{pw}}n^{O(1)}$, but not in $(2-\epsilon)^{\text{pw}}n^{O(1)}$. The work of [62] (which was part of the thesis of Louis Dublois, co-supervised with Vangelis Paschos) closed these gaps by giving an improved algorithm of complexity $5^{\text{tw}}n^{O(1)}$ and a lower bound showing that the problem cannot be solved in $(5-\epsilon)^{\text{pw}}n^{O(1)}$. The idea of the algorithm was simply to reduce the problem to $(k,2)$-Center as described above and invoke the algorithm of [37]. The idea of the lower bound was, as usual, to construct n parallel paths to represent the n variables of a q-CSP-5 instance. The 5 natural states of the DP, which the reduction needed to represent, are (i) a vertex is selected as a vertex (ii) a vertex is an endpoint of a selected edge whose other endpoint has been seen (iii) a vertex is an endpoint of a selected edge whose other endpoint has not been seen (iv) a vertex is not selected and is not yet dominated by a neighbor (v) a vertex is not selected but has been dominated. Note that this DP does not reserve special states for dominated or not-dominated edges, as the status of edges follows indirectly from that of vertices and, since the upper and lower bounds match, this is optimal.

Upper Dominating Set. This problem was considered in [63]. We are given a graph G and are asked to select a maximum cardinality set $S\subseteq V(G)$ such that S dominates $V(G)$, but no proper subset of S dominates $V(G)$. In other words, we are looking for the maximum minimal dominating set of G. It had previously been shown in [11] that this problem admits an algorithm of complexity $7^{\text{pw}}n^{O(1)}$, however using a DP table of size 6^{pw}. The idea of the DP table is that every selected vertex u of a feasible solution must have a private neighbor, that is, a vertex that is not dominated by $S\setminus\{u\}$ (this private neighbor could be u itself). Hence, vertices have six natural states in the DP: (i) selected, and the private neighbor is another vertex that has already been seen (ii) selected, and the private neighbor has not yet been seen (iii) selected, and the private neighbor is the same vertex (iv) not selected, not yet dominated (v) selected, dominated, serves as someone's private neighbor (vi) selected, dominated, is not someone's private neighbor. The first contribution of [63] was to slightly speed up the handling of some cases of the DP algorithm to make the running time quasi-linear in the table size. We thus get an upper bound of $6^{\text{pw}}n^{O(1)}$. More importantly, [63] gave evidence that this is optimal, via a reduction which, as usual, creates n parallel paths. The paths are partitioned into blocks of 4 vertices, call them a,b,c,d, and the

[7] Sometimes the problem is defined as requiring that vertices be at distance *at least* d. This is clearly equivalent, but we prefer the current definition because it makes the problem equivalent to Independent Set for $d=1$.

[8] Note that since [37] was published before [130], the reduction given there is directly from SAT, rather than q-CSP, but it is easy to simplify the reduction in the way we describe here.

intended solutions are any pair of these four vertices, that will be repeated throughout the path. Interestingly, it remains open whether the complexity of this problem by treewidth has the same base in the exponent.

Orientable Deletion. This problem was considered in [104]. We are given a graph and an integer d and are asked to orient the edges so that all vertices have in-degree at most d. Remarkably, this problem is polynomial-time solvable, so we consider the more general version where we are asked to delete the minimum number of vertices so that the graph becomes orientable (since d-orientability is hereditary, this is trivially NP-hard). The special case where $d = 1$ is known as Pseudoforest Deletion [32]. We first remark that the natural DP algorithm for this problem has complexity $(d+2)^{\mathrm{tw}}n^{O(1)}$: for each vertex v we either delete v, or if we do not, every time we introduce one of its neighbors u in the bag, we have to keep track of whether (u, v) is oriented towards v or u, and hence the current in-degree of v, which must be in $\{0, \ldots, d\}$. Hence, we have $d+2$ possible states for each vertex. Indeed, for Pseudoforest Deletion, [32] gave a $3^{\mathrm{tw}}n^{O(1)}$ time algorithm. The contribution of [104] was to show that this is best possible, and more strongly that for any $d \geq 1$, d-Orientable Deletion cannot be solved faster than $(d+2)^{\mathrm{pw}}n^{O(1)}$.

Chapter 5

Approximation

In this chapter we consider how the ideas of the previous chapters can be combined with the notion of approximation algorithms. Recall that, as we discussed in Chapter 1, one of the main motivations for the development of approximation algorithms was the hope that NP-complete problems were only hard because we were asking to find an exact solution. In the context of polynomial-time approximation algorithms for most graph problems, these hopes turned out to be too optimistic: most problems are APX-hard, and hence they are not only NP-hard to solve exactly, but also NP-hard to approximate within some particular constant accuracy. Nevertheless, approximation algorithms have so far been much less explored in the context of parameterized complexity, and even less so in combination with the structural parameters that we focus on in this work. As a result, there is still hope that approximation schemes, that is, algorithms which are able to approximate the optimal value to arbitrarily great accuracy, may not be such a rare occurrence in this domain. In particular, even the negative results which have recently appeared regarding the FPT inapproximability of k-Clique and k-Dominating Set [24, 42, 137, 138], do not preclude that many problems which are W[1]-hard parameterized by, say, treewidth, admit FPT approximation schemes[1].

In this chapter we present some recent results indicating that combining approximation with structural parameterizations may indeed be a very fruitful direction. The bulk of this chapter is devoted to the presentation of a general technique, introduced in [129] and then refined and used in several follow-up works, for adapting exact dynamic programming algorithms that run in XP time parameterized by treewidth into FPT approximation schemes. The main intuition of this technique is quite appealing: we have already observed (see e.g. Section 3.2.3) that a reason many natural problems are W[1]-hard parameterized by treewidth is that the straightforward dynamic programming algorithm needs to store a number for each vertex in a bag. The core idea of the technique of [129] is to speed up this process by storing approximate values in the dynamic program. Even though this sounds conceptually simple, much care is needed to make sure that rounding errors do not pile up, so this technique cannot always be applied. It has, however, found applications in a wide variety of situations, as we explain in Section 5.1.

We then move on to address a problem we already visited in Section 3.3.1, where it was shown that SAT is W[1]-hard parameterized by neighborhood diversity. Here, we consider whether it is possible to work around this hardness by trying to approximately solve Max-SAT, the problem where we want to find an assignment that satisfies as many clauses as possible. In Section 5.2 we present a positive result in this direction which shows that an FPT approximation scheme is achievable even in the much more general parameterization by clique-width. The core idea is a Win/Win approach which either determines that the instance is so dense that setting variables at random is likely to satisfy the vast majority of clauses, or reduces the problem to the case where we parameterize by treewidth.

Finally, in Section 5.3 we continue in the theme of dense instances being easy to solve approx-

[1]This is because typical reductions for such problems are *not* approximation preserving.

imately and consider approximation algorithms in the context of a structural parameterization that has a different flavor from most of the other results of this work. We recall that Max-Cut is known to admit a PTAS when the input graph is dense [6]. Treating this as an island of tractability, we investigate how the complexity of the problem changes as the average degree of the input graph decreases, keeping in mind that for sparse graphs the problem cannot be approximated arbitrarily well even in time $2^{n^{1-\epsilon}}$ (under the ETH). We show that the average degree is a crucial parameter here, because for graphs of average degree Δ, the time complexity needed to obtain an approximation scheme is exponential in n/Δ and this is best possible under the ETH.

5.1 Rounding and Dynamic Programming

Our goal in this section is to explain a general algorithmic technique, introduced in [129], for obtaining FPT algorithms parameterized by treewidth (and similar parameters) by using dynamic programming and rounding. We first give a high-level explanation of the key ideas in Section 5.1.1 and then review a few examples of applications in Section 5.1.2 to give the reader an idea of the kind of problem that this technique can deal with. We then consider the converse question and close this section by visiting an example where it is *not* possible to obtain an FPT approximation scheme using this technique (or any technique, under standard assumptions) in Section 5.1.3.

5.1.1 Main Idea

Let us review the basic idea of our technique via a representative example from [3]. Recall the Power Vertex Cover problem, discussed in Section 3.2.3. In this problem we are given a graph with positive integer weights on the edges. The goal is to assign a value (power level) to each vertex so that (i) the sum of assigned values is minimized (ii) for each edge, at least one of its endpoints has a power level that is greater or equal to the weight of the edge. In other words, assigning power level p to a vertex v allows us to cover all edges incident on v with weight at most p (but does not cover edges with larger weights).

Recall that in Section 3.2.3 we explained that this problem can easily be solved in time $(W+1)^{\mathrm{tw}}n^{O(1)}$, where W is the maximum weight of the input graph. The algorithm performs dynamic programming and stores, for each vertex of a bag, the power level we have assigned to this vertex, which must be an integer from 0 to W. The first problem that poses itself is that this algorithm is only pseudo-polynomial when treewidth is constant: W could be exponential in the size of the input (if numbers are written in binary). If we are interested in a $(1+\epsilon)$-approximation, though, this can be handled via standard techniques for turning pseudo-polynomial time algorithms into approximation schemes, namely, it suffices to divide all weights by an appropriately chosen value. The details of this part are explained in [3], but we do not focus more on this phase of the algorithm, as the ideas are similar to those for classical problems, such as Knapsack.

The case we are interested in, then, is when W is polynomially bounded in n, say, $W = O(n^2)$. As we mentioned in Section 3.2.3, the problem is still W[1]-hard parameterized by treewidth in this case, so solving it exactly in time less than $n^{O(\mathrm{tw})}$, which matches the performance of the trivial DP algorithm, seems impossible. This is where our approximation technique comes into play.

Suppose that we are willing to accept a solution that is at most $(1+\epsilon)$-times worse than the optimal, where $\epsilon > 0$ is some arbitrary supplied desired parameter. The key idea now is that we will round all the values used in our instance to integer powers of $(1+\epsilon)$. In other words, when an edge originally had weight w, we will replace its weight with $(1+\epsilon)^{\lceil \log_{(1+\epsilon)} w \rceil}$. Notice that we have rounded the exponent up, meaning that we may have increased the weight of the edge by a factor of $(1+\epsilon)$ (at most). Nevertheless, doing this for all edges cannot increase the value of the optimal solution by more than a factor of $(1+\epsilon)$, as taking any feasible solution of the old instance and rounding up the power levels assigned to vertices in the same way gives a feasible solution of the new instance.

