Technical Report

UCAM-CL-TR-903
ISSN 1476-2986

Number 903

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Parameterized complexity of distances to sparse graph classes

Jannis Bulian

February 2017

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

© 2017 Jannis Bulian

This technical report is based on a dissertation submitted February 2016 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Clare College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Abstract

This dissertation offers contributions to the area of parameterized complexity theory, which studies the complexity of computational problems in a multivariate framework. We demonstrate that for graph problems, many popular parameters can be understood as distances that measure how far a graph is from belonging to a class of sparse graphs. We introduce the term *distance parameter* for such parameters and demonstrate their value in several ways.

The parameter *tree-depth* is uncovered as a distance parameter, and we establish several fixed-parameter tractability and hardness results for problems parameterized by tree-depth.

We introduce a new distance parameter *elimination distance to a class* $\mathcal{C}$. The parameter measures distance in a way that naturally generalises the notion of vertex deletion by allowing deletions from every component in each deletion step. We show that tree-depth is a special case of this new parameter, introduce several characterisations of elimination distance, and discuss applications. In particular we demonstrate that Graph Isomorphism is FPT parameterized by elimination distance to bounded degree, extending known results. We also show that checking whether a given graph has elimination distance k to a minor-closed class $\mathcal{C}$ is FPT, and, moreover, if we are given a finite set of minors characterising $\mathcal{C}$, that there is an explicit FPT algorithm for this.

We establish an algorithmic meta-theorem for several distance parameters, by showing that all graph problems that are both slicewise nowhere dense and slicewise first-order definable are FPT.

Lastly, several fixed-parameter tractability results are established for two graph problems that have natural distance parameterizations.

Acknowledgements

First and foremost I thank my supervisor, Anuj Dawar, whose advice has been invaluable to me. I could not have asked for a more patient, accessible and supportive mentor. I also thank Timothy Griffin and Andrew Pitts for their helpful feedback on my first-year report.

This work would not have been possible without the financial support from the Qualcomm Research Scholarship and the EPSRC.

I thank Pengming Wang and Nik Sultana for their comments on a draft of this dissertation.

I also thank my parents for their continuous support throughout my years of study (and my life in general). Lastly, I am particularly grateful to Doerte Letzmann for her invaluable support over the last decade.

Contents

chapter 1

Introduction

The question which computational tasks can be efficiently solved in a systematic way has been studied for a long time. The first algorithms date back as far as ancient Greece – famous examples include the *Sieve of Eratosthenes* for identifying prime numbers and *Euclid's algorithm* for finding the greatest common divisor of two numbers. Today, with computers being ubiquitous, algorithms are more important than ever. One of the central questions is: what computational tasks (usually called *problems*) can we expect to solve algorithmically. Or, which problems can we practically solve on a computer?

Alan Turing showed that there are limits to what an algorithm can do. Turing [152] famously formalised computation by introducing Turing machines, and proved that there are problems that can be solved by no algorithm: He showed that the halting problem for Turing machines is undecidable – that there is no algorithm that, taking another algorithm as an input, can decide whether that input algorithm will ever halt. He used this to show that Hilbert's 'Entscheidungsproblem' is undecidable.

Turing pointed out a fundamental limit of computation, and other problems have been shown to be uncomputable. However, even if we know that a computational problem can in principle be solved by an algorithm, it might still not be feasible. Solving the problem might require more computational resources (i.e. space or time) than available. With the advent of machines that perform computations in the last century, the question of required resources to solve a computational problem has become even more relevant.

Although there is some previous work, the systematic study of the resources required to solve a problem began in the 1960s. The area of *computational complexity theory* focuses on problems rather than algorithms and its main goal is to determine the intrinsic complexity of computational problems. Problems are classified by the computational resources needed by an algorithm to solve them, for example the required time or space. The goal is to distinguish problems that can be solved in reasonable time or space, called *tractable* problems, from problems that use too many resources, called *intractable* problems.

Many problems that are studied in complexity theory are either *graph problems* or can be rephrased in terms of graphs. Graphs are mathematical objects used to model the relations between entities, for instance the 'friendship' relation in a social network or the structure of molecules in chemistry. A graph problem is then a well-defined question about graphs. This dissertation is concerned with graph problems.

The (time) complexity of a problem is the number of computational steps required by the best known algorithm to solve the problem, measured as a function of the size of the

input. To give an example, a graph with n vertices can be represented as a table with an entry for every pair of vertices, where every field tells us if a pair of vertices is adjacent. Such a table has n^2 entries, and that is the size of the input of a graph problem.

A problem is usually considered to be tractable if the time complexity is bounded by a polynomial. The class of all problems that can be solved in polynomial time is called P. It is harder to say when a problem is intractable. Researchers in complexity theory lack the tools to show lower bounds on problems, i.e. we do not know how to show that a problem is not contained in P. In the absence of provable lower bounds, a problem is usually considered to be intractable if a problem that we think is hard can be reduced to it (that is, a solution to the problem we want to show is hard can be used to solve a known hard problem). The most prominent example is the class of NP-hard problems.

The class NP is the class of problems where, given a solution to the problem, we can verify the solution in polynomial time. A problem is NP-hard if it is at least as hard as all the problems in NP. Clearly P is a subclass of NP (if we can find a solution, we can also verify it). The question whether P is a proper subclass of NP, i.e. if $\mathsf{P} \neq \mathsf{NP}$, is one of the major open questions in complexity theory (and mathematics[1] as a whole) since the 1970s.

Problems that are contained in NP (i.e. they have an easily checkable solution) and that are also NP-hard (i.e. an algorithm solving them is powerful enough to solve all problems in NP) are called NP-complete. Many graph problems have been shown to be NP-complete and only exponential algorithms are known that solve them in general. However, many such problems are important and still have to be solved in practice, for instance in the areas of computational biology or the study of social networks.

Moreover, the distinction between problems that are in P and problems that are NP-complete does not seem to fully answer our question above: what problems can we expect to solve using computers? Practitioners still succeed in solving NP-complete problems in practice. For example, algorithms for the boolean satisfiability problem (SAT) continue to improve, and researchers submit their implementations to competitions that solve instances in reasonable time. Does that mean that the theory is irrelevant? No, it just means that the instances that are solved in practice are very different from the worst case and classical complexity theory does not address this issue well.

A more fine-grained study of computational complexity, introduced in the 1990s by Downey and Fellows, helps address these issues. In *parameterized complexity theory*, we are not just looking at the runtime of an algorithm as a function of the size of the input, but instead we also take other 'parameters' of the problem into account. As an illustration, the SAT problem becomes easy if we have just a few variables, or just a few clauses. This is what parameterized complexity tries to capture. A problem is said to be *fixed-parameter tractable* if it can be solved in time $O(f(k) \cdot p(n))$ for some computable function f (of the parameter k) and polynomial p (of the size of the input n). The SAT problem is fixed-parameter tractable parameterized by the number of variables in the formula, and also by the number of clauses. This theory gives a good explanation of why some boolean satisfiability problems can be solved in practice: if the formulas were written by a human (or are derived directly from a human specification in some way), then one would expect a bias in these formulas to use few variables or few clauses. Thus it makes sense that in practice many instances of the SAT problem are solvable in reasonable time.

[1] It is one of the famous Millenium Problems and there is a $1,000,000 prize from the Clay Mathematical institute for solving it: http://www.claymath.org/millennium-problems/p-vs-np-problem

Parameterized complexity theory also comes with its own notion of hardness. Just as in classical complexity theory there are no tools to prove that a problem is not fixed-parameter tractable. Instead a problem is considered to be intractable if it is shown to be W[1]-hard.

The parameters are supposed to capture what is hard about a problem, and the intuition is that a problem is fixed-parameter tractable if it becomes tractable for small values of the parameter. Of course this is very valuable for practical applications, where practitioners can take this into account to design algorithms that exploit small values of the parameter. But it is also very interesting for computer science theory: it gives us an idea of what, fundamentally, makes the problem hard.

How do we choose the parameters? The parameters should be broad enough and relevant for applications, while still allowing many problems to become tractable. A popular choice of parameter is just the size of the output. For instance, if we are looking for a vertex cover in a graph, it would be reasonable to expect that the problem is easier if we can find a small vertex cover. This is sometimes called the 'natural parameterization'.

Another approach used for graph problems is to take the structure of the input graph into account. Many NP-complete graph problems become tractable on classes of sparse graphs, such as planar graphs or graphs of bounded tree-width. Tree-width is a measure of how far a graph is from being a tree and is in fact a widely used parameter in the study of parameterized graph problems.

In this dissertation we identify a particularly interesting group of structural graph parameters that we refer to as *distance parameters*. They measure how far a graph is from belonging to some class of graphs. To give an example, the so called 'Vertex Deletion' problems ask whether we can remove k vertices from a given graph to place the graph in some class $\mathcal{C}$, where k is the parameter. Many graph problems are in fact of that form, even if it is not immediately apparent: the vertex cover number of a graph is just the minimum number of vertices we have to delete from the graph in order to obtain an edgeless graph. The feedback vertex set number is just the minimum number of vertices we have to delete from the graph in order to obtain a forest. The graph bipartization problem asks to delete the minimal amount of vertices to obtain a bipartite graph. More generally, Cai [32] studies the parameterized complexity of distance measures defined in terms of addition and deletion of vertices and edges to hereditary classes $\mathcal{C}$, i.e. graph classes that are closed under induced subgraphs.

But the notion of distance can be interpreted much more widely. Counting deletions of vertices and edges gives a rather simple notion of distance, and many more involved notions have also been studied. For example the popular parameter tree-width. Another classic example is the crossing number of a graph, which provides a notion of distance to the class of planar graphs.

A common motivation for studying distance parameters is the following: if a problem one wishes to solve is known to be tractable on a class of graphs $\mathcal{C}$, then it is reasonable to conjecture that the problem might also be tractable for graphs that are 'almost' in $\mathcal{C}$, or close to $\mathcal{C}$ in some sense. Thus the distance to $\mathcal{C}$ provides an interesting parameterization of that problem (called distance to triviality by Guo et al. [95]). Since many problems are tractable on classes of sparse graphs, distances to sparse graphs are a good target for this.

Take for example the Vertex Cover problem, one of the most studied problems in parameterized complexity theory. It can be solved in polynomial time on the class of

bipartite graphs, and on trees. If we consider the deletion distance to trees as a parameter, then it is fixed-parameter tractable – one way to see this is by considering that the tree-width of a graph is in this case bounded by the parameter, and Vertex Cover is known to be FPT parameterized by tree-width [17]. Other successful examples of this strategy include the study of modulators to graphs of bounded tree-width in the context of kernelization (see [73, 78]), or the parameterizations of colouring problems (see [121]). In the context of satisfiability problems, this strategy has lead so-called backdoors[81, 83]; parameterizations by the number of variables that need to be fixed to place the problem in a class that can be solved in polynomial time.

Another topic of research is the relationship between parameters. The study of the relationship between parameters leads to a better understanding of the complexity landscape and generally offers a good way to discover new tractable or intractable parameterizations of problems. Figure 1.1 shows which of the distance parameters in this dissertation provide an upper bound for other graph parameters. For many problems it is unknown for exactly which parameters they are tractable. For example, it might be known that the graph problem is fixed-parameter tractable parameterized by *vertex cover number*, and that it is W[1]−hard parameterized by *tree-width*. But its complexity status for the parameters *tree-depth* or *path-width* might be unknown. A better understanding of the relationship between the parameters allows us to ask the right questions, which might eventually lead us to close these gaps and draw the line between tractability and intractability in a more precise way.

In this dissertation we make contributions to the study of distance parameters in several ways. In Chapter 4 we discuss the distance parameter *tree-depth*, which is located in-between vertex cover number and tree-width. We establish several fixed-parameter tractability and hardness results for problems parameterized by tree-depth, closing some gaps in the knowledge for these problems.

In Chapter 5 we introduce a new parameter, *elimination distance to a class* $\mathcal{C}$, that is a natural generalisation of both deletion distance to a class $\mathcal{C}$ and tree-depth. Just as deletion distance is defined with respect to some class $\mathcal{C}$, it is defined as the elimination distance to some class $\mathcal{C}$. Recall that vertex cover number is the deletion distance to the class of edgeless graphs. Similarly, we show in Section 5.4 that tree-depth can be seen to be the elimination distance to the class of edgeless graphs. Indeed, elimination distance generalises deletion distance in the same way that tree-depth generalises vertex cover number.

Broadly speaking, there are two kinds of questions one can ask about a graph parameter: Given a graph G, is it FPT to determine the value of the parameter for the graph, or, in other words, is it FPT to check whether the graph has parameter at most k parameterized by k? The second kind of question is: What kind of graph problems become tractable when parameterized by the parameter? We discuss both these questions for elimination distance.

We show an application of the parameter in Chapter 6, where we demonstrate that Graph Isomorphism is FPT parameterized by elimination distance to bounded degree. This generalises a result of Bouland et al. [23] and is an example of the second kind of question.

In Chapter 7 we give an answer to a question of the first kind. We show that given the finite set of minors that characterize a minor-closed class of graphs $\mathcal{C}$, we can find the minors that characterize the graphs with elimination distance k to $\mathcal{C}$. So if the set of

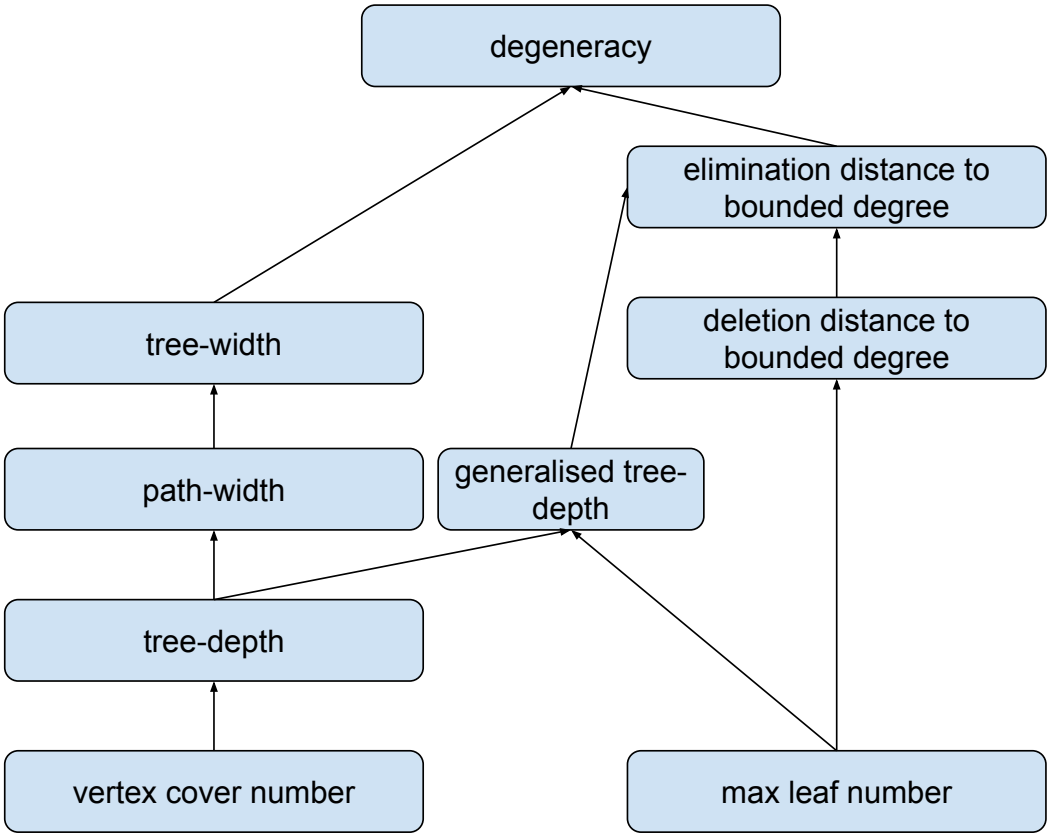

Figure 1.1: A diagram that illustrates the relationships between distance parameters to sparse graph classes that are part of this dissertation. An arrow between two parameters should be interpreted as 'a function of the first parameter gives an upper bound for the second parameter'.

excluded minors is available for a minor-closed class $\mathcal{C}$, there is an explicit FPT algorithm to test for membership in the class of graphs with elimination distance k to $\mathcal{C}$.

Algorithmic meta-theorems are algorithmic results that apply not only to a single problem, but to a whole class of problems. A seminal result in the area is Courcelle's Theorem, which states that every problem definable in monadic second-order logic can be decided in linear time on bounded tree-width[2]. Our result in Chapter 7 is an algorithmic meta-theorem because it applies to all minor-closed graph classes.

In Chapter 8, building on recent work of Grohe et al. [92], we establish another algorithmic meta-theorem. We show that every graph problem that is both slicewise nowhere dense and slicewise first-order definable (these terms are defined precisely in Definition 8.2.1 and Definition 8.2.2) is FPT.

Lastly, in Chapter 9 we discuss two problems that have natural distance parameters. We parameterize by these parameters, and also consider additional structural distance parameters. We establish fixed-parameter tractability results for these parameterizations.

1.1 Related areas of research

Here we give an overview of the different branches of mathematics and theoretical computer science that are related to the research of this dissertation. More specific related work is mentioned in the individual chapters.

Parameterized complexity theory. This dissertation explores the parameterized complexity of several problems with regard to distance parameters. Parameterized complexity was introduced in the 1990s by Downey and Fellows [56] and extends the classical (one-dimensional) complexity theory to a two-dimensional theory that measures the complexity of a problem not only as a function of the size of the input, but also in terms of an additional parameter.

The general aim is a more fine-grained analysis of (NP-)hard problems, gaining a better understanding of tractable instances. There is a large amount of related work in that area, which is referred to in the individual chapters.

We formally introduce the theory and state all the important definitions in Section 2.2.

Graph editing: vertex and edge deletion/addition problems. A natural notion of distance of a graph to a class of graphs comes from editing operations on a graph. How many vertex or edge deletions or additions do we have to perform on a graph G to put it into some class $\mathcal{C}$?

In the 1980s Lewis and Yannakakis [112] showed that vertex deletion problems are NP-hard for any hereditary non-trivial classes, and Yannakakis [159] showed that edge deletion problems are NP-hard for several graph classes.

These hardness results made these problems a natural candidate for the study of their parameterized complexity. For this reason researchers in the 1990s focused on showing tractability or hardness of such problems within the framework of parameterized complexity.

Kaplan et al. [102] showed the tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. Cai [32] showed that the deletion

[2]The statement can in fact be extended to clique-width; see Section 3.5.

distance problem to any hereditary property that is characterised by a finite set of forbidden subgraphs is fixed-parameter tractable (also see below). Cai [33] also showed that Graph Colouring is FPT parameterized by the some edit distances to split graphs and bipartite graphs. Mathieson [123] studies the parameterized complexity of a range of graph editing problems in his doctoral dissertation, where the graphs are edited to satisfy a variety of degree constraints.

It is also possible to go one step further and consider the structure of the subgraph induced by the deleted vertices. A set of vertices that, if removed from a graph, places it in a certain class of graphs is sometimes also called a *modulator*. Instead of considering the size, one can also consider the structure of the modulators . [58, 82].

Finite model theory. Model theory is the study of mathematical structures (e.g. graphs) from the perspective of logic.

Finite model theory is a branch of model theory that only deals with finite structures, i.e. structures over a finite universe. Many important theorems, e.g. the compactness theorem, do only hold for infinite structures, so finite model theory is quite different from general model theory that also allows infinite structures. The objects studied in computer science are usually finite, so it found many applications here: it is used in descriptive complexity theory, database theory and formal language theory. Most of the research in the area is about the expressive power of logics on classes of structures. Libkin [113] gives a good introduction to the area in his book.

In Chapter 8 we use model theoretic language to study problems that are (slicewise) definable in first-order logic and slicewise nowhere dense.

Descriptive complexity theory. Descriptive complexity theory is concerned with the application of logic to questions in complexity theory. The main driving question is the application of logic to the P vs NP question by attempting to solve the corresponding question in model theory: Is there a logic that captures P and a logic that captures NP, and a way to show that these logics are different?

An important result in the area is Fagin's theorem [61], first proved in his PhD dissertation, which states that the class NP is exactly the set of languages that can be defined using existential second-order logic. But while there is a logic that captures NP, a logic that captures P continues to be elusive. However, the search for such a logic has spawned a tremendous amount of research. Most results in descriptive complexity are about characterising complexity classes by the logics that can express the problems in them; see the book of Immerman [100] for a good overview.

Algorithmic meta-theorems. Descriptive complexity theory spawned another subfield, often called algorithmic meta-theorems. It (mostly) studies the computational complexity of the model checking problem on different classes of structures, usually graphs. Results in the area are called algorithmic meta-theorems, because they yield algorithmic results that (instead of just a single problem) apply to a whole class of problems.

The first main result in the area was Courcelle's Theorem [39], which showed that all graph properties definable in monadic second-order logic can be decided in linear time on graphs of bounded tree-width. In the last 20 years there has been ample research on similar questions and it is now an established area, closely related to finite model theory and descriptive complexity [91, 109].

Although most results in the area are related to evaluating formulas from different logics, there are also meta-theorems unrelated to logic. For example the result of Cai [32] that the deletion distance problem to any hereditary property that is characterised by a finite set of forbidden subgraphs is fixed-parameter tractable can be viewed as an algorithmic meta-theorem.

A major quest over the last 20 years was to pin down the graph classes (closed under taking subgraphs) for which first-order model checking is FPT. It is well known that evaluating first-order formulas on general structures is PSPACE-hard. This holds even on structures with only two elements, which is why structural restrictions alone will not reduce the complexity. However, the tools from parameterized complexity allow for a finer analysis. The general problem of first-order model checking parameterized by the size of the formula is in XP. So the interesting question is: for which graph classes is it FPT?

Showing that first-order model checking is FPT on a class of graphs gives us an algorithmic meta-theorem in the following way: if a parameterized problem is slicewise first-order definable (i.e. the size of the formula depends on the parameter), then the problem becomes fixed-parameter tractable on that class. For instance the *independent set problem* parameterized by the size of the independent set can be translated into a first-order formula with size bounded in the parameter; the formula for the kth slice is the following:

$$\exists x_1 \ldots \exists x_k . \bigwedge_{i<j} \neg E(x_i, x_j).$$

The question has been researched for the last two decades. In particular, there has been important research on proving such results for classes of sparse graphs in the recent past. Courcelle's result on monadic second-order logic was later generalised to clique-width/rank-width [42] and there is reason to believe it can not be pushed any further. For many sparse graph classes, i.e. where the number of edges as a function of the minimum number of vertices grows only slowly, the problem of evaluating first-order sentences in the class becomes fixed-parameter tractable. Seese [147] showed this for the class of graphs of bounded degree. Frick and Grohe [76] succeeded in generalising this further, to the class of graphs of bounded local tree-width, which includes the classes of bounded tree-width, bounded degree and planar graphs. Flum and Grohe [71] proved that evaluating a first-order sentence is also fixed-parameter tractable for the class of graphs that exclude a minor. Dawar, Grohe and Kreutzer [50] further generalised this result to the class of graphs that locally exclude a minor, which includes the class of graphs of bounded local tree-width and also the class of graphs that exclude a minor. The result by Dvořák, Král and Thomas [57] generalises this even more, by describing a linear time algorithm for evaluating first-order sentences on graphs of (locally) bounded expansion, which includes graphs that (locally) exclude minors. The latest result by Grohe et al. [92] likely pushed this to its limit by showing that first-order logic is FPT on nowhere dense classes of graphs. Kreutzer and Dawar [53] showed that first-order logic is not fixed-parameter tractable on somewhere dense classes of graphs (unless FPT = AW[$*$]).

So there has been tremendous progress over the years in generalising the result to more and more general classes of sparse graphs.[3] See Figure 1.2 for an illustration of the

[3]It should be remarked that sparse graphs are not the only graphs for which first-order model checking is fixed-parameter tractable. For example the class of graphs of bounded clique-width mentioned above is not sparse, but does allow for efficient evaluation of first-order sentences [42].

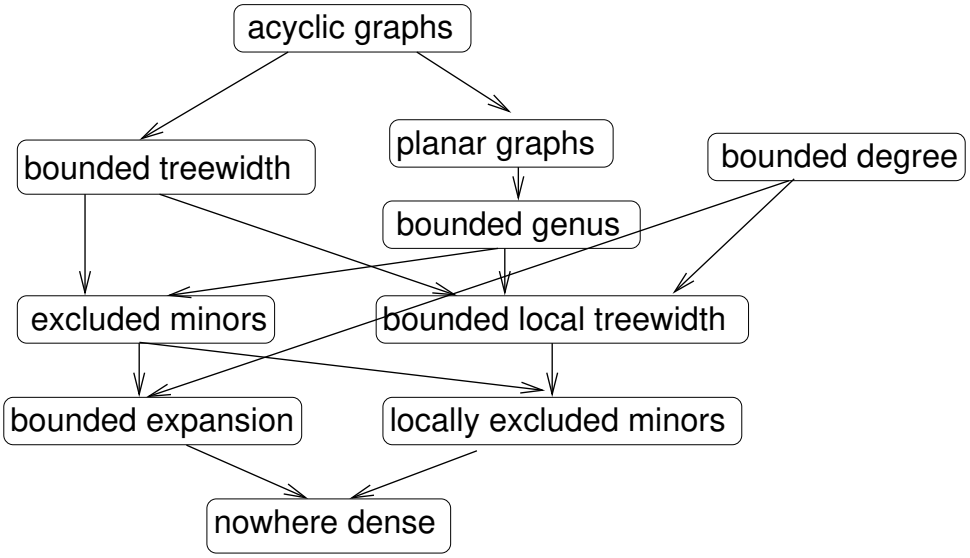

Figure 1.2: An illustration of sparse graph classes and their inclusion relation (taken from [48]).

relationships between some sparse graph classes.

In Chapter 8 we extend the result by Grohe et al. [92] to show that any graph property that is slicewise first-order definable and slicewise nowhere dense (these terms are defined precisely in Definition 8.2.1 and Definition 8.2.2) is decidable in FPT time.

1.2 Contribution

This section gives an overview of the chapters, detailing our contribution in each of the chapters.

1.2.1 Chapter overview

Chapter 2 In order to provide the necessary background for the rest of the dissertation, we introduce the basic notions of first-order model theory, parameterized complexity theory, some areas of graph theory, and order theory.

Chapter 3 This chapter provides additional background. It introduces the notion of a *distance parameter*, defined to be a parameter that measures how far a graph is from being contained in some class. It then discusses several structural graph parameters that can be understood as distance parameters.

Chapter 4 This chapter is concerned with the distance parameter *tree-depth* and exhibits several results about it. We prove fixed-parameter tractability and hardness results for a number of problems parameterized by tree-depth.

Chapter 5 In this chapter we introduce the new graph parameter *elimination distance to a class* $\mathcal{C}$. The parameter naturally generalises the notion of deletion in deletion dis-

tance, and we prove that tree-depth is just the elimination distance to the class of edgeless graphs. We show that the parameter offers a range of algorithmically interesting characterisations.

Chapter 6 In this chapter we give an application of the parameter elimination distance introduced in Chapter 5. We discuss the Graph Isomorphism problem and establish that Graph Canonisation, and thus Graph Isomorphism, is FPT when parameterized by elimination distance to bounded degree, extending results of Bouland et al. [23].

Chapter 7 This chapter discusses another application of the parameter elimination distance introduced in Chapter 5. We prove an algorithmic meta-theorem, establishing that the problem of determining elimination distance to any minor-closed class $\mathcal{C}$ is FPT. We show that we can provide an explicit FPT algorithm if a set of forbidden minors characterising $\mathcal{C}$ is given.