What have we gained from this manipulation? Thanks to the rounding step, the number of possible power levels that we need to consider for each vertex is now at most $1+\log_{(1+\epsilon)} W$, because the only reasonable power levels to consider are the values which also appear as weights of an edge, and now the weights of edges are integer powers of $(1+\epsilon)$ upper bounded by $(1+\epsilon)W$. Assuming $W = O(n^2)$, we have $\log_{(1+\epsilon)} W = \frac{O(\log n)}{\log(1+\epsilon)} = O(\frac{\log n}{\epsilon})$, where we have used the approximation $\log(1+\epsilon) = O(1/\epsilon)$, which is valid for small enough ϵ. Hence, the easy dynamic programming algorithm runs in time $\left(\frac{\log n}{\epsilon}\right)^{O(\mathrm{tw})} n^{O(1)}$. Observe that via standard arguments this running time is FPT in $\mathrm{tw}+\frac{1}{\epsilon}$: if $\mathrm{tw} < \sqrt{\log n}$ then $(\log n)^{O(\mathrm{tw})} = n^{o(1)}$, while if $\sqrt{\log n} < \mathrm{tw}$, then the algorithm runs in $\left(\frac{\mathrm{tw}}{\epsilon}\right)^{O(\mathrm{tw})} n^{O(1)}$. We have therefore obtained a $(1+\epsilon)$-approximation for a problem that is W[1]-hard to solve exactly parameterized by treewidth.

It's not always so easy. The example with Power Vertex Cover has the advantage of being clear and nicely captures one main idea of the technique of [129]: namely, that values should be stored in a way that ensures a multiplicative error of $(1+\epsilon)$, that is, as integer powers of $(1+\epsilon)$. Nevertheless, Power Vertex Cover is a rather misleading example, because the corresponding DP is very simple: once we fix the power level of a vertex, the only arithmetic operation we perform on this number is to compare with the weights of edges incident on this vertex. However, we never use this value to compute other intermediate values. Hence, the initial rounding error is also our final approximation ratio.

Things become more complex if we want to start with a DP that does more with the numerical values it involves. Let us therefore take another instructive example. Recall the Defective Coloring problem of Section 3.2.3. We are given a graph G and two integers c, Δ^* and are asked if we can partition $V(G)$ into at most c sets, each of which induces a graph of maximum degree at most Δ^*. We mentioned that this problem can be solved in $(c\Delta^*)^{O(\mathrm{tw})} n^{O(1)}$ and that $c < \mathrm{tw}$ in all non-trivial instances, so the question is how to deal with large values of Δ^*.

The algorithm proposed in [18] starts with the same basic idea as before: we should store for each vertex, instead of its exact degree on the current partial solution, the closest integer power of $(1+\epsilon)$. However, we have a problem. Suppose that for some partial solution, we have stored that the degree of a vertex v of the current bag is d, and we introduce a neighbor of v assigned the same color as v. We would therefore need to update the degree of v to $d+1$. In an exact algorithm, this is not a problem; however, if instead of d we have stored the value $\hat{d} = (1+\epsilon)^{\lceil \log_{(1+\epsilon)} d \rceil}$, then the value $\hat{d}+1$ is (probably) not an integer power of $(1+\epsilon)$. If we round up this value to the next integer power of $(1+\epsilon)$, our approximation ratio becomes $(1+\epsilon)^2$. Repeating this operation leads to a sequence of ever-increasing errors, preventing us from obtaining any approximation ratio at all. Of course, the same situation applies in a Join node, where v may have degree d_1 on one side and d_2 on the other (which we approximate with $\hat{d}_1$ and $\hat{d}_2$). The sum of the two approximate values $(\hat{d}_1 + \hat{d}_2)$ is not necessarily an integer power of $(1+\epsilon)$, so we need to either round up (increasing our error), or use a different set of values in our DP.

The way out of this conundrum is through two observations: first, instead of storing integer powers of $(1+\epsilon)$, we can define a smaller value δ (depending on ϵ) and store integer values of $(1+\delta)$. Concretely, we will set δ to be approximately equal to $\frac{\epsilon}{\log n}$. The advantage of this approach is that our algorithm can tolerate a few errors piling up: indeed, even if we approximate a value d by $\hat{d} = (1+\delta)^{\log n} d$ we could still have $\hat{d} \leq (1+\epsilon)d$, by manipulating the constants in the definition of δ appropriately[2]. This idea is a good first step, because it maintains the running time of our algorithm, which would now be around $\left(\log_{(1+\delta)} n\right)^{O(\mathrm{tw})} = \left(\frac{\log n}{\epsilon}\right)^{O(\mathrm{tw})}$ [3]. Observe, however, that we cannot push this idea too much further: setting $\delta = \frac{\epsilon}{n}$ would make our algorithm's approximation much more robust, but then the running time would become XP. It seems that the minimum reasonable value we can give to δ only allows us to accept a logarithmic number of accumulated rounding errors.

[2]Here we are using the approximation $(1+\frac{\epsilon}{\log n})^{\log n} \approx 1+\epsilon$.

[3]Here we are again using the approximation $\ln(1+\delta) \approx \delta$

What we need to do then is make sure that no more than $\log n$ errors of order $(1+\delta)$ can pile up on any value. This can be achieved thanks to our second observation: if we execute our (approximate) DP in the obvious way and then always round up intermediate results, a little thought reveals that we can bound the largest accumulated error by $(1+\delta)^h$, where h is the height of the tree decomposition, because intuitively, every intermediate value is calculated using values calculated at a bag of lower height. We therefore need to make sure that we are working with a tree decomposition of logarithmic height. Thankfully, this is always possible! A theorem of Bodlaender and Hagerup states that, given any tree decomposition, one can in polynomial time produce another decomposition of the same graph with logarithmic height, while only increasing the width by a constant factor [29].

We are therefore done! The main ingredients of our technique are: (i) balance the given tree decomposition so that its height is at most $h = O(\log n)$, using the algorithm of [29] (ii) execute the DP, replacing every value with the closest integer power of $(1+\delta)$, rounding up when necessary, where $\delta = \frac{\epsilon}{2h} = \Omega\left(\frac{\epsilon}{\log n}\right)$.

When will this work? In the following section we review some characteristic applications of this technique, but to give an intuitive feeling of its applicability we observe that this idea is guaranteed to work under the following conditions: suppose we try to execute our DP algorithm in the way we described and at some point we have calculated two values $\hat{d}_1, \hat{d}_2$ which are approximations for the values d_1, d_2 calculated by the exact dynamic programming algorithm; furthermore, suppose that $\hat{d}_1 \in [d_1, \rho_1 d_1]$ and $\hat{d}_2 \in [d_2, \rho_2 d_2]$, that is, the approximation ratios for the two values are ρ_1, ρ_2. Then, the condition we need to satisfy is that if we want to produce some new value $\hat{d}_3$ out of $\hat{d}_1, \hat{d}_2$, we should be able to guarantee that the new approximation error should be at most $(1+\delta)\max\{\rho_1, \rho_2\}$. In other words, the basic condition that the DP must satisfy is that each operation accumulates a new error of at most $(1+\delta)$.

The most notable case where this property *is* indeed satisfied is when the arithmetic operations we perform are additions of non-negative numbers. On the other hand, this condition is *not* satisfied for subtractions. Therefore, dynamic programs which only use additions and comparisons are generally amenable to this technique. This is exactly the case of the program for Defective Coloring, because the degree of each vertex only increases as the dynamic programming algorithm progresses through the graph.

5.1.2 Overview of Applications

The technique sketched in the previous section, which was introduced in [129] has been applied to several natural problems which are W[1]-hard parameterized by treewidth. In this section we review a few such examples to give the reader an idea of the kind of result we can expect from this technique. As we explained, the common aspect of all these problems is that the natural exact DP algorithm runs in XP time because it has to store a large number for each vertex in a bag; and the manipulations this algorithm performs with these numbers are mostly additions and comparisons. The general technique can, in most cases, be easily applied to problems that satisfy these criteria.

One phenomenon that we will repeatedly notice is that this technique often leads to so-called bi-criteria approximations, that is, approximation algorithms which approximate something other than the natural objective function. To see what we mean by this, consider the example of Defective Coloring, for which we presented in the previous section an algorithm that approximates Δ^*. Arguably, one could say that the natural objective in this problem is to minimize c, given a fixed Δ^*, rather than the other way around. However, our algorithm is only able to produce solutions which "almost" satisfy the degree bound, and not solutions which respect Δ^* exactly at the cost of a few extra colors. In a similar way, most of the algorithms of this section may be considered to approximate a secondary objective of each problem. A reason that this happens is that many of the problems we consider are of the form "Make a selection (of a set of vertices/a coloring/etc.) so that a constraint is satisfied in each vertex", where the constraint is something that naturally gives n states to each vertex in the worst case (for example, the degree the vertex

has in its color class). The hardness of solving the problem exactly typically comes from the fact that we have to consider all states of each vertex of a bag. Therefore, the approximation tactic we adopt naturally ends up only *approximately* satisfying the constraints of the problem. As a result, very often our algorithms approximate the problem by slightly bending the rules to accept some non-feasible solutions, and hence can be considered as bi-criteria approximation algorithms.

It is a very interesting question whether such algorithms can be tranformed into algorithms which approximate the natural objective function, while satisfying all hard constraints exactly. In the next section we give evidence that this is not the case, showing that for Defective Coloring approximating c is genuinely harder than approximating Δ^*.

Capacitated Covering. We begin with an example from [129]. In the Capacitated Dominating Set problem, we are given a graph where each vertex has a capacity. We need to select a minimum-size dominating set S with the extra restriction that each vertex $u \in V \setminus S$ is assigned to a neighbor in $N(u) \cap S$ (its dominator) so that no vertex of S has more vertices assigned to it than its capacity. A similar problem is Capacitated Vertex Cover, where each selected vertex can cover some of its incident edges, but not more than its capacity. Both problems are W[1]-hard by pathwidth [58] – in fact these are among the first problems discovered to be hard by treewidth. Nevertheless, both problems admit an easy XP algorithm parameterized by treewidth: we perform dynamic programming, and for each selected vertex we also remember how much of its capacity has already been used in the current partial solution. If the maximum capacity is C, this runs in time $C^{O(\mathrm{tw})}n^{O(1)}$, which is polynomial for constant treewidth, but not FPT, since C could go up to n.

Observe that in the previous paragraph we stated that the easy DP algorithm for each problem remembers for each selected vertex how much of its capacity has already been used. One could of course formulate this, in a completely equivalent way, as saying that we will remember for each vertex how much of its capacity *remains* available. Even though the two formulations are equivalent as far as the exact algorithm is concerned, they are *not* the same if we want to apply our approximation technique. Indeed, if we remember for each vertex how much of its capacity we have used, the natural DP will use *additions*: for example, in every Join bag, we add for each selected vertex, the amount of capacity it is using on each side in order to calculate the total capacity it is using in the whole sub-tree. Thanks to this observation, it is now straightforward to apply the ideas of the previous section to this problem: we only store used capacities that are integer powers of $(1+\delta)$, for $\delta = \Theta\left(\frac{\epsilon}{\log n}\right)$ and if the maximum capacity if $C = O(n)$, then we obtain a DP with size $\left((\log n/\epsilon)^{O(\mathrm{tw})}\right)$. Of course, the same approach would not have worked so easily if we started with the DP that performs subtractions.

What is the result we obtain in this way? We can guarantee that if we are given an instance where the minimum Capacitated Dominating Set (or Vertex Cover) has size k, then our algorithm with produce a solution that also has size at most k, but may violate some of the capacities by a factor $(1+\epsilon)$, for some arbitrarily small value $\epsilon > 0$ given in the input. Hence, we witness the phenomenon we previously described, where because what we are approximating is part of the constraints of the problem, the result we obtain in the end is a bi-criteria approximation algorithm, whose solution is as good as the optimal, but slightly infeasible. It is a natural question if this can be transformed into an FPT approximation scheme that produces a truly feasible solution of size $(1+\epsilon)k$.

Distance Domination and Scattered Set. Let us now revisit the (k,r)-Center and d-Scattered Set problems, which have served us as running examples throughout this work, presenting some more results from [119, 120]. The reader may recall that we have seen algorithms for these problems running in $(2r+1)^{\mathrm{tw}}n^{O(1)}$ and $(d+1)^{\mathrm{tw}}n^{O(1)}$ respectively. The question we may now ask is whether these algorithms, which have XP running times when r, d are given in the input, can be turned into FPT approximation schemes via our technique. To answer this, we once again have to check if the algorithms can be formulated in such a way that all our arithmetic operations are additions (and comparisons). A little thought reveals that we can indeed formulate

the two algorithms in this way: since for each vertex we use a label that encodes its distance from the closest selected vertex, in order to check if this label is satisfied we check if its value is at least the value of the label of a neighbor of the vertex *plus* the weight of their incident edge.

What kind of algorithm can be obtained in this way? Again, the natural objective function for these problems is the size of the selected set k. Our algorithms are, therefore, FPT approximation schemes which provide a bi-criteria approximation guarantee: given a graph where a (k, r)-Center exists, our algorithm can guarantee to find a solution that selects k vertices such that all other vertices are at a distance at most $(1+\epsilon)r$ from one selected (and similarly for d-Scattered Set).

An interesting aspect of the algorithm for (k, r)-Center sketched above is that it turns out to be useful for approximating also the much more general case of the problem where we parameterized by clique-width. The main idea is a Win/Win argument, similar to that of Section 3.2.4, which may be of further interest for transforming treewidth-parameterized algorithms to clique-width parameterized algorithms. Recally that in Section 3.2.4, the main trick we relied on to deal with the parameterization of Edge Hamiltonicity by clique-width was a theorem of Gurski and Wanke [102] stating that if we eliminate all large bi-cliques from a graph with small clique-width, the graph will have small treewidth. The trick for (k, r)-Center parameterized by clique-width is then the following: we take the clique-width expression, locate every large Join operation, and replace every such operation with an appropriate small gadget which maintains all distances constant but removes the large bipartite graph. Repeating this, we end up with a graph where we can bound the treewidth, so we obtain an FPT approximation scheme parameterized by clique-width simply by invoking our algorithm for the parameter treewidth. Hence, this is another nice example of an algorithmic application of the structural result of Gurski and Wanke separating treewidth and clique-width.

Minimum Stable Cut. For our final example of this section, we revisit the Min Stable Cut problem of [131]. Recall that in this problem we are given a (possible edge-weighted) graph and are asked to find a cut of minimum weight such that every vertex has the majority of its incident weight connecting it to the other side. As we mentioned in section 4.1.5, if the maximum weight of the given instance is W, this problem can be solved in time $(nW)^{O(\mathrm{tw})}$.

What can we expect if we try to apply our technique to this problem? First of all, examining the dynamic program we observe that it does seem amenable to our approximation trick: for each vertex in a bag we recall the weight of edges connecting it to the other side; and when we encounter a neighbor of a vertex in a bag that is placed on the other side, we increase this value appropriately (so we perform an addition). When forgetting a vertex, we just make sure that the value we remember for it in the current solution is at least one half of its total incident weight. Since we only use additions and comparisons, we obtain an algorithm which can guarantee to produce a cut with the following property: if there is a stable cut of weight k, our algorithm will produce a cut with weight at most k where every vertex has at least a $\frac{1-\epsilon}{2}$-fraction of its incident weight leading to the other side. Observe that once again the guarantee is of a bi-criteria approximation: because we only remember the weight connecting to a vertex to the other side approximately, we can only guarantee that the produced solution is "almost" stable. Conversely, its weight is as good as the optimal stable solution (or better).

There is, however, one complication with the ideas above. As with Power Vertex Cover, there is no immediate relation between W and n. Indeed, since W can be written in binary it could be exponential in n. Hence, blindly applying our technique will not result in an FPT algorithm, as in this case we would have a table of size $(\log W)^{\mathrm{tw}}$ which would become n^{tw}. As with Power Vertex Cover, we therefore need to pre-process the instance so that weights become polynomially bounded. This is achieved by dividing edge weights by an appropriate number, with the notable complication that this number has to be different for each vertex. The problem here is that an edge uv could simultaneously be one of the heaviest edges incident on u and one of the lightest edges incident on v, hence it is not immediately clear how to manipulate its weight in a way that approximately preserves the stability information for both vertices. The solution proposed in [131] is to reduce this to a more general (directed) version of the problem, where we

essentially replace the edge uv with two anti-parallel arcs with distinct weights. Even though this version of the problem is more general, the DP of the previous paragraph still works with minor modifications and we are able to obtain an FPT approximation scheme, as expected.

5.1.3 An Example of a Negative Result

Taking another look at the examples of the previous section we see again that the technique we use tends to produce bi-criteria approximations: we obtain a solution with size as good as the optimal, but which may slightly violate some capacity constraints for Capacitated Dominating Set; in which some vertices may be slightly further than distance r from the selected set in (k, r)-Center; or in which some vertices may be "almost" stable in Min Stable Cut. In all such situations, it is a natural question if this algorithm can be transformed into an algorithm that produces a feasible solution (satisfying all the constraints of the problem) while approximating the natural objective function.

In this section we give a partial answer to this question regarding Defective Coloring (taken from [18]). Recall that the algorithm we gave produces a coloring with c colors and maximum degree $(1+\epsilon)\Delta^*$, that is, it approximates Δ^* while using exactly the given number of colors. The question for this problem is then what is the minimum number of colors we can guarantee to use in a solution that respects the degree bound Δ^* exactly and can be produced in FPT time? In other words, what is the best approximation ratio achievable in FPT time (for parameter treewidth) if our objective is to minimize the number of colors?

On the positive side, we can observe that the algorithm we have can in fact be useful for approximating c[4]. In particular, we can do the following: given values c, Δ^* we can run our approximation algorithm for, say, $\epsilon = 1/2$. If the input graph is (c, Δ^*)-colorable, the algorithm is then guaranteed to produce a $(c, 3\Delta^*/2)$-coloring. We now consider the graph induced by each color class, which has maximum degree at most $3\Delta^*/2$ and notice that there always exists a 2-coloring of such a graph with maximum degree at most $3\Delta^*/4 \leq \Delta^*$ in each class (simply take any stable cut of the given graph). We can therefore obtain a $(2c, \Delta^*)$-coloring of the input graph, hence a 2-approximation to the number of colors. Notice that this type of approximation guarantee is non-trivial, as Coloring (which is the case $\Delta^* = 0$) is known to be a completely inapproximable problem in polynomial time [73].

Since Coloring is a problem that is generally very hard to approximate and we are able to obtain a 2-approximation on the number of colors even for large values of Δ^*, we might consider the above result quite positive. However, the question still remains whether we can do as well approximating c as we can approximating Δ^*. The answer is, unfortunately, no (under standard complexity assumptions). One way to justify this is to observe that the reduction given in [18] proving that the problem is W[1]-hard parameterized by treewidth works for $c = 2$. Hence, if there was an (FPT) approximation algorithm with ratio better than $3/2$ with respect to c, we could execute this algorithm on the instances produce by the reduction, and the algorithm should be able to 2-color all positive instances, solving a W[1]-hard problem in FPT time. As a result, no FPT approximation can achieve a ratio better than $3/2$ for c (for parameter treewidth).

Although the hardness argument above is correct, it is not completely convincing, because it only happens to cover one case ($c = 2$). One could still reasonably hope that a better than $3/2$ approximation is achievable for larger values of c[5]. Unfortunately, this type of result can also be ruled out if we parameterize by treewidth: there is a parametric reduction which, given an integer i, starts from k-Clique and produces a graph G of treewidth $O(k)$ and an integer Δ^* such that (i) if the original instance had a k-clique, G can be $(2i, \Delta^*)$-colored (ii) otherwise, G cannot be $(3i-1, \Delta^*)$-colored. Hence, the bound on $3/2$ approximation ratio is not an artifact of the hardness for $c = 2$. Rather, it is a natural barrier that manifests itself for all numbers of colors.

[4] This is in general not obvious for the other problems.

[5] Indeed, one of the results of [18] is that if we parameterize by feedback vertex set, the problem admits an FPT approximation with an *additive* error of $+1$, as the only hard case is $c = 2$.