Chapter 8 In this chapter we prove another algorithmic meta-theorem. We establish that any problem that is both *slicewise nowhere dense* and *slicewise first-order definable* (these terms are defined in Definition 8.2.1 and Definition 8.2.2) is FPT.

Chapter 9 In this chapter we discuss two graph problems parameterized by their natural distance parameters. First we show that Clique Dominating Set is FPT on nowhere dense classes of graphs. Next, we consider the Anchored k-core problem and show that it is FPT parameterized by tree-width and budget, as well as degree, size of the core and clique-width.

Chapter 10 Here we give a final discussion of the results established in this dissertation, and discuss a wide range of potential further research directions.

1.2.2 Contribution details

The content of Chapter 5 and Chapter 6 is based on [28] and [30], both being coauthored with Anuj Dawar. The content of Chapter 7 and Chapter 8 is based on [29], also coauthored with Anuj Dawar.

CHAPTER 2

General background

In order to provide the necessary background for the rest of this dissertation, we here introduce the basic notions of first-order model theory, parameterized complexity theory, some areas of graph theory and order theory.

We introduce model theory in Section 2.1, giving a brief overview of first-order logic and second-order logic. Next in Section 2.2 we introduce the elementary notions of parameterized complexity theory, briefly mentioning kernelizations and an equivalent formalisation using model theoretic notions. In Section 2.3, we give a brief introduction of graph theory and discuss the basics of graph minor theory, nowhere dense graph classes and graph classes with bounded expansion, as well as the the graph isomorphism problem. Lastly we introduce some basic order theoretic notions in Section 2.4.

In Chapter 3 we introduce more specific background: there we motivate the notion of *distance parameters*, i.e. structural graph parameters that measure how far a graph is from being contained in some class of graphs, and introduce several such parameters.

2.1 Model theory

Here we introduce the elementary notions of model theory, roughly following the book by Hodges [98].

The core notion of model theory is that of a structure, and based on that we define substructures, expansions and reducts.

Definition 2.1.1. A *(relational) signature* or *vocabulary* σ is a finite set of relation symbols, each with an associated non-negative integer, called its *arity*, and a finite set of constant symbols.

Definition 2.1.2. A *σ-structure* A consists of a set $V(A)$ called the *universe of* A and for each k-ary relation symbol $R \in \sigma$ a relation $R(A) \subseteq V(A)^k$, and for each constant symbol $c \in \sigma$, an element $c(A) \in V(A)$. A σ-structure A is called finite if its universe is finite.

Remark. Our structures are mostly (coloured) graphs, so $\sigma = \{E\}$ or $\sigma = \{E, C_1, C_2, \ldots, C_r\}$ where E is the (binary) edge relation and the C_i are unary relation symbols and we do not have any constants. A graph G is then a finite σ-structure with vertex set $V(G)$, edge relation $E(G)$, and colours $C_i(G)$.

Considering just a part of the elements of a structure gives rise to substructures.

Definition 2.1.3. Let A be a σ-structure. For a subset of the universe $V' \subseteq V(A)$, the *substructure* $A[V']$ *of* A *induced by* V' is the structure with universe $V(A[V']) = V'$ and for each k-ary relation symbol $R \in \sigma$ interpretation $R(A[X]) = R(A) \cap V'^k$.

It is also possible to reduce and expand a structure.

Definition 2.1.4. Let σ be a signature. For a subset $\sigma' \subseteq \sigma$, the σ'*-reduct* of a structure A is the σ'-structure A' with $V(A') = V(A)$ and $R'(A') = R'(A)$ for all $R' \in \sigma'$.

Whenever a σ'-structure A' is a σ'-restriction of a σ-structure A, we say that A is a σ*-expansion* of A'.

2.1.1 First-order logic

In this section we review the syntax and semantics of first-order logic.

The main notion here is that of a formula, and terms are the basic building blocks of formulas.

In the following we assume a given (countably infinite) set of variables, usually denoted by x, y or $x_1, x_2, \ldots$.

Definition 2.1.5. A σ*-term* is either a variable or a constant from σ.

Definition 2.1.6. A σ*-formula* is defined as follows.

- If R is an n-ary relation symbol from σ and $t, u, t_1, \ldots, t_n$ are terms, then $R(t_1, \ldots, t_n)$ and $t = u$ are σ*-formulas*. They are said to be *atomic*.
- If φ, ψ are σ-formulas and x is a variable, then $\neg\varphi$, $\varphi \wedge \psi$ and $\exists x\varphi$ are σ-formulas.

We also define the following abbreviations:

Definition 2.1.7. Let φ and ψ be σ-formulas. Then we define:

- $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$;
- $\varphi \to \psi := \neg\varphi \vee \psi$;
- $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$;
- $\forall x\varphi := \neg\exists x\neg\varphi$;
- $\top := \forall x(x = x)$
- $\bot := \neg\top$.

Definition 2.1.8. The *subformulas* of a formula φ are defined as follows.

- If φ is atomic, then φ is the only subformula.
- If φ is of the form $\neg\psi$ or $\exists x\psi$, then φ and the subformulas of ψ are subformulas of φ
- If φ is of the form $\psi_1 \wedge \psi_2$, then φ and the subformulas of ψ_1 and ψ_2 are subformulas of φ.

Definition 2.1.9. An occurrence of a variable x is *bound* if it is in a subformula of the form $\exists x\varphi$. Otherwise the occurrence is *free*. The *free variables* of a formula are the variables with free occurrences. A formula with no free variables is called a *sentence*. A *theory* is a set of sentences.

Definition 2.1.10. The *quantifier rank* of a formula φ is the nesting depth of quantifiers in φ.

2.1.2 Second-order logic

Second-order logic is an extension of first-order logic that also allows quantification over relations. In addition to variables x that can range over elements of the universe, we also assume the existence of a given (countably infinite) set of variables R that can range over relations.

The fragment of second-order logic that only allows quantification over unary relations, i.e. sets, is called *monadic second-order logic* (or just *MSO*).

Following the convention in Flum and Grohe's book [72], we call a second-order formula where all relation variables occur freely a *first-order formula with free relation variables*.

Next we define some more useful notation.

Definition 2.1.11. Let A be a structure. Let $\varphi(X)$ be a first-order formula with exactly one free relation variable X of arity s. We say that a set $S \subseteq A^s$ is a *solution* for φ if $A \models \varphi(S)$.

In the following we (inductively) define two families of classes of formulas Σ_i and Π_i that are needed to define the parameterized complexity classes $\mathsf{W}[i]$ in Section 2.2.3.

Definition 2.1.12. We let Σ_0 and Π_0 both denote the class of all quantifier-free formulas.

For $i > 0$, we let Σ_i be the class of all second-order formulas of the form

$$\exists x_1 \dots \exists x_k.\psi$$

for all $k \in \mathbb{N}$ and $\psi \in \Pi_{i-1}$, and let Π_i be the class of all second-order formulas of the form

$$\forall x_1 \dots \forall x_k.\psi$$

for all $k \in \mathbb{N}$ and $\psi \in \Sigma_{i-1}$.

2.2 Parameterized complexity theory

Parameterized complexity theory is a two-dimensional approach to the study of the complexity of computational problems. A thorough discussion of the subject can be found in the books by Downey and Fellows [56], Flum and Grohe [72] and Niedermeier [135]. We follow notation and terminology from [72].

Here we introduce the basic definitions of the theory. In Section 2.2.4 we also give non-standard definitions of parameterized problems and fixed-parameter tractability based on model theoretic notions. These are more convenient for our study of slicewise nowhere dense and slicewise first-order definable problems in Chapter 8.

The notions of language and problem are standard notions within complexity theory.

Definition 2.2.1. Let Σ be a finite alphabet. A *language* (or *problem*) L is a set of strings $L \subseteq \Sigma^*$.

The main addition of parameterized complexity theory is the association of an additional *parameter* with each problem instance.

Definition 2.2.2. Let Σ be a finite alphabet. A *parameterization* is a computable function $\kappa : \Sigma^* \to \mathbb{N}$. For a parameterization κ and a problem $L \subseteq \Sigma^*$, we say that (L, κ) is a *parameterized problem over* Σ.

2.2.1 Fixed-parameter tractability

The central notion of parameterized complexity is that of *fixed-parameter tractability*, which captures the idea of tractability within the framework. If we can decide a problem in polynomial time and the degree of the polynomial is fixed and independent of the parameter, then the problem is fixed-parameter tractable (even though there might be a factor of an arbitrary computable function of the parameter involved).

Definition 2.2.3. Let Σ be a finite alphabet. We say that L is *fixed-parameter tractable* with respect to a parameterization κ if we can decide whether an input $x \in \Sigma^*$ is in L in time $O(f(\kappa(x)) \cdot |x|^c)$, where c is a constant and f is some computable function. The class of all fixed-parameter tractable problems is called FPT.

Remark. Although FPT is a class of problems, the convention is to say a problem is FPT instead of saying it is **in** FPT. If the algorithm decides the problem in time $O(f(\kappa(x)) \cdot |x|)$, we say that the problem is (in) *linear* FPT.

Remark. To emphasise the parameter dependence, we sometimes write $O^*(f(k))$ instead of $O(f(k) \cdot n^c)$, for some computable function f and constant c.

2.2.2 Kernelization

Central to the study of parameterized complexity of computational problems is an alternative characterisation of fixed-parameter tractability called *kernelization*. Kernelization captures the idea of pre-computation: a problem has a kernel if we can reduce the problem instance to a smaller one (i.e. bounded in size by a function of the parameter).

Definition 2.2.4. Let Σ be a finite alphabet and let (L, κ) be a parameterized problem over Σ. A polynomial-time computable function $K : \Sigma^* \to \Sigma^*$ is a *kernelization* of (L, κ) if there exists a computable function $h : \mathbb{N} \to \mathbb{N}$ such that for all $x \in \Sigma^*$ we have that

- $x \in L \iff K(x) \in L$; and
- $|K(x)| \leq h(\kappa(x))$.

The instance $K(x)$ is called the *kernel* of $x \in \Sigma^*$.

It is straightforward to show that a parameterized problem admits a kernelization if, and only if, it is fixed-parameter tractable. (See [72, Theorem 1.39] for a proof.)

Theorem 2.2.5. *A decidable parameterized problem (L, κ) over a finite alphabet Σ is fixed-parameter tractable if, and only if, it has a kernelization.*

Kernelizations give a different perspective on fixed-parameter tractability. We can obtain a more fine-grained understanding of the complexity of a problem by considering the size of the kernel. A large amount of current research is dedicated to showing not only the existence of a kernelization for a specific problem, but to showing the existence of a small kernelization, i.e. a kernelization that has its size bounded by a polynomial function of the parameter, or even a linear one.

2.2.3 The W[i] hierarchy and the class XP

Parameterized complexity theory also has a notion of intractability. In classical complexity theory we consider a problem to be intractable if it is not contained in the class P. However, we cannot prove this for problems, and the usual way to demonstrate intractability is by showing that a problem is NP-hard. Most complexity theorists believe that P $\neq$ NP, so it is generally thought to be unlikely that an NP-hard problem is contained in P.

The situation in parameterized complexity mirrors that in classical complexity theory. The notion of tractability in parameterized complexity is that of fixed-parameter tractability and FPT is the class of such problems. There is also a notion of intractability that plays a role comparable to NP. However, there is not a single class, but a whole hierarchy: the classes W[i] for $i > 0$. On top of those classes is the class XP that contains all of them. Sometimes just the first of these classes W[1] is considered to be the parameterized version of NP.

The W[i] classes were originally defined by Downey and Fellows [56] using notions from circuit complexity; each class is based on a circuit satisfiability problem. We give an equivalent definition here in terms of logic.

At the core is the Weighted Fagin Definability problem for φ defined as follows. Let $\varphi(X)$ be a formula with one free relation variable X of arity s.

Weighted Fagin Definability Problem for φ (WD$_\varphi$)
Input: A structure A and a non-negative integer k
Parameter: k
Problem: Is there a solution $S \subseteq A^s$ for φ of cardinality $\lvert S\rvert = k$?

Then the W-hierarchy is defined as follows. Recall from Definition 2.1.11 the notion of a solution of a formula, and from Definition 2.1.12 the class of formulas Π_i. We write $[\cdot]^{\text{fpt}}$ to denote the closure of a class under FPT-reductions.

Definition 2.2.6. For every $i > 0$ we define

$$\mathsf{W}[i] := [\{\text{WD}_\varphi \mid \varphi(X) \in \Pi_t\}]^{\text{fpt}}$$

Next we define the class XP that subsumes all the previously defined ones. It is sometimes compared to the class PSPACE in classical complexity theory, although it is not defined in terms of space resources. The class XP contains all the parameterized problems where the growth of the polynomial is allowed to depend on the parameter.

Definition 2.2.7. Let Σ be a finite alphabet. The class of parameterized problems XP contains all the parameterized problems L with parameterization κ that can be decided in time $O(n^{f(\kappa(x))})$ for some computable function f.

Without proof we state the following inclusions between the classes defined here (see e.g. [72, Chapter 7]):

$$\mathsf{FPT} \subseteq \mathsf{W}[x1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{XP}$$

2.2.4 Model theoretic definitions

We find it convenient in same parts of the dissertation to not define parameterized problems as classes of strings, but as classes of structures. This avoids the need to deal with particularities of the string encoding. The idea of defining problems this way is inspired by Dawar and He [51]. We introduce the definitions in this section.

We always explicitly mention when we refer to these non-standard definitions instead of the ones given above.

Definition 2.2.8. Let σ be a signature. A *problem* $Q \subseteq \mathit{str}(\sigma)$ is an (isomorphism-closed) class of σ-structures.

Definition 2.2.9. Let σ be a signature. A *parameterization* is a computable function $\kappa : \mathit{str}(\sigma) \to \mathbb{N}$.

Definition 2.2.10. Let σ be a signature and let Q be a problem. We say that Q is *fixed-parameter tractable* with respect to a parameterization κ if we can decide whether an input $A \in \mathit{str}(\sigma)$ is in Q in time $O(f(\kappa(A)) \cdot |A|^c)$, where c is a constant and f is some computable function.

2.3 Graph theory

Here we introduce the elementary notions of graph theory, graph minor theory, the isomorphism problem, and certain sparse graph classes.

A good introductory text to the topic is the book of Diestel [55], and for sparse graph classes the book of Nešetřil and Ossona de Mendez [133]. Our notation is mostly based on these books.

Definition 2.3.1. A *graph* G is a set of vertices $V(G)$ and a set of edges $E(G) \subseteq V(G) \times V(G)$. We write $|G|$ for the number of vertices $|V(G)|$ and $||G||$ for the cardinality of the edge set $|E(G)|$.

Remark. We usually assume that graphs are loop-free and undirected, i.e. that E is irreflexive and symmetric. If E is not symmetric, we call G a *directed graph*.

Remark. There is also an obvious connection to model theory discussed above: A graph is a relational structure with $\sigma = \{E\}$ – a signature that only contains one binary relation, the edge relation. If σ also contains unary relations the structure can be conceptualized as a coloured graph.

A basic notion of graph theory is that of a neighbour, and the number of neighbours – the degree.

Definition 2.3.2. Let G be a graph. The *neighbourhood* of a vertex v is $N_G(v) := \{w \in V(G) \mid vw \in E(G)\}$. For a set of vertices $S \subseteq V(G)$ its neighbourhood is defined to be $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$.

Definition 2.3.3. Let G be a graph. If $v \in G$ and $S \subseteq V(G)$, we write $E_G(v, S)$ for the set of edges $\{vw \mid w \in S\}$ between v and S.

Definition 2.3.4. Let G be a graph. The *degree* of a vertex $v \in V(G)$ is the size of its neighbourhood $\deg_G(v) := |N_G(v)|$. If it is clear from the context what the graph is, we may omit the subscript. The *maximum degree* of a graph G is the maximum degree of its vertices $\Delta(G) := \max\{\deg_G(v) \mid v \in V(G)\}$.

Just as for structures in general, we can think of induced subgraphs as induced substructures. We also define the more general notion of a subgraph that might inherit only some of the edges from the original graph.

Definition 2.3.5. Let G be a graph. A *subgraph* H of G is a graph with vertices $V(H) \subseteq V(G)$ and edges $E(H) \subseteq (V(H) \times V(H)) \cap E(G)$. If $A \subseteq V(G)$ is a set of vertices of G, we write $G[A]$ for the subgraph *induced* by A, i.e. $V(G[A]) = A$ and $E(G[A]) = E(G) \cap (A \times A)$.

If A is a subset of $V(G)$, we write $G \setminus A$ for $G[V(G) \setminus A]$. For a vertex $v \in V(G)$, we write $G \setminus v$ for $G \setminus \{v\}$.

Graphs can be thought of as networks, where the vertices are connected by strings of edges – so it is natural to define reachability, paths and connectedness.

Definition 2.3.6. Let G be a graph. A vertex $v \in V(G)$ is said to be *reachable* from a vertex w in G if $v = w$ or if there is a sequence of edges $a_1a_2, \ldots, a_{s-1}a_s \in E(V)$ with the a_i pairwise distinct and $w = a_1$ and $v = a_s$. We call the subgraph P of G with vertices $V(P) = \{a_1, \ldots, a_s\}$ and edges $E(P) = \{a_1a_2, \ldots, a_{s-1}a_s\}$ a *path from w to v*.

Definition 2.3.7. Let H be a subgraph of a graph G and $v, w \in V(G)$. A *path through H from w to v* is a path P from w to v in G with all vertices, except possibly the endpoints, in $V(H)$, i.e. $(V(P) \setminus \{v, w\}) \subseteq V(H)$.

This notion also gives rise to an extended definition of neighbourhood.

Definition 2.3.8. Let G be a graph. The *r-neighbourhood* of a vertex $v \in V(G)$ is

$$N_r(v) := \{w \in V(G) \mid \text{there is a path from } v \text{ to } w \text{ of length at most } r\}.$$

It is easy to see that for undirected graphs reachability defines an equivalence relation on the vertices of G.

Definition 2.3.9. A subgraph of an undirected graph induced by a reachability class is called a *(connected) component*.

Definition 2.3.10. Let $r \in \mathbb{N}$. An r-independent set in a graph G is a set of vertices of G such that their pairwise distance is at least r.

Definition 2.3.11. Let G be a connected graph. The *eccentricity* of a vertex $v \in V(G)$ is the maximum length of a shortest path between v and any other vertex $u \in V(G)$. The *radius* of G is the minimum eccentricity of all vertices in G. The *diameter* of G is the maximum eccentricity of all vertices in G.

2.3.1 Graph isomorphism

The graph isomorphism problem is the question whether two graphs are structurally equivalent, or more precisely, given two graphs if there is an edge relation preserving bijection between them. One of the major questions in complexity theory is whether it can be decided in polynomial time. A detailed discussion of the problem can be found in Chapter 6, which is dedicated to the problem.

We assume in the following that a graph G is given by its adjacency matrix, so that every vertex of G naturally corresponds to a number between 0 and $|G| - 1$.

Formally, the notion of isomorphic graphs is defined as follows:

Definition 2.3.12. Two graphs G, G' are *isomorphic* if there is a bijection $\varphi : V(G) \to V(G')$ such that for all $v, w \in V(G)$ we have that $vw \in E(G)$ if, and only if, $\varphi(v)\varphi(w) \in E(G')$. We write $G \cong G'$ if G and G' are isomorphic. We write GI to denote the problem of deciding, given G and G', whether $G \cong G'$.

Definition 2.3.13. A *(k-)colouring* of a graph G is a map $c : V(G) \to \{1, \dots, k\}$ for some $k \in \mathbb{N}$. We call a graph together with a colouring a *coloured* graph.

If no two adjacent vertices share the same colour, we say the colouring is proper.

Definition 2.3.14. A colouring $c : V(G) \to \{1, \dots, k\}$ of a graph G is a *proper (k-)colouring* if $c(u) \neq c(v)$ whenever $uv \in E(G)$.

Definition 2.3.15. Two coloured graphs G, G' with respective colourings $c : V(G) \to \{1, \dots, k\}, c' : V(G') \to \{1, \dots, k\}$ are *isomorphic* if there is a bijection $\varphi : V(G) \to V(G')$ such that:

- for all $v, w \in V(G)$ we have that $vw \in E(G)$ if, and only if, $\varphi(v)\varphi(w) \in E(G')$;
- for all $v \in V(G)$, we have that $c(v) = c'(\varphi(v))$.

Remark. Note that we require the colour classes to match exactly, and do not allow a permutation of the colour classes.

Definition 2.3.16. Let $\mathcal{C}$ be a class of (coloured) graphs closed under isomorphism. A *canonical form for* $\mathcal{C}$ is a function $F : \mathcal{C} \to \mathcal{C}$ such that

- for all $G \in \mathcal{C}$, we have that $F(G) \cong G$;
- for all $G, H \in \mathcal{C}$, we have that $G \cong H$ if, and only if, $F(G) = F(H)$.

For the following definition we need the notion of a pre-order, defined below in Definition 2.4.1. Note that we can define a lexicographical ordering on graphs via their adjacency matrices.

Definition 2.3.17. A canonical form F for a class $\mathcal{C}$ gives rise to a natural pre-order on $\mathcal{C}$, which we denote $\sqsubseteq_F$ whereby $G \sqsubseteq_F H$ just in case $F(G)$ is lexicographically smaller than $F(H)$. Then for any pair of graphs G and H in $\mathcal{C}$, at least one of $G \sqsubseteq_F H$ or $H \sqsubseteq_F G$ must hold, and both hold if, and only if, $G \cong H$.

2.3.2 Graph minor theory

A graph H is a *minor* of another graph G, if H can be obtained from G by repeatedly deleting vertices and edges, and by contracting[1] edges. Many of the important results in

[1] An edge uv is contracted by creating a new vertex w that shares the neighbours of both sides of the edge uv (i.e. $N(w) = N(u) \cup N(v)$) and then removing u and v from the graph.

the area were developed by Robertson and Seymour in a series of twenty papers. This section introduces the basic notions of graph minor theory.

Instead of the definition above, we define the notion of *minor* in an equivalent[2] way using the notion of a *minor map*. This definition makes it easier to talk about *shallow minors* later that only allow contractions in a limited radius.

Definition 2.3.18. A graph H is a *minor* of a graph G, written $H \preceq G$, if there is a map, called the *minor map*, that takes each vertex $v \in V(H)$ to a tree T_v that is a subgraph of G such that for any $u \neq v$ the trees are disjoint, i.e. $T_v \cap T_u = \emptyset$, and such that for every edge $uv \in E(H)$ there are vertices $u' \in T_u, v' \in T_v$ with $u'v' \in E(G)$.

Definition 2.3.19. A class of graphs $\mathcal{C}$ is *minor-closed* if $H \preceq G$ and $G \in \mathcal{C}$ implies $H \in \mathcal{C}$.

Definition 2.3.20. The *set of minimal excluded minors* $M(\mathcal{C})$ is the set of graphs in the complement of $\mathcal{C}$ such that for each $G \in M(\mathcal{C})$ all proper minors of G are in $\mathcal{C}$.

Robertson and Seymour [142] show that every minor-closed class is characterised by a finite set of 'forbidden minors'.

Theorem 2.3.21 (Robertson-Seymour Theorem [142])**.** *If a graph class $\mathcal{C}$ is minor-closed, then $M(\mathcal{C})$ is finite.*

Remark. It is an important consequence of this theorem that membership in a minor-closed class can be tested in polynomial time. This is because, as part of the proof, Robertson and Seymour [142] also show that we can test whether a graph is a minor of another graph in polynomial time.

Definition 2.3.22. For a set M of graphs, we write $\mathrm{Forb}(M)$ for the class of graphs which forbid M as minors, i.e. $\mathrm{Forb}(M) = \{G \mid H \not\preceq G \text{ for all } H \in M\}$.

2.3.3 Nowhere dense classes of graphs and classes with bounded expansion

In the following we use the language of minors to define nowhere dense classes of graphs. These are a very general notion of sparse graph classes that subsume many other sparse graph classes.

For that we need the notion of a shallow minor:

Definition 2.3.23. Let $r \in \mathbb{N}$. A minor H of G is a *shallow minor of G of depth r* or just a *depth-r minor of G*, written $H \preceq_r G$, if there is a minor map that takes vertices in H to trees that have radius at most r. We write $G\nabla r$ for the set of shallow minors of depth r of G.

Definition 2.3.24. A class of graphs $\mathcal{C}$ is *nowhere dense* if for every $r \in \mathbb{N}$ there is a graph H_r such that $H_r \not\preceq_r G$ for all $G \in \mathcal{C}$. A nowhere-dense class of graphs $\mathcal{C}$ is called *effectively nowhere dense* if there is a computable function h from integers to graphs such that if $G \in \mathcal{C}$ then for all r we have $h(r) \not\preceq_r G$.

[2] See [55, Proposition 1.7.1] for a proof.

Remark. We are only interested in effectively nowhere-dense classes so we simply use the term *nowhere dense* to mean effectively nowhere dense.

Dawar [47, 49] introduced the notion of quasi-wide graph classes that was proved by Nešetřil and Ossona de Mendez [133] to be equivalent to that of nowhere-dense classes of graphs closed under induced subgraphs.

Definition 2.3.25. Let G be a graph and let $r \geq 0$ be an integer. Then a set $A \subseteq V(G)$ is *r-scattered* if for all $u \neq v \in A$, we have $N_r(u) \cap N_r(v) = \emptyset$.

Definition 2.3.26. Let $s : \mathbb{N} \to \mathbb{N}$ be a function. A class $\mathcal{C}$ of graphs is *quasi-wide with margin s* if for all $r \geq 0$ and $m \geq 0$ there exists an $N \geq 0$ such that if $G \in \mathcal{C}$ and $|G| > N$ then there is a set $S \subseteq V(G)$ with $|S| < s(r)$ such that $G \setminus S$ contains an r-scattered set of size at least m.

Theorem 2.3.27 (Nešetřil and Ossona de Mendez [133])**.** *Let $\mathcal{C}$ be a class of graphs closed under taking induced subgraphs. Then $\mathcal{C}$ is nowhere dense if, and only if, it is quasi-wide.*

Nešetřil and Ossona de Mendez also introduce a characterisation of nowhere dense graph classes using limits on the ratio of the number of edges to the number of vertices. They prove the following trichotomy:

Definition 2.3.28. Let $\mathcal{C}$ be a class of graphs and let $a > 0$. Then

$$\mathcal{C}\nabla a := \bigcup_{G \in \mathcal{C}} G\nabla a.$$

Definition 2.3.29. Let $\mathcal{C}$ be a class of graphs. Then

$$\overline{\ell\mathrm{dens}}(\mathcal{C}^{\tilde{\nabla}}) := \sup_{i \to \infty} \limsup_{G \in \mathcal{C}\tilde{\nabla} i} \frac{\log ||G||}{\log |G|}.$$

Note that if none of the graphs in an infinite class of graphs $\mathcal{C}$ contain an edge, $\overline{\ell\mathrm{dens}}(\mathcal{C}^{\tilde{\nabla}}) = -\infty$. If we exclude this (rather uninteresting) case, we obtain the following trichotomy:

Theorem 2.3.30 (Class Trichotomy, Nešetřil and Ossona de Mendez [133])**.** *Let $\mathcal{C}$ be an infinite class of graphs such that at least one graph in $\mathcal{C}$ contains an edge. Then $\overline{\ell dens}(\mathcal{C}^{\tilde{\nabla}})$ can only take the values $0, 1$ or 2. Moreover*

- *$\mathcal{C}$ is of bounded size (or asymptotically edgeless) if, and only if, $\overline{\ell dens}(\mathcal{C}^{\tilde{\nabla}}) = 0$;*
- *$\mathcal{C}$ is nowhere dense if, and only if, $\overline{\ell dens}(\mathcal{C}^{\tilde{\nabla}}) \in \{0, 1\}$;*

Remark. In the third case, i.e. when $\overline{\ell\mathrm{dens}}(\mathcal{C}^{\tilde{\nabla}}) = 2$, we say that $\mathcal{C}$ is *somewhere dense*.