5.2 Win/Win for Max-SAT Parameterized by Clique-width

We now switch gears and present an FPT approximation scheme (which appeared in [55]) that uses a completely different technique from that of Section 5.1. Our problem is Max-SAT, the maximization version of SAT, where we are looking for an assignment that satisfies as many clauses as possible, and the hardness we will try to overcome is the fact that SAT is W[1]-hard already parameterized by neighborhood diversity (as we saw in Section 3.3.1). This hardness is a significant obstacle for dealing with SAT on dense instances, and indeed explains why this problem, which is FPT parameterized by treewidth, becomes intractable parameterized by clique-width.

The key idea behind our algorithm will be a Win/Win strategy that attempts to build on the tractability of SAT and Max-SAT parameterized by treewidth and extend this to clique-width. Along the way, we will show a third example of how the structural result of Gurski and Wanke, stating that graphs of small clique-width without large bicliques actually have small treewidth, can be invaluable in bridging the algorithmic gap between the two parameters[6].

Without going into all the technical details, we sketch the main ideas of the algorithm. We are given in the input a SAT formula and a parameter ϵ such that we would like to satisfy at least a $(1-\epsilon)$-fraction of the number of clauses satisfied by the best assignment. We now make the following basic observations:

1. If we realize that the formula only contains "long" clauses, the problem is easy. Indeed, if all clauses have length at least k, then setting all variables at random satisfies a $(1-2^{-k})$-fraction of all clauses, so if k is large enough as a function of ϵ, we are done.

2. If, on the other extreme, we realize that the formula only contains "short" clauses, the problem is also easy. In this case, the maximum degree of one side of the incidence graph is small, therefore by the theorem of Gurski and Wanke the incidence graph actually has small treewidth. We can therefore solve the problem exactly in FPT time.

The two observations above encapsulate the two basic ingredients of our algorithm: if clauses are large and the instance is very dense, a random assignment is an easy but excellent approximation; if on the other hand the instance is sparse, we are actually dealing with the easier case of treewidth. The rest of the work we need to do then is to figure out how to handle the general situation where the input instance does not easily fall into one of these two extreme cases.

Our strategy is now the following. We define two numbers ℓ, L and partition the clauses into three categories: (i) short clauses are those that have size at most ℓ (ii) long clauses are those that have size at least L (iii) medium clauses are all remaining clauses. We will make sure that L is much larger that ℓ, say $L = \ell/\epsilon^{10}$ so that long clauses are significantly larger than short clauses. Furthermore, we choose ℓ appropriately so that the number of medium clauses is at most $\epsilon^2 m$. This is always possible and guarantees that if we delete all the medium clauses from our instance, we will not significantly affect the value of the optimal. Hence, we may assume that our instance has only short and long clauses.

Given now an instance with only short and long clauses, we observe that if one of the two categories contains very few clauses (say, fewer than $\epsilon^2 m$) then we can simply remove them and only have clauses of the other category. This corresponds to one of the easy cases we identified above. Hence, the interesting case is when we have a non-trivial number of both short and long clauses.

The intuition now is that we would like to handle the long clauses randomly (they are long enough that this will satisfy almost all of them) and the short clauses via treewidth. The problem is that setting at random a variable that appears in a long clause could have a significant negative effect on the part of the instance made up of short clauses. We therefore need to find variables which are "safe" in the sense that they allow us to make progress in the long clauses without damaging too many short clauses.

[6]The previous two use cases of this structural result were in Sections 3.2.4 and 5.1.2.

The key idea of the algorithm of [55] is now a simple counting argument which locates such a variable. In particular, we use the fact that long clauses are much longer than short clauses but not too few, hence by counting occurrences it follows that some variable must appear almost exclusively in long clauses. We set this variable at random and repeat until we have set at least $1/\epsilon^2$ variables of each long clause at random. Then, we forget all the long clauses (if any remain) and solve the rest of the instance using treewidth. The selection of the variables ensures that we have not damaged the optimal of the low-treewidth part of the instance too much, while we have set enough variables of each long clause at random to have a very high probability of satisfying each such clause.

What have we learned? The approximation scheme for Max-SAT parameterized by clique-width is a nice way to leverage a purely structural result (concerning the relationship between treewidth and clique-width) into an approximation algorithm. It would be very interesting to see if there are other situations where something like this may be done, as Win/Win-type arguments are an excellent but currently underutilised tool in parameterized approximation. This is of particular interest because there seem to be quite a few problems which become intractable in the transition from treewidth to clique-width.

Another interesting aspect of the algorithm we presented is that it is structure-agnostic, in the sense that it can function *without* having access to the clique-width expression of the input incidence graph. Indeed, clique-width is only used as part of the analysis, and the rest can be executed either in polynomial time or using algorithms that find tree decompositions. This is quite positive because, as we have mentioned, computing clique-width is a major open problem, so assuming that a clique-width expression is given in the input is a major weakness for most algorithms parameterized by clique-width.

5.3 Almost Dense Max-CSPs

In this final section we present some results from [89] which combine the notion of input structure with approximation in a way that is somewhat different from what we have done in the rest of this work. The main problem we will be interested in is Max Cut: we are given a graph G and want to partition $V(G)$ into two parts in a way that maximizes the number of edges that have endpoints on both parts.

The starting point of our investigation is the work of Arora, Karger, and Karpinski [6], as well as Fernandez de la Vega [78] which established that Max Cut admits a PTAS on dense graphs, that is, graphs of average degree $\Delta = \Omega(n)$, or equivalently, graphs with $\Omega(n^2)$ edges. In this section we will therefore consider this case as our island of tractability and will measure the structure of the input instance as the distance from being dense. The simplest way to do this is to consider the complexity of the problem as a function of the average degree Δ of the input graph. Even though this takes us a bit outside the parameterized complexity framework, this approach retains the main spirit of this work, which is that structural measures of the input must be taken into account in our analysis.

The basic question we would like to answer is the following: how easy is it to obtain an arbitrarily good approximation (that is, an approximation scheme) for Max Cut on graphs of average degree Δ? Observe that approximation is necessary here, as restricting only the average degree of an instance is not sufficient to make Max Cut tractable to solve exactly[7]. Furthermore, because Max Cut is APX-hard, the existence of almost-linear PCPs [148] implies the following: if the ETH is true, then there exists $r < 1$ such that for all $\epsilon > 0$ no algorithm can r-approximate Max Cut on bounded degree graphs in time $2^{n^{1-\epsilon}}$. This implies that, wherever we may manage to place Δ in the performance of our approximation scheme, when Δ is a constant the performance of our algorithm will need to degrade to exponential (in n).

[7] Max Cut is NP-complete on bounded-degree graphs, and we can add an appropriately-sized clique to any such instance to obtain any desired average density.

Taking this previous work into account, what we are looking at is the following situation: the best we can hope for if we want to $(1-\epsilon)$-approximate Max Cut is an algorithm running in time $2^{f(n,\Delta)}$, where the function $f(n,\Delta)$ becomes constant (or at most logarithmic) when $\Delta = \Theta(n)$, but degenerates to $\Theta(n)$ when $\Delta = O(1)$. This would agree with both the algorithms of [6, 78] and the hardness results implied by efficient PCPs and the ETH. Given these constraints, the most natural guess is probably something like $f(n,\Delta) = n/\Delta$, that is, that approximating Max Cut arbitrarily well takes time exponential in n/Δ.

The main result of [89] is to confirm this guess and extend it to constraint satisfaction problems of higher arity. On the algorithmic side, Fernandez de la Vega and Karpinski [79] already showed that for Max Cut specifically, there does exist an approximation scheme running in time exponential in n/Δ, confirming the algorithmic side of our guess. The basic idea is rather simple: just like the algorithm of [78] selects a sample from the input, guesses its optimal solution, and then greedily sets the rest of the graph according to the sample, the algorithm of [79] does the same but while selecting a sample of size roughly n/Δ. Hence, the running time is dominated by the fact that we consider all possible solutions for our sample. The first contribution of [89] was to show that, assuming the ETH, this is optimal: no algorithm can $(1-\epsilon)$-approximate Max Cut in time significantly faster than $2^{n/\Delta}$. As a result, not only is the size of the sample selected by the algorithm of [79] correct, but no algorithm better than sampling exists.

The above results give a complete understanding of the trade-off between density and time needed to approximate Max Cut: the problem gradually degrades from polynomial-time solvable for dense instances to unsolvable in less than exponential time for sparse instances, and for each particular value of Δ we have a very precise idea of its complexity. Motivated by this encouraging result, [89] then goes on to study the much more general Max-k-SAT problem, for fixed k, taking into account that this more general problem also admits a PTAS on dense instances [6], where now dense instances are those with $\Theta(n^k)$ constraints. In this context the contribution is two-fold: on the positive side, it was shown that a generalization of the algorithm of [79] is possible. However, the generalization only works in the density area between n^k and n^{k-1}. More precisely, an instance with $n^{k-1+\delta}$ constraints can be approximated arbitrarily well in time exponential in $n^{1-\delta}$. For example, for Max-3-SAT these results imply that instances with $n^{2.3}$ clauses can be $(1-\epsilon)$-approximated in time roughly $2^{n^{0.7}}$. More broadly, the message of this result is that the time-density trade-off we have for Max Cut can also be found in more general problems of higher arity $k \geq 3$, but only in the area of instances with between n^{k-1} and n^k constraints. Even though this seems a bit arbitrary, it is not an artifact of the techniques of [89]: the same work showed this behavior is optimal under the ETH! In other words, all such problems are inapproximable in sub-exponential time if the number of clauses is less than n^{k-1}, and then their complexity gradually improves in the same way as Max Cut as we increase the number of clauses from n^{k-1} to n^k. This was somewhat surprising, as it disproved a conjecture of [79] that the algorithm for Max Cut should be generalizable to cover all semi-dense instances of problems of higher arity.

Bottom line. Even though the results of [89] are in an area adjacent to parameterized complexity, their point of view is the same: we are trying to understand the complexity of a problem (in this case an approximation problem) as a function not only of n, but also a second parameter that measures the structure of the input (in this case Δ). Using the tools of fine-grained complexity theory (in particular, the ETH) and efficient PCPs, we have managed to map out precisely the time complexity of Max Cut and its generalizations as a function of density: for Max Cut the correct answer is "exponential in n/Δ" and for problems with larger arity k the same behavior is confined to the region of instances containing between n^{k-1} and n^k constraints. This gives a clear picture of the trade-off between density and time complexity.