This also allows us to define a special case where the ratio $||G||/|G|$ is bounded by a constant $c(t)$ for every $G \in \mathcal{C}\nabla t$, calling these kinds of classes, classes with *bounded expansion*.

Definition 2.3.31. Let $\mathcal{C}$ be a class of graphs. Then $\mathcal{C}$ has *bounded expansion* if for every $t > 0$ there is $c(t)$ such that $\frac{||G||}{|G|} < c(t)$ for every $G \in \mathcal{C}\nabla t$.

There is an interesting connection between these classes and the parameter tree-depth (cf. Definition 4.0.4) introduced below. Both nowhere-dense classes and bounded expansion classes can be characterised using colourings where each colour induces a graph with low tree-depth.

Definition 2.3.32. Let G be a graph and $p \in \mathbb{N}$. Then we denote by $\chi_p(G)$ the least number of colours required to properly colour G in such a way that every set of colours S with $|S| \leq p$ induces a subgraph of H with tree-depth at most $|S|$. We call such a colouring a *low tree-depth colouring*.

Theorem 2.3.33 (Nešetřil and Ossona de Mendez [131, 133])**.** *Let $\mathcal{C}$ be a class of graphs. Then*

- *$\mathcal{C}$ has bounded expansion if, and only if, for every integer p there is an integer N_p such that for any graph $G \in C$ we have $\chi_p(G) \leq N_p$.*
- *$\mathcal{C}$ is nowhere dense if, and only if, for every $\epsilon > 0$ we can find sufficiently large p, N_ϵ such that for all $G \in \mathcal{C}$ with $|G| > N_\epsilon$ we have $\chi_p(G) \leq |G|^\epsilon$.*

2.4 Order theory

When we talk about graphs, it is often easier to think about orders on the vertices instead of defining additional structures. In this section we introduce some elementary notions of order theory, leading to the new notion of an *elimination order*. This notion is used to establish an alternative characterisation of *tree-depth* in Chapter 4, and is crucial for the study of *elimination distance* in Chapter 5.

A *pre-order* captures important aspects of the intuitive idea of an ordering of a set.

Definition 2.4.1. A *pre-order* (or *quasiorder*) is a binary relation $\leq$ on a set S which is reflexive and transitive.

Remark. For two elements a, b in a set S partially ordered by $\leq$, we write $a < b$ as an abbreviation for $a \neq b$ and $a \leq b$.

The notion of *partial order* is a formalisation of the intuitive idea of an ordering of a set. It is like a pre-order, but is also antisymmetric.

Definition 2.4.2. A *partial order* is a binary relation $\leq$ on a set S which is reflexive, antisymmetric and transitive.

In a partial order there is no requirement that any two elements have to be comparable. If we require this, an order is called *total*:

Definition 2.4.3. A partial order $\leq$ on S is a *total order* (or *linear order*) if either $a \leq b$ or $b \leq a$ for all $a, b \in S$.

If all the elements of a subset of a partially ordered set are comparable, it is called a *chain*.

Definition 2.4.4. Let S be a set and let $\leq$ be a partial order on S. We say a set $T \subseteq S$ is a *chain* if it is *totally ordered* by $\leq$.

This allows us to introduce a notion of height of an order.

Definition 2.4.5. Let S be a set and let $\leq$ be a partial order on S. We say that $\leq$ has *height* k if the length of the longest chain in it is k.

Sometimes we want to restrict the order relation of a partially ordered set, so that it only holds between comparable elements that are immediate neighbours (i.e. distinct without an intermediate element). For example, in the case of a linear order, we might only be interested in the successor relation. This is called a *covering relation*.

Definition 2.4.6. Let S be a set and let $\leq$ be a partial order on S. For two elements $a, b \in S$ we say b *covers* a if $a < b$ and there is no $c \in S$ such that $a < c < b$. This relation is called the *covering relation of* $\leq$.

Note that if we require that for each element the predecessors form a chain, we obtain a partial order with a covering relation that is a directed tree. This is because any two predecessors must be comparable, so there can only be one path that contains both of them.

Definition 2.4.7. If $\leq$ is a partial order on S, and for each element $a \in S$, the set $\{b \in S \mid b \leq a\}$ is a chain, we say $\leq$ is a *tree order*.

Remark. Note that the covering relation of a tree order is not necessarily a tree, but may be a forest.

Analogue to trees we define the level of a vertex in a tree order.

Definition 2.4.8. Let G be a graph and let $\leq$ be a tree order for G. The *level* of a vertex $v \in V(G)$ is the length of the chain $\{w \in V(G) \mid w \leq v\}$. We denote the level of v by $level_{<}(v)$.

If we consider the tree-orderings of the vertices of a graph that additionally require that adjacent vertices are comparable we obtain a notion similar to that of a tree-depth decomposition (defined below in Definition 4.2.1).

Definition 2.4.9. Let G be a graph. An *elimination order* $\leq$ of G is a tree order on the vertices of G, such that for each edge $uv \in E(G)$ we have either $u \leq v$ or $v \leq u$.

Remark. We show below in Proposition 4.2.3 that this gives us an order-theoretic characterisation of tree-depth (introduced in Chapter 4): there is an elimination order $\leq$ of height k for a graph G if, and only if, $td(G) \leq k$.

CHAPTER 3

Distance parameters for graph problems

This chapter introduces several graph parameters. Graph parameters are numeric properties of graphs that are used in the study of the parameterized complexity of graph problems.

The graph parameters introduced in this section are all of a certain kind. We say that a graph parameter is a *distance parameter* if it measures how far a graph is from being contained in some class of graphs. This is one of the central notions of this dissertation and all of the parameters discussed in this section are distance parameters. Figure 1.1 demonstrates the relationships between the distance parameters introduced here (except for clique-width).

That a graph parameter is a measure of distance is not always immediately apparent from its definition. It is obvious for a parameter such as *deletion distance to a class* $\mathcal{C}$, as it is just the number of vertices we need to remove to place a graph in a certain class $\mathcal{C}$, but also applies to parameters such as *tree-width*, which measures how far a graph is from being a tree. We make clear for each parameter discussed in this chapter why we consider it to be a notion of distance.

The distance parameters we are concerned with in this dissertation measure distances to classes of *sparse* graphs. We do not define sparsity precisely, but intuitively it means that the graphs in such classes do not contain many edges. For example the class of trees is a a sparse graph class, and the notion of nowhere dense graph classes (see Definition 2.3.24) defines a very general notion of sparse graph classes. Note that sparsity is a property of graph classes, not single graphs, and a graph class can still be sparse even if it contains some dense graphs (e.g. complete graphs). We will use the terms *sparse graph class* and *class of sparse graphs* interchangeably.

Distance parameters are widely studied by researchers in parameterized complexity theory. The main goal of that research is to study different parameterizations of a problem to gain a better understanding of the problem's hardness through a refined analysis of the different parameters, and subsequently identify parameterizations that lead to fixed-parameter tractability. Recall from our discussion in the introduction that there are generally two motivations for studying a parameter in parameterized complexity: the parameter itself can be the focus of interest, or the parameter is mainly a tool to better understand a problem. The study of distance parameters allows both: Consider as an

example the problem Vertex Cover: given a graph G and an integer k, the question is whether G has a vertex cover (defined below in Definition 3.2.1) of size at most k. We see below that the vertex cover number, i.e. the size of a minimum vertex cover in a graph G is just the vertex-deletion distance of G to the class of edge-less graphs.

Distance parameters pose an interesting problem by themselves: determining the distance of an input graph G to a class $\mathcal{C}$ is, in general, a computationally challenging problem in its own right. For example, the Vertex Cover problem is NP-hard and often studied with the vertex cover number of the graph as parameter.

On the other hand, the parameter can be used as a way to better understand what kind of restrictions make other problems tractable. A canonical example is again Vertex Cover: a popular parameterization in the study of many problems is the vertex cover number of a graph.

Distance parameters suggest a certain strategy to obtain fixed-parameter tractability results. If a problem one wishes to solve is tractable on the class $\mathcal{C}$, then distances to $\mathcal{C}$ provide interesting parameterizations of that problem. Guo et al. [95] recognized this as a pattern: when studying the complexity of a parameterized problem, often fixed-parameter tractability results are obtained by considering a class of graphs $\mathcal{C}$ where the problem is tractable, i.e. there is a known polynomial-time algorithm that solves the problem restricted to $\mathcal{C}$, and parameterizing the problem by some notion of distance to that class $\mathcal{C}$. They coined the term 'distance to triviality' for this and suggested it as a general strategy, giving several examples to demonstrate its fruitfulness.

We only discuss parameters in this chapter that are used in one or more places throughout the dissertation. There are many other interesting distance parameters, such as the crossing number of a graph, which is a notion of distance to the class of planar graphs, to give just one example of a distance parameter not mentioned.

Note that we defer the introduction of the distance parameter *tree-depth* to Chapter 4, which is dedicated to it. Tree-depth receives special attention in this dissertation, because of its relationship to Chapter 5, where we introduce the new graph parameter *elimination distance to a class* $\mathcal{C}$. Elimination distance naturally generalises the notion of deletion distance, with tree-depth being a special case. Tree-depth also has importance in Chapter 8, where we apply the method developed in that chapter to establish that tree-depth is FPT.

3.1 Deletion distance to a class $\mathcal{C}$

A popular template for graph parameters is the *deletion distance* to some class of graphs $\mathcal{C}$.

Definition 3.1.1. A graph G has *deletion distance k to a class $\mathcal{C}$* if there are k vertices $v_1, \ldots, v_k \in V(G)$ such that $G \setminus \{v_1, \ldots, v_k\} \in \mathcal{C}$.

The study of the parameterized complexity of deletion distances was motivated by a powerful hardness result: Lewis and Yannakakis [112] show that vertex deletion to a class $\mathcal{C}$ is NP-hard on graph classes $\mathcal{C}$ that are non-trivial and closed under induced subgraphs. This means that it is unlikely that we will discover efficient algorithms that check whether a graph has a certain distance on the class of all graphs.

The search for fixed-parameter tractability results for deletion distances was initiated by Cai [32], and all kinds of vertex deletion problems are now being studied in that

context. Deletion distances have many applications and are now among the most studied problems in the world of parameterized complexity.

A large number of parameterized parameters are either directly deletion distance problems, or can be rephrased in that way. For example Cluster Vertex Deletion (deletion distance to the class of disjoint unions of cliques), Graph Bipartization (deletion distance to the class of bipartite graphs), and Feedback Vertex Set (deletion distance to the class of forests) are all problems where the natural parameterization (by the output size) is a distance to a graph class.

Deletion distance is used in Section 8.3.1 (where we show that if $\mathcal{C}$ is nowhere dense and first-order definable, then the problem of determining the deletion distance to $\mathcal{C}$ is slicewise first-order definable and slicewise nowhere dense and thus FPT), and Section 6.2 (where we show that Graph Isomorphism is FPT parameterized by the deletion distance to bounded degree graphs).

3.2 Vertex cover number

A vertex cover of a graph G is a set of vertices $A \subseteq V(G)$ so that each edge of G is incident to at least one vertex in A. As mentioned above, the problem Vertex Cover is the problem of deciding whether there is a vertex cover of a given graph of at most a given size. Determining the vertex cover number of a graph is a classical optimization problem, and deciding whether there is a vertex cover of at most a given size is NP-complete – in fact, it is on the list of Karp's [103] original NP-complete problems.

Definition 3.2.1. Let G be a graph. A *vertex cover* of G is a set of vertices $S \subseteq V(G)$ such that for all edges $uv \in E(G)$ either $u \in S$ or $v \in S$.

The size of a minimal vertex cover is a popular parameter called the *vertex cover number*.

Definition 3.2.2. Let G be a graph. The *vertex cover number* of G, written $vc(G)$, is the size of a minimal vertex cover of G.

Remark. Note that the vertex cover number is in fact a deletion distance – it is just the deletion distance to the class of edgeless graphs.

The vertex cover number is possibly the most popular parameter studied in parameterized complexity theory and there has been a long sequence of papers proving the existence of smaller and smaller kernelizations for Vertex Cover.

3.3 Edit distance to a class $\mathcal{C}$

Instead of just removing vertices from a graph, we can also consider more general editing operations on a graph; e.g. removing or adding an edge.

In the following we abbreviate the editing operations vertex deletion, edge deletion, vertex addition and edge addition as d_v, d_e, a_v and a_e respectively. Then for each non-empty $O \subseteq \{d_v, d_e, a_v, a_e\}$ we define the O-edit-distance as follows:

Definition 3.3.1. A graph G has *O-edit-distance k to a class $\mathcal{C}$* if we can obtain a graph $G' \in \mathcal{C}$ from G using k editing operations from O.

Many graph editing problems are hard in general. In addition to the result by Lewis and Yannakakis [112], discussed above in Section 3.1 who show that many graph deletion problems are NP-hard, Yannakakis [159] also shows that this holds for edge deletion problems on many graph classes. This hardness motivates the study of the parameterized complexity of graph editing problems.

Just as for deletion distance, many problems of this form are studied in parameterized complexity. For example Cluster Editing (edit distance to the class of disjoint unions of cliques), Twin Graph Editing (edit distance to the class of twin graphs) and Chordal Editing (edit distance to the class of chordal graphs) are generalisations of the versions mentioned above in Section 3.1.

In Section 8.3.2 we demonstrate how our result from Chapter 8 can be applied when $\mathcal{C}$ is a class of (sparse) graphs defined by degree constraints.

3.4 Path-width & tree-width

Tree-width is one of the most studied graph parameters in parameterized complexity. The concept was initially introduced under the name of dimension by by Bertel and Brioschi [15]. After being rediscovered by Halin [97] it was again rediscovered and popularized by Robertson and Seymour [141] in the context of graph minor theory.

At the core of the definition is a notion of a tree decomposition of a graph, which makes it fairly easy to apply dynamic programming techniques. Path decompositions are a special case of tree decompositions, where the decomposition is a not just a tree, but a path.

Definition 3.4.1. Let G be a graph. A *tree decomposition* of G is a pair $(T, \mathcal{B})$ where T is a tree with nodes $\mathcal{B}$, called the *bags* of the tree decomposition. Each node in $\mathcal{B}$ is a set of vertices of G that satisfy the following:

- $V(G) = \bigcup_{B \in \mathcal{B}} B$;
- for each $uv \in E(G)$ there is a $B \in \mathcal{B}$ such that $u, v \in B$;
- for each $v \in V(G)$, the subtree induced by $\{B \in \mathcal{B} \mid v \in B\}$ is connected.

If T is a path, then we call $(T, \mathcal{B})$ a *path decomposition*.

The *width* of a path or tree decomposition is one less than the size of the largest bag, i.e.

$$\max_{B \in \mathcal{B}} |B| - 1.$$

Definition 3.4.2. The *tree-width* of a graph G, abbreviated as $tw(G)$, is the minimal width of a tree decomposition of G.

Definition 3.4.3. The *path-width* of a graph G, abbreviated as $pw(G)$, is the minimal width of a path decomposition of G.

Remark. Since every path decomposition is a tree decomposition, we have that the tree-width of a graph is bounded above by its path-width.

Tree-width (and path-width) are lower bounds for vertex cover number.

Proposition 3.4.4. *Let G be a graph with vertex cover number k. Then G has path-width and tree-width at most k.*

Proof. Let A be a vertex cover of size k of G. For each $v \in V(G) \setminus A$ define the set

$$B_v := A \cup \{v\}.$$

Let $\mathcal{B} := \{B_v \mid v \in V(G) \setminus A\}$. Let T be a path with node set $\mathcal{B}$, in any order. Then $(T, \mathcal{B})$ is a path decomposition of G (and thus also a tree decomposition).

Note that all the bags have size $k+1$, thus the path-width (and thus also tree-width) of G is at most k. □

We study a problem (the ANCHORED k-CORE problem) parameterized by tree-width in Chapter 9.

3.5 Clique-width

Clique-width was introduced as a graph parameter by Courcelle and Olariu [43] to measure the complexity of graphs in terms of hierarchical decompositions. The clique-width of a graph is defined as the smallest amount of colours needed to construct the graph with a certain set of rules.

To formally define clique-width, we first need to introduce some notation.

Definition 3.5.1. Let $k > 0$. A graph G is *k-constructible* if there is a colouring $\chi : V(G) \to \{1, \ldots, k\}$ of G such that (G, χ) can be constructed using the following operations:

- *introduce*: Create a new graph H with a single vertex v and colouring χ_H such that $\chi_H(v) = i$ for some $1 \leq i \leq k$. The one-vertex graph with vertex v and colour i is denoted $i(v)$.
- *disjoint union*: Take the disjoint union of two graphs.
- *join*: Add an edge between every vertex of colour i and every vertex of colour j for some $1 \leq i, j \leq k$.
- *recolour:* Colour all vertices of i with colour j for some $1 \leq i, j \leq k$.

Remark. These operations can be defined in algebraic terms, so that there is an algebraic expression that constructs a k-constructible graph (with a colouring).

Definition 3.5.2. Let G be a graph. The *clique-width* of G is the smallest integer k such that G is k-constructible.

Clique-width is a distance parameter in the sense that it measures the number of colours needed to construct a graph from the class containing only the empty graph.

Although it is certainly a distance to a (very) sparse graph class in this sense, graph classes with bounded clique-width can contain dense graphs as well, e.g. it is easy to see that every complete graph has clique-width 1. So clique-width does not give rise to sparse graph classes, i.e. the class of graphs with clique-width at most k is not a sparse graph class for $k > 0$ and for example contains all cliques. However, Gurski and Wanke [96] show that clique-width can in a certain way be thought of as a generalisation of tree-width; a class of graphs of bounded clique-width that is also sparse must have bounded tree-width:

Theorem 3.5.3 ([96]). *Let G be a graph with clique-width k, and let $n > 1$. If G does not contain the complete bipartite graph on n vertices $K_{n,n}$ as a subgraph, then G has tree-width at most $3k(n-1)-1$.*

Moreover, Courcelle [40] showed the following:

Theorem 3.5.4 (Courcelle's Theorem). *Every problem that can be expressed in monadic second-order logic has a linear time algorithm on graphs of bounded tree-width.*

Later the theorem was generalised to clique-width by Courcelle et al. [42]:

Theorem 3.5.5. *Every problem that can be expressed in monadic second-order logic has a linear time algorithm on graphs of bounded clique-width.*

As the clique width of a graph is bounded in terms of its rank width, the decomposition can be found via an algorithm for rank-width.

We use this generalisation of Courcelle's Theorem in Chapter 9, where we study a problem (the Anchored k-core problem) parameterized by clique-width.

3.6 Max leaf number

The max leaf number of a connected graph G is just the maximum number of leaves in a spanning tree for G. (We give the relevant definitions below.) Fellows et al. [66, 68] have initiated its study as a parameter in the context of parameterized complexity.

The max leaf number of a graph is formally defined as follows:

Definition 3.6.1. Let G be a connected graph. A *spanning tree of G* is a connected subgraph of G that is a tree.

Definition 3.6.2. The *max leaf number* of a connected graph G is the largest number of leaves in any of its spanning trees.

We can *subdivide* an edge uv in a graph G by removing the edge uv, adding a new vertex w and adding the edges uw and wv. A graph G' that is obtained from a graph G by subdividing edges is called a *subdivision of G*. Kleitman and West [106] show that if a graph G has max leaf number k, then it is a subdivision of a graph with at most $4k-2$ vertices, in particular all vertices of G, except for at most $4k-2$, must have degree 2. The max leaf number therefore gives us a deletion distance to the (very sparse) graph class of degree-2 graphs, that is, paths and cycles. It is thus a distance to a sparse graph class.

The above also demonstrates that if a graph has max leaf number k it can have tree-width at most $4k-3$, because subdividing edges does not affect the tree-width of a graph. So both bounded vertex cover number and bounded max leaf number imply bounded tree-width. However, neither of them implies a bound on the other parameter: A star graph has a vertex cover of size 1, while it can have unbounded max leaf number. A path has max leaf number 2, while it can have unbounded vertex cover number.

We introduce the max leaf number of a graph in this section because the parameter *generalised tree-depth*, discussed in Section 4.4, is explicitly designed to be a common generalisation (i.e. a parameter that gives a lower bound) for both max leaf number and tree-depth.

3.7 Degeneracy and the k-core of a graph

Degeneracy is yet another measure for how sparse a graph is. It was first introduced under the name *colouring number* by Erdős and Hajnal [59] as the least number k for which there is an ordering of the vertices such that every vertex has at most k neighbours that come before it in the ordering. Lick and White [114] gave an equivalent definition under the name *degeneracy*. They defined a graph to be k-degenerate for some integer k if every subgraph contains a vertex with at most k neighbours. This is the more common definition and the one we will use.

Definition 3.7.1. Let $k \geq 0$. A graph G is *k-degenerate* if every subgraph of G contains a vertex of degree at most k. We say that the least k for which G is k-degenerate is the *degeneracy* of G.

The degeneracy of a graph can be interpreted as a distance parameter in the following way: since every subgraph of a graph G with degeneracy k contains a vertex of degree at most k, we can, starting with the whole graph G, keep isolating vertices by deleting at most k edges from the graph. Thus we can isolate all the vertices with at most $k|G|$ edge deletions, and the graph has thus an edge deletion distance (or $\{d_e\}$-edit-distance) of at most $k|G|$ to the class of edgeless graphs.

Bodlaender et al. [19] show that the degeneracy of a graph is a lower bound for its tree-width:

Lemma 3.7.2 (Bodlaender et al. [19], Lemma 3)**.** *Let G be a graph. Then the degeneracy of G is at most the tree-width of G.*

The degeneracy of a graph is also called its k-core number for the following reason: The k-core of a graph G is what is left of a graph when we, repeatedly, remove all vertices from G that have less than k neighbours; or formally:

Definition 3.7.3. Let G be a graph and $k \geq 0$. The *k-core* of G is a maximal subgraph of G with minimum degree k. The largest k for which the k-core of G is non-empty is called the *k-core number of G*.

Remark. Note that there is one unique k-core. We obtain the k-core by repeatedly removing all vertices of degree less than k.

The k-core is connected to k-degeneracy in the following way: If the k-core is non-empty, then G must have degeneracy at least k. Conversely the degeneracy of G is the largest k for which G has a non-empty k-core, i.e. the k-core number of G.

The degeneracy (and k-core) of a graph will be used in Chapter 9 in the study of the Anchored k-core problem. A class of graphs of bounded degeneracy is itself a sparse graph class, and we will consider a notion of distance to such classes in Chapter 9.

chapter 4

Tree-depth: Exploring the space between vertex cover number and tree-width

This chapter has the dual purpose of showing why tree-depth is an interesting parameter, as well as being a preparation for the next chapter. In Chapter 5 we introduce the new parameter *elimination distance to a class* $\mathcal{C}$, which is a generalisation of tree-depth.

The *tree-depth* of a graph G is (non-recursively defined to be) the minimum height of a directed forest that contains G in its symmetric transitive closure. Formally, tree-depth is recursively defined as follows:

Definition 4.0.4 (Nešetřil and Ossona de Mendez [130])**.** Let G be a graph. We write $td(G)$ for the *tree-depth* of G, which is defined as follows:

$$td(G) := \begin{cases} 0, & \text{if } V(G) = \emptyset; \\ 1 + \min\{td(G \setminus v) \mid v \in V(G)\}, & \text{if } G \text{ is connected;} \\ \max\{td(H) \mid H \text{ a component of } G\}, & \text{otherwise.} \end{cases}$$

This chapter is organised as follows. After first giving some background in Section 4.1, we establish a number of equivalent characterisations in Section 4.2. Tree-depth is a very versatile parameter: the combinatorial definition in terms of a tree-depth decomposition (see Definition 4.2.1) makes it an ideal target for dynamic programming approaches. But it can also be defined recursively (see Definition 4.0.4) or even order-theoretically (see Definition 2.4.9). In Section 4.3 we show that tree-depth is a parameter that is located in-between the (more popular) parameters vertex cover number and tree-width. We then discuss the parameter *generalised tree-depth* in Section 4.4, a parameter introduced by Bouland et al. [23] that is based on a variant of a game that defines tree-depth. The generalised tree-depth of a graph is a lower bound for both its tree-depth and its max leaf number. Next, in Section 4.5 we prove fixed-parameter tractability and hardness results for some problems parameterized by tree-depth:

- Precolouring Extension: We show that Precolouring Extension is $\mathsf{W}[1]$-hard parameterized by tree-depth. We give a reduction from the Multicolour Clique problem to List Colouring, and from List Colouring to Precolouring Extension.

- Boxicity: We state an additive 1-approximation algorithm for Boxicity parameterized by tree-depth. We show that, given a graph G, we can find a $(b+1)$-box representation of G in linear FPT time such that the boxicity of G is either b or $b+1$.

Lastly we discuss some open questions related to tree-depth in Section 4.6.

4.1 Background

Nešetřil and Ossona de Mendez [130] introduced the name tree-depth and established its close connection to nowhere-dense and bounded expansion graph classes: It is possible to characterise nowhere-dense and bounded expansion graph classes in terms of a partition into graphs of bounded tree-depth (cf. Definition 2.3.32 and Theorem 2.3.33). However, even though this revived the interest in the parameter, tree-depth was not a new concept – it was studied before under the names of *vertex ranking number ordered chromatic number*, *minimum elimination tree height* and *rank function* [18, 54, 134, 145].

Tree-depth is also closely related to the parameter *elimination distance to a class* $\mathcal{C}$ that we introduce in Chapter 5 as a new distance parameter. We show in Section 5.4 that tree-depth is in fact the special case of elimination distance to the class of edgeless graphs. The notion of deletion in tree-depth is that of elimination, which allows for parallel deletion in each component at each deletion step. So tree-depth is also clearly a distance parameter.

A useful fact about graphs of bounded tree-depth is that they do not contain long paths. We give a proof of this here.[1]

Lemma 4.1.1. *Let $k \geq 1$ and let G be a graph with tree-depth at most k. Then G does not contain a path of length greater than $2^k - 2$.*

Proof. The proof is by induction on k. We assume without loss of generality that G is connected, because otherwise we can just prove the statement for each component. If $k = 1$, then G only contains isolated vertices and no paths.

Suppose $k > 1$ and no graph with tree-depth less than k contains a path of length greater than $2^{k-1} - 2$. Let v be a vertex so that all components of $G \setminus \{v\}$ have tree-depth less than k. Then none of these components contain a path of length greater than $2^{k-1} - 2$. A path of length greater than $2^{k-1} - 2$ in G must therefore contain v and is of length at most $(2^{k-1} - 2) + (2^{k-1} - 2) + 2 = 2^k - 2$. □

4.2 Alternative characterisations of tree-depth

In the following we give a number of different characterisations of tree-depth. It is a very versatile parameter and the different definitions are useful in different situations.

The first characterisation is in terms of a decomposition of the graph. We now show that the definition given above (Definition 4.0.4) is equivalent to the following definition in terms of a tree-depth decomposition:

Definition 4.2.1. Let G be a graph. A *tree-depth decomposition of* G is a rooted directed forest that contains G in its symmetric transitive closure.