Even though we are happy to have matching upper and lower bounds, it seems that measuring density simply by counting the number of edges/constraints of the input instance is a rather crude tool. An interesting research direction would therefore be to investigate whether other more sophisticated (and algorithmically useful) ways exist to generalize the island of tractability given by the PTAS for Max Cut on dense graphs.

Chapter 6

Conclusions – Research Directions

In this chapter we give a brief high-level discussion of the results we presented in this work and their general scientific context. Our focus is first in identifying the general directions we believe have the most potential to lead to further progress. We then close our presentation with a list of concrete open problems (of varying difficulty) related with the results we have seen in this work.

6.1 Big Picture Overview

The question of understanding and dealing with the NP-completeness phenomenon has long been considered to be at the heart of theoretical computer science. In this work we have presented several results in this thematic area but in a way that has attempted to hammer home a specific take-home message at every opportunity. The take-home message was simply that input structure, as measured by appropriate graph parameters, is a variable that needs to be taken into account if we hope to truly understand what is going on with NP-hard graph problems. Beyond this basic message, which is of course not particular to this work but underlies a big part of modern parameterized complexity theory, what else have we learned?

We love treewidth! Once we recognize our need for structural parameters, the obvious next question is how are we supposed to measure the degree of structure of a graph. If parameterized complexity theory has one answer to this question, that answer is treewidth. As we have seen in this work, treewidth is a huge player in this line of research and in almost all situations it is the first structural graph parameter our community considers when dealing with a new problem. One may be tempted to believe that the reason for this is a combination of inertia (we have long been studying treewidth, so we continue) and satisfaction at positive results (treewidth helps us solve many problems). However, one message of this work is that the fine-grained complexity results of the last decade (some of which we presented here) indicate that the reasons for studying treewidth and treewidth-like parameters run much deeper. In particular, the fact that (assuming the ETH/SETH) we have exact tight bounds on the complexity of many (perhaps most) natural problems for this parameter, can be read as an indication that measuring the treewidth measures a graph property of great algorithmic importance. Despite the fact that at some point treewidth was perhaps considered well-understood in the parameterized complexity theory community, more recent work has shown that much is left to be done, especially with regards to the relationship among treewidth-like parameters (chiefly pathwidth, tree-depth, and clique-width).

Because of all the above, it seems safe to bet that treewidth will remain a central topic of study in this research area in the years to come. It is a means of quantifying graph structure which at the same time (i) is understandable and understood to the point where we are able to produce fruitful results (ii) seems to capture something profound complexity-wise as the complexity of most problems really does seem to depend on treewidth in interesting ways (iii) gives rise to a rich eco-system of related parameters that allow us to better understand why/how the tree-like

structure impacts the complexity of various problems. As a result, parameterized complexity questions around treewidth are likely to remain a theme of great interest.

We have to move away from treewidth. Despite all the positive aspects of treewidth we evoked in the previous paragraph, we have to admit that the obsessive focus on treewidth and its close relatives is a weakness of the current state of the art in this area. Even though treewidth is indeed an excellent way to measure graph structure, there must exist other fruitful ways to measure structure in a sense that is algorithmically meaningful. In this work we have talked about one attempt to advance in this direction by talking about modular width. However, the point is not that modular width is necessarily the next treewidth (though it can be argued that modular width is a measure that deserves further study). Rather, the point is that this line of research on the interplay between graph structure and complexity will only realize its full potential when it manages to incorporate as many algorithmically interesting measures of structure as possible. We therefore advocate for the study of parameters which attempt to measure algorithmically interesting properties other than "tree-likeness". Even though it is unlikely that we will be able to discover any measure that rivals the algorithmic efficacy of treewidth, it is still likely that there exist large families of graphs where large families of problems are tractable for reasons other than the low connectivity of the graph (indeed, the positive algorithmic properties of clique-width are an indication in this direction). Discovering such families of graphs is therefore a promising research frontier.

Other Frontiers. Finally, in addition to the open-ended quest for more parameters mentioned above, let us talk about two other open-ended research frontiers with a close connection to the results presented in this work. First, we recall a research frontier we already discussed in Section 4.1.5, where our focus was to study Min Stable Cut as a function of tw and Δ. This was a situation where we were naturally led to study a problem with two structural parameters, and it is worth noting that this particular pair of parameters is likely to be of interest to many problems. The traditional parameterized complexity approach to handle this is to simply declare that the parameter is $\mathrm{tw} + \Delta$ and then deal with the problem as if it had a single parameter. This is fine if all we care about is the distinction between FPT/non-FPT, but is a very unsatisfactory solution if we are aiming for more precise, fine-grained results. As we explained in Section 4.1.5 the tools to convincingly handle this type of multi-parameter problem are not yet well-developed, so we expect that the study of the fine-grained complexity of such multiple parameterizations to become an interesting frontier.

Second, even though we devoted a whole chapter to the interplay between approximation and parameterized complexity, this research area is still rather under-developed, especially with respect to structural parameters. The generic approximation technique given in Section 5.1 is a nice start, but treewidth (and related parameters) provides so much algorithmic structure that it's hard to believe that we cannot further exploit it in the context of approximation. We are therefore led to believe that the development of FPT approximation algorithms is likely to be of much interest in the years to come, with algorithms that attempt to "beat" fine-grained lower bounds even for FPT problems appearing as a prime first target (we give some concrete open problems along these lines in the next section).

6.2 Some Concrete Open Problems

We present a few open problems arising from the results we have presented, following roughly the same thematic structure as this work: we start with "price of generality" questions, distinguishing which problems are FPT for which parameters; continue with more fine-grained questions; and close this chapter with some questions related to FPT approximation.

1. Looking at our parameter map of Figure 2.1, probably the least well-understood border is that between modular width and neighborhood diversity. At the moment, natural problems

which are FPT for neighborhood diversity and W[1]-hard for modular width are very rare. One recent such example was given in [44], where it was shown that Equitable Coloring is FPT for neighborhood diversity but W[1]-hard for a parameter that is more restricted than modular width (and hence for modular width as well). It would be interesting to identify more such problems. Note that, in principle, this should not be expected to be too hard (neighborhood diversity is an extremely restricted parameter), but modular width is a measure that is hard to keep under control when designing a reduction.

2. In Section 3.2.1 we presented a problem (Grundy Coloring) that is FPT for pathwidth but W[1]-hard for treewidth. Unfortunately, this specimen is not entirely satisfactory, as the algorithm we used to prove tractability for pathwidth is not different from the treewidth-based algorithm – tractability follows from an unrelated bound on the value of the optimal. Are there other natural problems that separate these parameters? In particular, is there a natural problem where pathwidth allows us to develop an algorithm that would not work on tree decompositions (for example because handling Join nodes is hard)?

3. For another concrete open problem, in Section 3.2.1 we left unanswered the question whether Grundy Coloring is FPT parameterized by feedback vertex set.

4. The relation between treewidth and pathwidth is mirrored in the realm of dense graphs by the relation between clique-width and *linear* clique-width. In other words, linear clique-width is the "path-like" version of clique-width, where Union operations are forced to involve one graph containing a single vertex. Linear clique-width is a much less well-studied parameter, but to the extent that anything is known, it does not seem to make problems much easier than clique-width, despite being much more restricted. Does there exist a natural problem that is FPT for linear clique-width but W[1]-hard for clique-width?

5. All the parameters we have defined for sparse graphs can also be transformed into dense graph parameters if we look at them as parameters on the complement of the input graph. In this sense, one may define co-treewidth as the treewidth of the complement of the input graph. This was recently pointed out in [61] as an interesting direction, although previous results along these lines already existed (e.g. the FPT algorithm for Hamiltonicity parameterized by co-chromatic number in [133], the FPT algorithm for Graph Motif for the same parameter [36], and the FPT algorithm for SAT parameterized by the co-treewidth of the conflict graph in [99], to name a few). Understanding the properties of co-treewidth (and co-degeneracy, and co-chromatic number) and finding which problems are FPT for these parameters may be an interesting direction.

6. Returning to Grundy Coloring, the current best algorithms run in $n^{O(\mathrm{tw}^2)}$ and $2^{O(\mathrm{pw}^2)}n^{O(1)}$. Are these bounds optimal under the ETH? In particular, can the problem be solved in $2^{o(\mathrm{pw})^2}n^{O(1)}$. We suspect the answer is No, meaning that the known algorithm is also optimal in the case of pathwidth.

7. Vertex cover is a parameter so restricted that most problems which are W[1]-hard by treewidth become FPT by vertex cover. Nevertheless, the complexity of the easy algorithm for this parameter sometimes leaves something to be desired. For example, for Digraph Coloring, which we mentioned in Section 4.1.3 cannot be solved in $\mathrm{td}^{o(\mathrm{td})}n^{O(1)}$, the best algorithm for the parameter vertex cover runs in $\mathrm{vc}^{O(\mathrm{vc})}n^{O(1)}$. Can this be improved to $2^{O(\mathrm{vc})}n^{O(1)}$ or is this impossible under the ETH?

8. Is there a natural problem which admits a $c^{\mathrm{tw}}n^{O(1)}$ algorithm for which the base of the exponent decreases when we parameterized by pathwidth? Currently, for almost all problems for which precise bounds are known the complexity of the algorithm for treewidth matches the lower bound for pathwidth. The only exception is Hamiltonicity, for which the best algorithms run with dependence $(2+\sqrt{2})^{\mathrm{pw}}$ and 4^{tw}, and only the pathwidth bound is known to be optimal [50]. Does this mean that the treewidth algorithm can be improved or that

this problem has different complexity for these parameters? Is there any natural problem that does have different complexity for these parameters in this sense?

9. For a concrete problem in FPT approximation, in Section 5.1.3 we explained that there is an FPT 2-approximation (by treewidth) for the minimum number of colors needed to color a graph so that each color class has maximum degree below a given bound. We also explained why this approximation ratio cannot be below 3/2. Can the gap between 2 and 3/2 be closed?