[1] This fact also follows from the discussion in Section 6.2 in [133], but we give a direct proof here.

Proposition 4.2.2. *Let G be a graph. The tree-depth of G is the minimum height of a tree-depth decomposition of G.*

Proof. Assume without loss of generality that G is connected. We prove the statement by induction. Let $k = 0$. Then the graph has tree-depth 0 and an empty, i.e. height 0, tree-depth decomposition.

Let $k > 0$ and suppose the statement holds for smaller values. Suppose G has tree-depth k. Then there is $v \in V(G)$ such that $G \setminus \{v\}$ has tree-depth $k-1$. By the induction assumption there is a tree-depth decomposition of $G \setminus \{v\}$ of height $k-1$. We can construct a tree-depth decomposition of G by adding v as a root to that tree-depth decomposition, which then has height k.

Conversely assume that there is a tree-depth decomposition of height k of G. Since G is connected, it has a single root v. If we remove the root v, we obtain a forest of tree-depth decompositions, each for a component of $G \setminus \{v\}$, and each of height at most $k-1$ (but at least one of height exactly $k-1$). Thus, by the induction assumption, $G \setminus \{v\}$ has tree-depth $k-1$, and G has tree-depth k. □

There is also an order theoretic characterisation of tree-depth, using the notion of an elimination order defined in Definition 2.4.9. Instead of considering the tree in Definition 4.2.1 as a separate graph defined on the same vertices, we can view it as a tree-order defined on the vertices.

Proposition 4.2.3. *A graph G has $td(G) \leq k$ if, and only if, there is an elimination order of height at most k for G.*

Proof. We assume without loss of generality that G is connected and prove this by induction on k.

If $k = 0$, then G must be empty (by both definitions) and the statement holds.

Suppose $k > 0$ and the statement holds for smaller values. First assume that G has tree-depth k. Then there is $v \in V(G)$ such that $G \setminus \{v\}$ has tree-depth $k-1$. By the induction assumption there is an elimination order of $G \setminus \{v\}$ where the longest chain has length $k-1$. We obtain an elimination order for G by adding v as the minimal element to that order. The longest chain has now length k, so it has height k.

Conversely assume that there is an elimination order $\leq$ of height k for G. Note that since G is connected, there is a unique minimal element: Suppose for a contradiction that there are two minimal elements v, v'. Since G is connected, there is a path P connecting v and v'. In the elimination order $\leq$ every pair of adjacent vertices must be comparable, so every neighbouring pair of vertices on P must be comparable. Since v and v' are both minimal, there must be vertices a, b, c on P such that $b < a$, $b < c$ and b, c are incomparable. But this is impossible in a tree order.

Let v be the single minimal element of G. If we remove v, we obtain an elimination order of height $k-1$ for $G \setminus \{v\}$. Thus, by the induction assumption, $G \setminus \{v\}$ has tree-depth $k-1$, and G has tree-depth k. □

Another characterisation of tree-depth comes from games [79, 85]. Consider the following game played on a connected graph G. The game has two players: Cop player and Robber player. The Robber player controls one robber that occupies a vertex of G and that she can move along edges of G in each round. The Cop player tries to catch the robber, choosing a vertex of G in each round where he places a cop. The Cop player cannot move the cops after placing them.

At the beginning of the game the Robber player places the robber on some vertex in $V(G)$. Then the Cop player announces where he will place his first cop, which can be anywhere in the graph including the current position of the robber.

After $i-1$ rounds of the game, there are $i-1$ cops placed on vertices of the graph and the robber is placed on some vertex v. The Robber player can move the robber around along edges in the graph, but the robber cannot be moved through positions occupied by the first $i-1$ cops. If the robber shares a vertex with a cop at the beginning of the round, it can be moved away (but only if the neighbours are not also occupied by cops). Then the Cop player places a new cop on the graph.

The Cop player wins the k-round game if he can catch the robber in at most k rounds, i.e. if the robber shares a vertex with a cop at the end of a move and the Robber player would not be able to move the robber away in the next round. The Robber player wins the k-round game if after k rounds all cops are placed somewhere on the graph and the robber is still not caught.

This game characterises tree-depth in the following way [23]; the tree-depth of k is exactly the least number of rounds needed for the Cop player to have a winning strategy in the game.

Proposition 4.2.4. *Let G be a graph and let $k > 0$. The tree-depth of G is less than k if, and only if, the Robber player has a winning strategy in the k-round game.*

4.3 Relation to vertex cover number and tree-width

In this section we show that tree-depth as a distance parameter is located in-between (cf. Figure 1.1) the (more popular) parameters vertex cover number (cf. Definition 3.2.2) and tree-width (cf. Definition 3.4.2) in the sense that for any graph G we have

$$tw(G) < td(G) \leq vc(G) + 1.$$

We give a proof of both these statements in the following:

Proposition 4.3.1. *Let G be a graph with vertex cover number k. Then G has tree-depth at most $k+1$.*

Proof. Let A be a vertex cover of size k of the graph G. We use the alternative characterisation of tree-depth given in Definition 4.2.1 and observe that the k vertices of the vertex cover A, arranged as a path in any order, with the remaining vertices at the bottom, give a tree-depth decomposition of the graph of height $k+1$. □

Next we show that tree-depth is an upper bound for tree-width.

Proposition 4.3.2. *Let G be a (non-empty) graph which has tree-depth k. Then G has tree-width at most $k-1$.*

Proof. Let G be a graph with tree-depth k. We prove the statement by induction.

Suppose G only has a single vertex v. Then the only bag is $B := \{v\}$ and this gives us a one-node tree decomposition, which has width 0. This can be easily generalized to a graph with tree-depth 1, i.e. a graph that only contains isolated vertices.

Otherwise assume $k > 1$ and that the statement holds for graphs of tree-depth less than k. Since $k > 1$, there is a vertex $v \in G$ such that $G \setminus \{v\}$ has tree-depth $k-1$. By the induction assumption there is a tree decomposition of $G \setminus \{v\}$ of width at most $k-2$. Adding v to every bag gives us a tree decomposition of G of width at most $k-1$. □

Remark. Using the alternative characterisation of tree-depth in Definition 4.2.1 we can explicitly construct the tree decomposition by making each branch of the tree-depth decomposition (that is, the vertices on a path from the root to a leave of the tree-depth decomposition) a bag of the tree decomposition. This actually gives a path decomposition, and shows that tree-depth is in fact in-between vertex cover number and path-width.

This puts tree-depth in an interesting spot: Vertex cover and tree-width are very popular parameters and a large number of problems have been studied parameterized by one or both of them. Many of them have been shown to be fixed-parameter tractable parameterized by vertex cover number, and on the other hand $\mathsf{W}[1]$-hardness parameterized by tree-width (or path-width) is either conjectured or proven. Determining the complexity status of such problems with regard to tree-depth would help to determine where exactly the line between tractability and intractability is for them. This motivates our study of several problems in Section 4.5, and we discuss some further problems in Section 4.6.

4.4 Generalised tree-depth

Bouland et al. [23] introduce a new parameter called *generalised tree-depth* with the aim of generalising both tree-depth and max leaf number, i.e. so that the generalised tree-depth of a graph G gives a lower bound of both the tree-depth of G and the max leaf number of G.

It is defined via a modified version of the Cop and Robber game described above in Section 4.2. Now the Cop player wins if the robber is confined to either a simple path with cops at both endpoints or a simple cycle. The endpoints of this path (which must be occupied by cops) may be connected to other vertices of the graph, but the other vertices in the path may not be adjacent to any vertices in the rest of the graph. For the Cop player to win the game by confining the robber to a cycle, the cycle must be disconnected (by cops) from the rest of the graph.

Definition 4.4.1. Let G be a graph. Then the generalised tree-depth of G is the least k for which the Cop player has a winning strategy in the modified k-round Cops and Robber game described above. We write $gtd(G)$ for the generalised tree-depth of a graph G.

In Chapter 5 we introduce the parameter *elimination distance to a class* $\mathcal{C}$, and we already mentioned that tree-depth is a special case. We see in Section 5.4.1 that generalised tree-depth gives us another special case: elimination distance to the class $\mathcal{C}$, where $\mathcal{C}$ is a subclass of the graphs with degree at most 2.

4.5 Applications

Now we look at problems parameterized by tree-depth. In this section we consider the problems Precolouring Extension and Boxicity parameterized by tree-depth. We show that, with this parameter, Precolouring Extension is $\mathsf{W}[1]$-hard and that there is an additive 1-approximation algorithm for Boxicity that runs in linear FPT time.

Recall from Section 4.3 that tree-depth as a parameter is 'sandwiched' in-between vertex cover number and tree-width (and path-width) in the following sense:

$$tw(G) \leq pw(G) < td(G) \leq vc(G) + 1.$$

As we have mentioned above in Section 4.3, there are several problems that are known to be hard (e.g. W[1]-hard) parameterized by tree-width or path-width, and known to be fixed-parameter tractable parameterized by vertex cover number. Examples of this include Boxicity [4], Cutwidth [45], as well as several colouring problems [69]. So the border between tractability and intractability for these problems must lie somewhere in between vertex cover number and tree-width. Considering a parameterization by tree-depth allows us to zero in on the line between fixed-parameter tractability and hardness. In the following we exhibit results for two such problems: Precolouring Extension and Boxicity.

4.5.1 Hardness: List Colouring and Precolouring Extension

In the following we show that Precolouring Extension is W[1]-hard parameterized by tree-depth.

The problem is known to be fixed-parameter tractable parameterized by vertex cover number (shown by Fiala et al. [69]), and W[1]-hard parameterized by tree-width (shown by Fellows et al. [63]). So a natural question is: is it fixed-parameter tractable or W[1]-hard parameterized by tree-depth? We show that it is in fact W[1]-hard.

The problem is defined as follows:

Pre-Colouring Extension
Input: A graph G, a list of colours L, and a colouring $c_W : W \to L$ for a set of vertices $W \subseteq V(G)$
Parameter: $td(G)$
Problem: Is there a proper colouring $c : V(G) \to L$ for G such that $c(v) = c_W(v)$ for all $v \in W$?

The proof we give in this section uses a reduction from the List Colouring problem, which is defined as follows:

List Colouring
Input: A graph G, a list of colours L, and an assignment $\ell : V(G) \to 2^L$ of allowed colours for each vertex
Parameter: Vertex cover number of G
Problem: Is there a proper colouring $c : V(G) \to L$ such that $c(v) \in \ell(v)$ for all $v \in V(G)$?

The result List Colouring is W[1]-hard parameterized by vertex cover number is known[2], but we are not aware of a proof in the literature, so, for the sake of completeness, we give one here.

The proof we give here is a reduction from Multicolour Clique, which was shown to be W[1]-hard by Fellows et al. [65] through a reduction from Clique.

The Multicolour Clique is defined as follows:

[2]It is mentioned in [67], but no proof or reference is given.

Multicolour Clique **Input:** An integer k, and a properly k-coloured graph G **Parameter:** k **Problem:** Does G contain a k-clique consisting of one vertex of each colour?

Theorem 4.5.1. List Colouring *is* W[1]*-hard parameterized by vertex cover number.*

Proof. We give a reduction from the Multicolour Clique problem. Our argument mimics and extends the argument given by Fellows et al. [63, Theorem 1] to show that List Colouring is W[1]-hard parameterized by tree-width.

Let G be the given graph, and let c be the given proper colouring of G. We construct a graph G' and lists of allowed colours as follows. Let the global list of colours we use be the vertex set of G, i.e. define $L := V(G)$. The vertices $V(G')$ are as follows:

- For each $1 \leq i \leq k$ we have a vertex v_i. We define $\ell(v_i) := c^{-1}(i)$, i.e. the available colours in G' are the vertices of colour i in G;
- For each non-edge $uw \notin E(G)$ with $c(u) \neq c(w)$, we add a vertex v_{uw} adjacent to $v_{c(u)}$ and $v_{c(w)}$. The list of available colours for v_{uw} is $\ell(v_{uw}) := \{u, w\}$.

Note that G' is a bipartite graph, with $\{v_1, \ldots, v_k\}$ being one of the sides. So $\{v_1, \ldots, v_k\}$ is a vertex cover for G', and the vertex cover number of G' is at most k.

Suppose G has a multicolour clique. Then we can colour G' as follows: Assign to each v_i the vertex from the clique as the colour. All the vertices v_{uw} correspond to non-edges, so it cannot be that both u and w are part of the clique and one of them must be free and available as a colour for v_{uw}.

Conversely assume there is a valid list colouring c for G'. We argue that the vertices $c(v_1), \ldots, c(v_k)$ assigned as colours to $v_1, \ldots, v_k$ form a clique in G.

We have to show that between any pair of these vertices there is an edge. Consider two such vertices v_i and v_j with $1 \leq i < j \leq k$. First note that $c(v_i)$ (with colour i in G) has a different colour from $c(v_j)$ (with colour j in G) because $i \neq j$. Now observe that because they are in different colour classes, by our construction there is no edge between $c(v_i)$ and $c(v_j)$ if, and only if, they appear on the list of a vertex $v_{c(v_i)c(v_j)} \in V(G')$. But there can be no such vertex $v_{c(v_i)c(v_j)}$ because it would be adjacent to both v_i and v_j with colours $c(v_i)$ and $c(v_j)$ respectively, so there would be no free colour available for $v_{c(v_i)c(v_j)}$. Thus $c(v_i)c(v_j) \in E(G)$, and the $c(v_1), \ldots, c(v_k)$ form a clique. This completes the reduction. □

Remark. Note that the vertex cover $\{v_1, \ldots, v_k\}$ is also an independent set, so we are proving a more general statement here: the graph we are reducing to is in fact a bipartite graph with one side bounded by the parameter.

Next we give a reduction from the List Colouring problem to Precolouring Extension.