10. Fine-grained lower bounds, such as those of Section 4.2 can be seen as invitations for FPT approximation algorithms. Let us give a concrete example: according to known results, it is impossible to 3-color a 3-colorable graph in time $(3-\epsilon)^{\mathrm{tw}}n^{O(1)}$ (under the SETH). However, it is possible to 6-color such a graph, as follows: using a simple algorithm of Czumaj et al. [53] we can, given a tree decomposition of width tw of a graph G and an integer r, partition $V(G)$ into r sets, each of which induces a graph of treewidth tw/r. We can therefore, for example, set $r = 2$, and then run the 3-Coloring algorithm on each part. If we use disjoint sets of colors, this 6-colors the graph in time $3^{\mathrm{tw}/2}n^{O(1)} \approx 1.7^{\mathrm{tw}}n^{O(1)}$, so significantly faster than the time needed to 3-color the graph[1]. This situation gives rise to several interesting questions:

    (a) What is the minimum c such that we can 5-color a 3-colorable graph in time c^{tw}?

    (b) Is there an algorithm that produces a 2-approximation algorithm for Coloring faster than that of [53]? For instance, 5-colorable graphs can be 10-colored using this argument in time $(\sqrt{5})^{\mathrm{tw}}n^{O(1)}$. Can we do this in $(2.1)^{\mathrm{tw}}n^{O(1)}$ or prove that it is impossible under the SETH?

    (c) The same trick can be used for other problems as well: if a graph contains an independent set of size k, using this technique we can guarantee to find an independent set of size at least $k/2$ in time $(\sqrt{2})^{\mathrm{tw}}n^{O(1)}$. Is this best possible for a 2-approximation?

[1]This argument was recently rediscovered by Lokshtanov et al. [142] who were seemingly unaware of the algorithm of [53].

Bibliography

[1] Antonis Achilleos, Michael Lampis, and Valia Mitsou. Parameterized modal satisfiability. *Algorithmica*, 64(1):38–55, 2012. doi:10.1007/s00453-011-9552-z.

[2] Eyal Amir. Approximation algorithms for treewidth. *Algorithmica*, 56(4):448–479, 2010.

[3] Eric Angel, Evripidis Bampis, Bruno Escoffier, and Michael Lampis. Parameterized power vertex cover. *Discret. Math. Theor. Comput. Sci.*, 20(2), 2018. URL: http://dmtcs.episciences.org/4873.

[4] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

[5] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

[6] Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. *J. Comput. Syst. Sci.*, 58(1):193–210, 1999. doi:10.1006/jcss.1998.1605.

[7] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.

[8] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.

[9] Ashwin Arulselvan, Ágnes Cseh, Martin Groß, David F. Manlove, and Jannik Matuschke. Matchings with lower quotas: Algorithms and complexity. *Algorithmica*, 80(1):185–208, 2018. doi:10.1007/s00453-016-0252-6.

[10] Haris Aziz, Serge Gaspers, Edward J. Lee, and Kamran Najeebullah. Defender stackelberg game with inverse geodesic length as utility metric. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 694–702. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL: http://dl.acm.org/citation.cfm?id=3237486.

[11] Cristina Bazgan, Ljiljana Brankovic, Katrin Casel, Henning Fernau, Klaus Jansen, Kim-Manuel Klein, Michael Lampis, Mathieu Liedloff, Jérôme Monnot, and Vangelis Th. Paschos. The many facets of upper domination. *Theor. Comput. Sci.*, 717:2–25, 2018. doi:10.1016/j.tcs.2017.05.042.

[12] Rémy Belmonte, Tesshu Hanaka, Masaaki Kanzaki, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Michael Lampis, Hirotaka Ono, and Yota Otachi. Parameterized complexity of (a, ℓ)-path packing. In Leszek Gasieniec, Ralf Klasing, and Tomasz Radzik, editors,

Combinatorial Algorithms - 31st International Workshop, IWOCA 2020, Bordeaux, France, June 8-10, 2020, Proceedings, volume 12126 of *Lecture Notes in Computer Science*, pages 43–55. Springer, 2020. doi:10.1007/978-3-030-48966-3_4.

[13] Rémy Belmonte, Tesshu Hanaka, Ioannis Katsikarelis, Eun Jung Kim, and Michael Lampis. New results on directed edge dominating set. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 67:1–67:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.67.

[14] Rémy Belmonte, Tesshu Hanaka, Ioannis Katsikarelis, Michael Lampis, Hirotaka Ono, and Yota Otachi. Parameterized complexity of safe set. *J. Graph Algorithms Appl.*, 24(3):215–245, 2020. doi:10.7155/jgaa.00528.

[15] Rémy Belmonte, Tesshu Hanaka, Michael Lampis, Hirotaka Ono, and Yota Otachi. Independent set reconfiguration parameterized by modular-width. *Algorithmica*, 82(9):2586–2605, 2020. doi:10.1007/s00453-020-00700-y.

[16] Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 14:1–14:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.14.

[17] Rémy Belmonte, Michael Lampis, and Valia Mitsou. Defective coloring on classes of perfect graphs. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, volume 10520 of *Lecture Notes in Computer Science*, pages 113–126. Springer, 2017. doi:10.1007/978-3-319-68705-6_9.

[18] Rémy Belmonte, Michael Lampis, and Valia Mitsou. Parameterized (approximate) defective coloring. *SIAM J. Discret. Math.*, 34(2):1084–1106, 2020. doi:10.1137/18M1223666.

[19] Oren Ben-Zwi, Danny Hermelin, Daniel Lokshtanov, and Ilan Newman. Treewidth governs the complexity of target set selection. *Discret. Optim.*, 8(1):87–96, 2011. doi:10.1016/j.disopt.2010.09.007.

[20] Matthias Bentert, Klaus Heeger, and Dusan Knop. Length-bounded cuts: Proper interval graphs and structural parameters. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 36:1–36:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ISAAC.2020.36.

[21] Benjamin Bergougnoux, Édouard Bonnet, Nick Brettell, and O-joung Kwon. Close relatives of feedback vertex set without single-exponential algorithms parameterized by treewidth. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.IPEC.2020.3.

[22] Dietmar Berwanger, Anuj Dawar, Paul Hunter, Stephan Kreutzer, and Jan Obdrzálek. The dag-width of directed graphs. *J. Comb. Theory, Ser. B*, 102(4):900–923, 2012.

[23] Nadja Betzler, Robert Bredereck, Rolf Niedermeier, and Johannes Uhlmann. On bounded-degree vertex deletion parameterized by treewidth. *Discret. Appl. Math.*, 160(1-2):53–60, 2012. doi:10.1016/j.dam.2011.08.013.

[24] Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *J. ACM*, 68(3):16:1–16:40, 2021. doi:10.1145/3444942.

[25] Marzio De Biasi and Juho Lauri. On the complexity of restoring corrupted colorings. *J. Comb. Optim.*, 37(4):1150–1169, 2019. doi:10.1007/s10878-018-0342-2.

[26] Hans L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14(1):1–23, 1993. doi:10.1006/jagm.1993.1001.

[27] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[28] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A c^k n 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.

[29] Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM J. Comput.*, 27(6):1725–1746, 1998. doi:10.1137/S0097539795289859.

[30] Hans L. Bodlaender, Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, Yoshio Okamoto, Yota Otachi, and Tom C. van der Zanden. Subgraph isomorphism on graph classes that exclude a substructure. *Algorithmica*, 82(12):3566–3587, 2020. doi:10.1007/s00453-020-00737-z.

[31] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996.

[32] Hans L. Bodlaender, Hirotaka Ono, and Yota Otachi. A faster parameterized algorithm for pseudoforest deletion. *Discret. Appl. Math.*, 236:42–56, 2018. doi:10.1016/j.dam.2017.10.018.

[33] Hans L. Bodlaender, Erik Jan van Leeuwen, Johan M. M. van Rooij, and Martin Vatshelle. Faster algorithms on branch and clique decompositions. In Petr Hlinený and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 2010. doi:10.1007/978-3-642-15155-2_17.

[34] Marthe Bonamy, Lukasz Kowalik, Jesper Nederlof, Michal Pilipczuk, Arkadiusz Socala, and Marcin Wrochna. On directed feedback vertex set parameterized by treewidth. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science - 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2018. doi:10.1007/978-3-030-00256-5_6.

[35] Édouard Bonnet, Nick Brettell, O-joung Kwon, and Dániel Marx. Generalized feedback vertex set problems on bounded-treewidth graphs: Chordality is the key to single-exponential parameterized algorithms. *Algorithmica*, 81(10):3890–3935, 2019. doi:10.1007/s00453-019-00579-4.

[36] Édouard Bonnet and Florian Sikora. The graph motif problem parameterized by the structure of the input graph. *Discret. Appl. Math.*, 231:78–94, 2017. doi:10.1016/j.dam.2016.11.016.

[37] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 8:1–8:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.8.

[38] Hajo Broersma, Petr A. Golovach, and Viresh Patel. Tight complexity bounds for FPT subgraph problems parameterized by the clique-width. *Theor. Comput. Sci.*, 485:69–84, 2013. doi:10.1016/j.tcs.2013.03.008.

[39] Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Boolean-width of graphs. *Theor. Comput. Sci.*, 412(39):5187–5204, 2011. doi:10.1016/j.tcs.2011.05.022.

[40] Hubie Chen. Quantified constraint satisfaction and bounded treewidth. In Ramón López de Mántaras and Lorenza Saitta, editors, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 161–165. IOS Press, 2004.

[41] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.

[42] Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. *SIAM J. Comput.*, 48(2):513–533, 2019. doi:10.1137/17M1127211.

[43] Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971. doi:10.1145/800157.805047.

[44] Gennaro Cordasco, Luisa Gargano, and Adele A. Rescigno. Iterated type partitions. In Leszek Gasieniec, Ralf Klasing, and Tomasz Radzik, editors, *Combinatorial Algorithms - 31st International Workshop, IWOCA 2020, Bordeaux, France, June 8-10, 2020, Proceedings*, volume 12126 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2020. doi:10.1007/978-3-030-48966-3_15.

[45] Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM J. Comput.*, 34(4):825–847, 2005. doi:10.1137/S0097539701385351.

[46] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

[47] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. doi:10.1007/s002249910009.