Theorem 4.5.2. Precolouring Extension *is* W[1]*-hard parameterized by tree-depth.*

Proof. We give a reduction from List Colouring parameterized by vertex cover number. Let G be an instance of List Colouring with allowed colours specified by the function $\ell : V(G) \to 2^L$, where $L = \{a_1, \ldots, a_{|L|}\}$ is the list of all available colours.

We obtain G' from G as follows: For each vertex $v \in V(G)$, and each colour $a \in L$, we add a vertex a_v, i.e. we define

$$W := \{a_v \; midv \in V(G), a \in L\}$$

and $V(G') := V(G) \cup W$.

We then make sure every vertex v from $V(G)$ is adjacent to the vertices representing the colours in L that are not allowed on its list, i.e. we define

$$E(G') := E(G) \cup \{va_v \mid v \in V(G), a \in L, a \notin \ell(v)\}.$$

To create a complete Precolouring Extension instance, we also define a precolouring c_W of W as follows. We define c_W to be the function that maps a_v to a for each $a_v \in W$.

It is easy to see that our construction ensures that G has a valid list colouring if, and only if, G' has a proper precolouring extension.

So it is only left to show that the tree-depth of G' is bounded by a function of k. It is easy to see (and shown in Proposition 4.3.1) that if G has vertex cover number k, then G has tree-depth at most k. Note that W is an independent set because we have defined no edges between vertices from W. Moreover, every vertex $a_v \in W$ is adjacent to at most one vertex from G, namely v. Thus a tree-depth decomposition of G' of height $k+1$ can be obtained from a tree-depth decomposition of G by adding all the vertices from W as leaves. More precisely, we can add the vertex $a_v \in W$ as a leaf to the branch containing v. Thus $td(G') \leq k+1$. This completes the reduction. □

4.5.2 Approximation and Parameterized complexity: Boxicity

In the following we consider the Boxicity problem. Just as Precolouring Extension, it is known to be fixed-parameter tractable parameterized by the vertex cover number of the input graph (proved by Adiga et al. [4]). However, its complexity status with regard to tree-width is unknown, and it is only conjectured to be hard parameterized by tree-width [4].

For this reason, Boxicity is another natural candidate to consider for a parameterization by tree-depth in an attempt to get closer to an accurate understanding of where the problem becomes fixed-parameter tractable.

In this section we consider approximation algorithms for Boxicity. For a hard optimization problem, approximation algorithms and parameterized complexity are normally considered to be two separate approaches to work around the hardness. However, there has been much interest recently in combining approximation algorithms and parameterized complexity.

Approximation algorithms trade in some accuracy for an improved runtime. However, unlike heuristics, they provide a guaranteed runtime and a guaranteed quality of the solution: usually the approximate solution is optimal up to a constant factor (e.g. at most twice as large for a minimization problem). They are an increasingly popular way to deal with NP-hard problems, see e.g. [38, 154].

Recently there has been increasing interest in combining approximation algorithms with parameterized complexity and to study parameterized approximation algorithms, i.e. approximation algorithms that run in time $O(f(k) \cdot n^c)$ for some constant c and computable function f. (See Marx [122] for an overview of some of the different approaches

and parameterizations.) This section gives an example where combining parameterized complexity with approximation algorithms has been fruitful and lead to a new result.

The specific notion we need is that of an *additive p-approximation algorithm*, defined as follows:

Definition 4.5.3. Let P be a graph minimisation problem that, given a graph G, associates with it an integer $\xi(G)$. Let $p > 0$. An algorithm that computes a solution $\xi'(G)$ such that $\xi'(G) \leq \xi(G) + p$ is called an *additive p-approximation algorithm for P*.

Remark. In this section we measure the runtime of the approximation algorithm solving a parameterized problem as a function of both the parameter and the input size. We refer to such algorithms, designed to optimize the runtime in terms of both the size of the input and the parameter, as *parameterized approximation algorithms*.

In the following we exhibit a parameterized additive 1-approximation algorithm for Boxicity. The parameter we are considering is tree-depth and the algorithm runs in linear FPT time, i.e. in time $O(f(td(G) \cdot |G|))$, where G is the input graph and f is some computable function.

The boxicity of a graph G is the smallest dimension b that allows a b-dimensional representation of G as a box graph, i.e. it is the minimum dimension in which G can be represented as an intersection graph of axis-parallel boxes. It was introduced by Roberts [140]. The Boxicity problem is an optimisation problem where, given a graph G, we are trying to find the boxicity of G. We do not work with box graphs here, and instead use an alternative characterisation of boxicity. In the following we define a characterisation via intersection graphs that is more useful for our purposes here. For a reader familiar with box graphs it should be intuitively clear that both definitions are equivalent, and we do not give a proof here.

A graph is an interval graph if it can be expressed via intervals on the real line:

Definition 4.5.4. An *interval graph* is a graph where each vertex can be associated with an interval on the real line so that two vertices are adjacent if, and only if, their intervals overlap.

We can combine multiple such graphs by considering their intersection:

Definition 4.5.5. Let $G_1, \ldots, G_k$ be graphs on the same vertex set. Then a graph G on the same vertex set is called the *intersection* of $G_1, \ldots, G_k$ iff G contains an edge iff all the G_i contain it.

We give a definition of *boxicity* using these notions:

Definition 4.5.6. The *boxicity* of a graph G is the smallest positive integer b such that G is the intersection of b interval graphs. We call the b interval graphs a *b-box representation of G*.

In terms of this definition, the Boxicity problem is an optimisation problem where we are given a graph G and are trying the find the smallest integer b such that G admits a b-box interpretation. The decision version of the problem is formally defined as follows:

BOXICITY **Input:** A graph G and a positive integer b **Parameter:** $td(G)$ **Problem:** Does G have boxicity at most b?

The (classical, un-parameterized, version of the) problem is NP-complete [44], and the problem is known to be 'resistant' to the natural parameterization by the output value: just determining whether the boxicity of a graph is at most two is already NP-complete [107, 160]. Thus one has to look at structural parameters.

Adiga et al. [4] show that BOXICITY is FPT parameterized by the vertex cover number and also give an additive 2-approximation algorithm for BOXICITY parameterized by max leaf number (see Definition 3.6.2).

The complexity of BOXICITY parameterized by tree-width is unknown, in fact Adiga et al. [4] conjecture that it is NP-hard for some graphs of constant tree-width. This makes it an interesting candidate to consider for a parameterization with tree-depth.

4.5.2.1 A parameterized approximation algorithm for Boxicity

In the following we give an parameterized additive 1-approximation algorithm for BOXICITY. The parameter we are considering is tree-depth. We show that given a graph G, we can find a b-box or $(b+1)$-box representation of G in linear FPT time, where b is the boxicity of G.

The core idea of the algorithm is to compute a tree-depth decomposition of the input graph and reduce it to a 'pruned tree-depth decomposition' that only keeps a bounded number of vertices at each level of the tree, where the *ith level* of a tree-depth decomposition is the set of vertices with distance i to the root. We show that solving the BOXICITY problem on that smaller graph (almost) gives a solution for the whole graph. So this is essentially a kernelization argument – except that the boxicity of the kernel might be one less in some cases.

Definition 4.5.7. Let G be a graph and let T be a tree-depth decomposition of G. The *type* with respect to T of a vertex v in G is defined as a pair (A, B) of a set of integers A and a set of types B where

$$A := \{i \mid v \text{ is adjacent to a vertex at level } i \text{ in } T\};$$
$$B := \{type(w) \mid w \text{ is a child of } v \text{ in } T\}.$$

A *pruned tree-depth decomposition* of G with respect to T is a subgraph of T where in each set of siblings we have deleted all but one copy of each type present, including the subtrees the deleted siblings are rooted at. Formally we define it recursively as follows:

Definition 4.5.8. Let G be a graph and let T be a tree-depth decomposition of G of height k. If $k \leq 1$, then $T' := T$ is a *pruned tree-depth decomposition* of G with respect to T.

Otherwise, if $k > 1$, and R is the set of roots of T, let R' be a minimal subset of R with the property that

$$\{type(r) \mid r \in R\} = \{type(r') \mid r' \in R'\}.$$

Let $T_{r'}$ be a pruned tree-depth decomposition of $G[V(R_{r'})]$ with respect to the height $k-1$ tree-decomposition $T[V(R_{r'})]$, where $R_{r'}$ is the set of vertices reachable from r' in T. Then the subgraph T' of T induced by the vertices

$$R' \cup \bigcup_{r' \in R'} V(T_{r'}),$$

is a *pruned tree-depth decomposition* of G with respect to T.

Remark. Note that even though there is not a unique pruned tree-depth decomposition of a graph with respect to some fixed tree-depth decomposition, all the pruned tree-depth decomposition must be isomorphic.

Lemma 4.5.9. *Given a graph G and a tree-depth decomposition T of G we can compute a pruned tree-depth decomposition T' in linear time. Moreover, the size of T' is bounded by a function of $td(G)$.*

Proof. Let k be the tree-depth of G. We begin by computing the type of all leaves. For each leaf z in T, we can compute the type by looking at the at most $k-1$ ancestors of z. In each set of siblings we delete all vertices of duplicate types, i.e. if there is more than one vertex of some type, we delete all but one. The number of siblings we keep is bounded by 2^k – a function of the tree-depth.

We then proceed from the bottom up. At level i, we can again compute the type of a vertex by looking at the children (whose number is bounded by a function of k) and the ancestors in T, also bounded by k. We then again delete all duplicate siblings (and their respective subtrees). Again, the number of kept siblings is bounded by a function of k.

So overall the pruned tree-depth decomposition T' is a tree of height k with branching bounded by a function of k. Thus its size is bounded by a function of the tree-depth of G.

Note that we have considered every vertex of G once, with the number of steps at each vertex bounded by a function of the parameter k. Thus the algorithm runs in linear FPT time. □

Remark. We say that the vertex that is kept when deleting vertices of the same type *represents* the deleted vertices. Note that the pruning defines a graph homomorphism $\varphi : V(T) \to V(T')$ that maps each vertex in T to the vertex of the same type in T' that was kept to represent it. It should be clear that the mapping can be computed while pruning the tree.

Now we have all the tools to prove the following:

Theorem 4.5.10. *Given a graph G, we can, in linear FPT time, find a b-box representation of G, such that the boxicity of G is either b or $b-1$.*

Proof. Let k be the tree-depth of the input graph G. First we compute a tree-depth decomposition T of G, and then use Lemma 4.5.9 to compute a pruned tree-depth decomposition T' of G with respect to T. Let G' be the subgraph of G induced by the vertices of T'. The size of G' is bounded by a function of the tree-depth k of G, so we can find a minimal b-box representation of G' in time bounded by a function of k. Note that b is at most the boxicity of G, since G' is a subgraph of G.

Let $(I'_1, \ldots, I'_b)$ be a minimal b-box representation of G'. We extend it to a $(b+1)$-box representation of G. Recall that there is a homomorphism φ, computed while constructing

T', that maps each vertex $v \in G$ to the vertex $\varphi(v) \in G'$ that represents it in the pruned tree-depth decomposition. For each $v \in G \setminus V(G')$ we add an interval for v that is identical to the one for $\varphi(v)$ to each I'_j. Let $(I_1, \ldots, I_b)$ be the b-box representation obtained after adding all the missing vertices. This can be computed in linear time.

This gives us all the edges from G, but possibly more than just those. We add an additional interval graph J that removes unwanted edges. We define J recursively as follows. Starting at the root of T, we define the interval of a subtree as an interval for the root that contains the disjoint intervals of all the subtrees rooted at its children. This interval graph is just the symmetric transitive closure of T. This can also be computed in linear time.

We now prove that $(I_1, \ldots, I_b, J)$ is a $(b+1)$-box representation of G. Suppose $uv \in E(G)$. Since T is a tree-depth decomposition of G, we have that $uv \in E(J)$. Since φ is a homomorphism, we have mapped u and v such that $uv \in E(I_j)$ for all j. Thus the edge is present in the box-representation.

Conversely, assume that uv is an edge in the box-representation. Since $uv \in E(J)$, we must have that u and v occur in the same branch, say without loss of generality that u is an ancestor of v in T. Moreover, we know that $\varphi(u)\varphi(v) \in E(G')$ and thus $type(\varphi(v))$ records that it has an edge to a vertex in the same branch at the hight of $\varphi(u)$. Since v and $\varphi(v)$ agree on types, $type(v)$ tells us that v is adjacent to u. Thus $uv \in E(G)$. □

This gives us an additive 1-approximation algorithm for Boxicity parameterized by tree-depth. So it is theoretically possible that the boxicity computed by the algorithm is off by one – but how likely is it to happen in practice? Unfortunately it is easy to construct simple examples where it does, see for example:

Example 4.5.11. Consider the two graphs in Figure 4.1: The right graph G' is induced by the vertices of a pruned tree-depth decomposition of the left graph G. The right graph G' clearly is an interval graph, while the left G has no interval representation, i.e. boxicity of at least 2. It is easy to verify that the boxicity of G is indeed 2.

4.6 Open questions

What motivated us to study the problems in Section 4.5 is that tree-depth is a graph parameter that lies between vertex cover number and tree-width in the following sense: for any graph G we have $tw(G) \leq td(G) \leq vc(G)$. The problems we studied in Section 4.5 are known to be fixed-parameter tractable parameterized by vertex cover number, but either known to be hard parameterized by tree-width or with an unknown status.

There are several problems with these characteristics that have an open status with regard to parameterizations with tree-depth. A group of problems with this property are the following: Graph layout problems are optimisation problems where the objective is to linearly order the vertices to minimise some property. For example, the imbalance of a vertex is the absolute value of the difference of the number of its neighbours to the left and to the right. The Imbalance problem tries to find an order that minimises the sum of all the imbalances of vertices in a graph. Other graph layout problems are Cutwidth, Bandwidth and Distortion. Many graph layout problems have withstood attempts to prove that they are fixed-parameter tractable parameterized by the tree-width of the graph. However, it was shown by Fellows et al. [67] that a number of graph layout problems are fixed-parameter tractable parameterized by the vertex cover number of the