[48] Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669. SIAM, 2016. doi:10.1137/1.9781611974331.ch113.

[49] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

[50] Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.

[51] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *FOCS*, pages 150–159. IEEE Computer Society, 2011.

[52] Marek Cygan, Marcin Pilipczuk, and Michal Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM J. Comput.*, 45(1):67–83, 2016. doi:10.1137/130947076.

[53] Artur Czumaj, Magnús M. Halldórsson, Andrzej Lingas, and Johan Nilsson. Approximation algorithms for optimization problems in graphs with superlogarithmic treewidth. *Inf. Process. Lett.*, 94(2):49–53, 2005. doi:10.1016/j.ipl.2004.12.017.

[54] Guilherme de C. M. Gomes, Carlos V. G. C. Lima, and Vinícius Fernandes dos Santos. Parameterized complexity of equitable coloring. *Discret. Math. Theor. Comput. Sci.*, 21(1), 2019. URL: http://dmtcs.episciences.org/5464.

[55] Holger Dell, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Tobias Mömke. Complexity and approximability of parameterized max-csps. *Algorithmica*, 79(1):230–250, 2017. doi:10.1007/s00453-017-0310-8.

[56] Erik D. Demaine, Martin L. Demaine, Nicholas J. A. Harvey, Ryuhei Uehara, Takeaki Uno, and Yushi Uno. UNO is hard, even for a single player. *Theor. Comput. Sci.*, 521:51–61, 2014. doi:10.1016/j.tcs.2013.11.023.

[57] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[58] Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated domination and covering: A parameterized perspective. In Martin Grohe and Rolf Niedermeier, editors, *Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings*, volume 5018 of *Lecture Notes in Computer Science*, pages 78–90. Springer, 2008. doi:10.1007/978-3-540-79723-4_9.

[59] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

[60] Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016.

[61] Gabriel L. Duarte, Mateus de Oliveira Oliveira, and Uéverton S. Souza. Co-degeneracy and co-treewidth: Using the complement to solve dense instances. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.42.

[62] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. New algorithms for mixed dominating set. *Discret. Math. Theor. Comput. Sci.*, 23(1), 2021. URL: http://dmtcs.episciences.org/7407.

[63] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. Upper dominating set: Tight algorithms for pathwidth and sub-exponential approximation. In Tiziana Calamoneri and Federico Corò, editors, *Algorithms and Complexity - 12th International Conference, CIAC 2021, Virtual Event, May 10-12, 2021, Proceedings*, volume 12701 of *Lecture Notes in Computer Science*, pages 202–215. Springer, 2021. doi:10.1007/978-3-030-75242-2_14.

[64] Vida Dujmovic, Gwenaël Joret, and David R. Wood. An improved bound for first-fit on posets without two long incomparable chains. *SIAM J. Discret. Math.*, 26(3):1068–1075, 2012. doi:10.1137/110855806.

[65] Pavel Dvorák, Eduard Eiben, Robert Ganian, Dusan Knop, and Sebastian Ordyniak. Solving integer linear programs with a small number of global variables and constraints. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 607–613. ijcai.org, 2017. doi:10.24963/ijcai.2017/85.

[66] Pavel Dvorák and Dusan Knop. Parameterized complexity of length-bounded cuts and multicuts. *Algorithmica*, 80(12):3597–3617, 2018. doi:10.1007/s00453-018-0408-7.

[67] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.

[68] Eduard Eiben, Robert Ganian, K. Kangas, and Sebastian Ordyniak. Counting linear extensions: Parameterizations by treewidth. *Algorithmica*, 81(4):1657–1683, 2019. doi:10.1007/s00453-018-0496-4.

[69] Eduard Eiben, Robert Ganian, and Sebastian Ordyniak. A structural approach to activity selection. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 203–209. ijcai.org, 2018. doi:10.24963/ijcai.2018/28.

[70] Rosa Enciso, Michael R. Fellows, Jiong Guo, Iyad A. Kanj, Frances A. Rosamond, and Ondrej Suchý. What makes equitable connected partition easy. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 122–133. Springer, 2009. doi:10.1007/978-3-642-11269-0_10.

[71] Alina Ene, Matthias Mnich, Marcin Pilipczuk, and Andrej Risteski. On routing disjoint paths in bounded treewidth graphs. In *SWAT*, volume 53 of *LIPIcs*, pages 15:1–15:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[72] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM J. Comput.*, 38(2):629–657, 2008.

[73] Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *J. Comput. Syst. Sci.*, 57(2):187–199, 1998. doi:10.1006/jcss.1998.1587.

[74] Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. doi:10.3390/a13060146.

[75] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Inf. Comput.*, 209(2):143–153, 2011. doi:10.1016/j.ic.2010.11.026.

[76] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph layout problems parameterized by vertex cover. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, volume 5369 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2008. doi:10.1007/978-3-540-92182-0_28.

[77] Michael R. Fellows, Frances A. Rosamond, Udi Rotics, and Stefan Szeider. Clique-width is NP-complete. *SIAM J. Discret. Math.*, 23(2):909–939, 2009. doi:10.1137/070687256.

[78] Wenceslas Fernandez de la Vega. MAX-CUT has a randomized approximation scheme in dense graphs. *Random Struct. Algorithms*, 8(3):187–198, 1996.

[79] Wenceslas Fernandez de la Vega and Marek Karpinski. Approximation complexity of non-dense instances of MAX-CUT. *Electron. Colloquium Comput. Complex.*, (101), 2006. URL: https://eccc.weizmann.ac.il/eccc-reports/2006/TR06-101/index.html.

[80] Jirí Fiala, Petr A. Golovach, and Jan Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theor. Comput. Sci.*, 412(23):2513–2523, 2011. doi:10.1016/j.tcs.2010.10.043.

[81] Johannes Klaus Fichte, Markus Hecher, and Andreas Pfandler. Lower bounds for qbfs of bounded treewidth. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 410–424. ACM, 2020. doi:10.1145/3373718.3394756.

[82] Krzysztof Fleszar, Matthias Mnich, and Joachim Spoerhase. New algorithms for maximum disjoint paths based on tree-likeness. In *ESA*, volume 57 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[83] Jacob Focke, Dániel Marx, and Pawel Rzazewski. Counting list homomorphisms from graphs of bounded treewidth: tight complexity bounds. *CoRR*, abs/2107.06889, 2021. URL: https://arxiv.org/abs/2107.06889, arXiv:2107.06889.

[84] Fedor V. Fomin, Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Bidimensionality. In *Encyclopedia of Algorithms*, pages 203–207. Springer, 2016. doi:10.1007/978-1-4939-2864-4_47.

[85] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Clique-width: on the price of generality. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 825–834. SIAM, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496860.

[86] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010. doi:10.1137/080742270.

[87] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Clique-width III: hamiltonian cycle and the odd case of graph coloring. *ACM Trans. Algorithms*, 15(1):9:1–9:27, 2019. doi:10.1145/3280824.

[88] Fedor V. Fomin and Dimitrios M. Thilikos. Branchwidth of graphs. In *Encyclopedia of Algorithms*, pages 232–237. Springer, 2016. doi:10.1007/978-1-4939-2864-4_55.

[89] Dimitris Fotakis, Michael Lampis, and Vangelis Th. Paschos. Sub-exponential approximation schemes for csps: From dense to almost sparse. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 37:1–37:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.STACS.2016.37.

[90] Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Log.*, 130(1-3):3–31, 2004. doi:10.1016/j.apal.2004.01.007.

[91] Jakub Gajarský and Petr Hlinený. Kernelizing MSO properties of trees of fixed height, and some consequences. *Log. Methods Comput. Sci.*, 11(1), 2015. doi:10.2168/LMCS-11(1:19)2015.

[92] Jakub Gajarský, Michael Lampis, Kazuhisa Makino, Valia Mitsou, and Sebastian Ordyniak. Parameterized algorithms for parity games. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, volume 9235 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 2015. doi:10.1007/978-3-662-48054-0_28.

[93] Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In Gregory Z. Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, volume 8246 of *Lecture Notes in Computer Science*, pages 163–176. Springer, 2013. doi:10.1007/978-3-319-03898-8_15.

[94] Robert Ganian, Petr Hlinený, Joachim Kneis, Alexander Langer, Jan Obdrzálek, and Peter Rossmanith. Digraph width measures in parameterized algorithmics. *Discret. Appl. Math.*, 168:88–107, 2014.

[95] Robert Ganian, Petr Hlinený, Joachim Kneis, Daniel Meister, Jan Obdrzálek, Peter Rossmanith, and Somnath Sikdar. Are there any good digraph width measures? *J. Comb. Theory, Ser. B*, 116:250–286, 2016.

[96] Robert Ganian, Fabian Klute, and Sebastian Ordyniak. On structural parameterizations of the bounded-degree vertex deletion problem. *Algorithmica*, 83(1):297–336, 2021. doi:10.1007/s00453-020-00758-8.

[97] Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artif. Intell.*, 257:61–71, 2018. doi:10.1016/j.artint.2017.12.006.

[98] Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. On structural parameterizations of the edge disjoint paths problem. *Algorithmica*, 83(6):1605–1637, 2021. doi:10.1007/s00453-020-00795-3.

[99] Robert Ganian and Stefan Szeider. New width parameters for SAT and #sat. *Artif. Intell.*, 295:103460, 2021. doi:10.1016/j.artint.2021.103460.

[100] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[101] Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, and Yota Otachi. Exploring the gap between treedepth and vertex cover through vertex integrity. In Tiziana Calamoneri and Federico Corò, editors, *Algorithms and Complexity - 12th International Conference, CIAC 2021, Virtual Event, May 10-12, 2021, Proceedings*, volume 12701 of *Lecture Notes in Computer Science*, pages 271–285. Springer, 2021. doi:10.1007/978-3-030-75242-2_19.

[102] Frank Gurski and Egon Wanke. The tree-width of clique-width bounded graphs without $K_{n,\,n}$. In Ulrik Brandes and Dorothea Wagner, editors, *Graph-Theoretic Concepts in Computer Science, 26th International Workshop, WG 2000, Konstanz, Germany, June 15-17, 2000, Proceedings*, volume 1928 of *Lecture Notes in Computer Science*, pages 196–205. Springer, 2000. doi:10.1007/3-540-40064-8_19.

[103] Gregory Z. Gutin, Mark Jones, and Magnus Wahlström. The mixed chinese postman problem parameterized by pathwidth and treedepth. *SIAM J. Discret. Math.*, 30(4):2177–2205, 2016. doi:10.1137/15M1034337.

[104] Tesshu Hanaka, Ioannis Katsikarelis, Michael Lampis, Yota Otachi, and Florian Sikora. Parameterized orientable deletion. *Algorithmica*, 82(7):1909–1938, 2020. doi:10.1007/s00453-020-00679-6.

[105] Frank Harary and C St JA Nash-Williams. On eulerian and hamiltonian graphs and line graphs. *Canadian Mathematical Bulletin*, 8(6):701–709, 1965.

[106] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[107] Ararat Harutyunyan, Michael Lampis, and Nikolaos Melissinos. Digraph coloring and distance to acyclicity. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 41:1–41:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.41.

[108] Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[109] Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. *Theor. Comput. Sci.*, 399(3):206–219, 2008.

[110] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

[111] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

[112] Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. In Dimitris Fotakis, Aris Pagourtzis, and Vangelis Th. Paschos, editors, *Algorithms and Complexity - 10th International Conference, CIAC 2017, Athens, Greece, May 24-26, 2017, Proceedings*, volume 10236 of *Lecture Notes in Computer Science*, pages 345–356, 2017. doi:10.1007/978-3-319-57586-5_29.

[113] Pallavi Jain, M. Jayakrishnan, Fahad Panolan, and Abhishek Sahu. Mixed dominating set: A parameterized perspective. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, volume 10520 of *Lecture Notes in Computer Science*, pages 330–343. Springer, 2017. doi:10.1007/978-3-319-68705-6_25.

[114] Ramin Javadi and Amir Nikabadi. On the parameterized complexity of sparsest cut and small-set expansion problems, 2019. arXiv:1910.12353.

[115] Thor Johnson, Neil Robertson, Paul D. Seymour, and Robin Thomas. Directed tree-width. *J. Comb. Theory, Ser. B*, 82(1):138–154, 2001.

[116] Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983. doi:10.1287/moor.8.4.538.

[117] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987. doi:10.1287/moor.12.3.415.

[118] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

[119] Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structural parameters, tight bounds, and approximation for (k, r)-center. *Discret. Appl. Math.*, 264:90–117, 2019. doi:10.1016/j.dam.2018.11.002.

[120] Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structurally parameterized d-scattered set. *Discrete Applied Mathematics*, 2020. URL: https://www.sciencedirect.com/science/article/pii/S0166218X20301517, doi:https://doi.org/10.1016/j.dam.2020.03.052.

[121] Chamanvir Kaur and Neeldhara Misra. On the parameterized complexity of spanning trees with small vertex covers. In Manoj Changat and Sandip Das, editors, *Algorithms and Discrete Applied Mathematics - 6th International Conference, CALDAM 2020, Hyderabad, India, February 13-15, 2020, Proceedings*, volume 12016 of *Lecture Notes in Computer Science*, pages 427–438. Springer, 2020. doi:10.1007/978-3-030-39219-2_34.

[122] Leon Kellerhals and Tomohiro Koana. Parameterized complexity of geodetic set. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.IPEC.2020.20.

[123] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.

[124] Dusan Knop, Tomás Masarík, and Tomás Toufar. Parameterized complexity of fair vertex evaluation problems. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 33:1–33:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.33.

[125] Daniel Kobler and Udi Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discret. Appl. Math.*, 126(2-3):197–221, 2003. doi:10.1016/S0166-218X(02)00198-1.

[126] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012. doi:10.1007/s00453-011-9554-x.

[127] Michael Lampis. Parameterized maximum path coloring. *Theor. Comput. Sci.*, 511:42–53, 2013. doi:10.1016/j.tcs.2013.01.012.

[128] Michael Lampis. Model checking lower bounds for simple graphs. *Log. Methods Comput. Sci.*, 10(1), 2014. doi:10.2168/LMCS-10(1:18)2014.

[129] Michael Lampis. Parameterized approximation schemes using graph widths. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 775–786. Springer, 2014. doi:10.1007/978-3-662-43948-7_64.

[130] Michael Lampis. Finer tight bounds for coloring on clique-width. *SIAM J. Discret. Math.*, 34(3):1538–1558, 2020. doi:10.1137/19M1280326.

[131] Michael Lampis. Minimum stable cut and treewidth. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 92:1–92:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.92.

[132] Michael Lampis, Georgia Kaouri, and Valia Mitsou. On the algorithmic effectiveness of digraph decompositions and complexity measures. *Discret. Optim.*, 8(1):129–138, 2011. doi:10.1016/j.disopt.2010.03.010.

[133] Michael Lampis, Kazuhisa Makino, Valia Mitsou, and Yushi Uno. Parameterized edge hamiltonicity. *Discret. Appl. Math.*, 248:68–78, 2018. doi:10.1016/j.dam.2017.04.045.

[134] Michael Lampis, Stefan Mengel, and Valia Mitsou. QBF as an alternative to Courcelle's theorem. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 235–252. Springer, 2018. doi:10.1007/978-3-319-94144-8_15.

[135] Michael Lampis and Valia Mitsou. Treewidth with a quantifier alternation revisited. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, volume 89 of *LIPIcs*, pages 26:1–26:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.IPEC.2017.26.

[136] Michael Lampis and Valia Mitsou. Fine-grained meta-theorems for vertex integrity, 2021. Accepted to ISAAC 2021. arXiv:2109.10333.

[137] Bingkai Lin. A simple gap-producing reduction for the parameterized set cover problem. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 81:1–81:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.81.

[138] Bingkai Lin. Constant approximating k-clique is w[1]-hard. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1749–1756. ACM, 2021. doi:10.1145/3406325.3451016.

[139] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bull. EATCS*, 105:41–72, 2011. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/92.

[140] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.

[141] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM J. Comput.*, 47(3):675–702, 2018. doi:10.1137/16M1104834.

[142] Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Fpt-approximation for FPT problems. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 199–218. SIAM, 2021. doi:10.1137/1.9781611976465.14.

[143] Dániel Marx and Valia Mitsou. Double-exponential and triple-exponential bounds for choosability problems parameterized by treewidth. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.28.

[144] Dániel Marx and Michal Pilipczuk. Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask). In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 542–553. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.542.

[145] Kitty Meeks and Alexander Scott. The parameterised complexity of list problems on graphs of bounded treewidth. *Inf. Comput.*, 251:91–103, 2016. doi:10.1016/j.ic.2016.08.001.

[146] Kitty Meeks and Fiona Skerman. The parameterised complexity of computing the maximum modularity of a graph. *Algorithmica*, 82(8):2174–2199, 2020. doi:10.1007/s00453-019-00649-7.

[147] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

[148] Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *J. ACM*, 57(5):29:1–29:29, 2010. doi:10.1145/1754399.1754402.

[149] Jaroslav Nesetril and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006. doi:10.1016/j.ejc.2005.01.010.

[150] André Nichterlein, Rolf Niedermeier, Johannes Uhlmann, and Mathias Weller. On tractable cases of target set selection. *Soc. Netw. Anal. Min.*, 3(2):233–256, 2013. doi:10.1007/s13278-012-0067-7.

[151] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. doi:10.1093/ACPROF:OSO/9780198566076.001.0001.

[152] Sebastian Ordyniak, Daniël Paulusma, and Stefan Szeider. Satisfiability of acyclic and almost acyclic CNF formulas. *Theor. Comput. Sci.*, 481:85–99, 2013. doi:10.1016/j.tcs.2012.12.039.

[153] Sang-il Oum. Rank-width and vertex-minors. *J. Comb. Theory, Ser. B*, 95(1):79–100, 2005. doi:10.1016/j.jctb.2005.03.003.

[154] Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B*, 96(4):514–528, 2006. doi:10.1016/j.jctb.2005.10.006.

[155] Guoqiang Pan and Moshe Y. Vardi. Fixed-parameter hierarchies inside PSPACE. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 27–36. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.25.

[156] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075. SIAM, 2010. doi:10.1137/1.9781611973075.86.

[157] Michal Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011. doi:10.1007/978-3-642-22993-0_47.

[158] Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In *ICALP (1)*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. Springer, 2014.

[159] Sigve Hortemo Sæther and Jan Arne Telle. Between treewidth and clique-width. *Algorithmica*, 75(1):218–253, 2016. doi:10.1007/s00453-015-0033-7.

[160] Marko Samer and Stefan Szeider. Constraint satisfaction with bounded treewidth revisited. *J. Comput. Syst. Sci.*, 76(2):103–114, 2010. doi:10.1016/j.jcss.2009.04.003.

[161] Marko Samer and Stefan Szeider. Tractable cases of the extended global cardinality constraint. *Constraints An Int. J.*, 16(1):1–24, 2011. doi:10.1007/s10601-009-9079-y.

[162] Petra Scheffler. A linear algorithm for the pathwidth of trees. In *Topics in combinatorics and graph theory*, pages 613–620. Springer, 1990.

[163] Friedrich Slivovsky and Stefan Szeider. Model counting for formulas of bounded clique-width. In Leizhen Cai, Siu-Wing Cheng, and Tak Wah Lam, editors, *Algorithms and Computation - 24th International Symposium, ISAAC 2013, Hong Kong, China, December 16-18, 2013, Proceedings*, volume 8283 of *Lecture Notes in Computer Science*, pages 677–687. Springer, 2013. doi:10.1007/978-3-642-45030-3_63.

[164] Stefan Szeider. Not so easy problems for tree decomposable graphs. *CoRR*, abs/1107.1177, 2011. URL: http://arxiv.org/abs/1107.1177, arXiv:1107.1177.

[165] Marc Tedder, Derek G. Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *ICALP (1)*, volume 5125 of *Lecture Notes in Computer Science*, pages 634–645. Springer, 2008.

[166] Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discrete Math.*, 10(4):529–550, 1997. doi:10.1137/S0895480194275825.

[167] Luca Trevisan. Inapproximability of combinatorial optimization problems. *CoRR*, cs.CC/0409043, 2004. URL: http://arxiv.org/abs/cs.CC/0409043.

[168] Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.

[169] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001. URL: http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-65367-7.

[170] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE.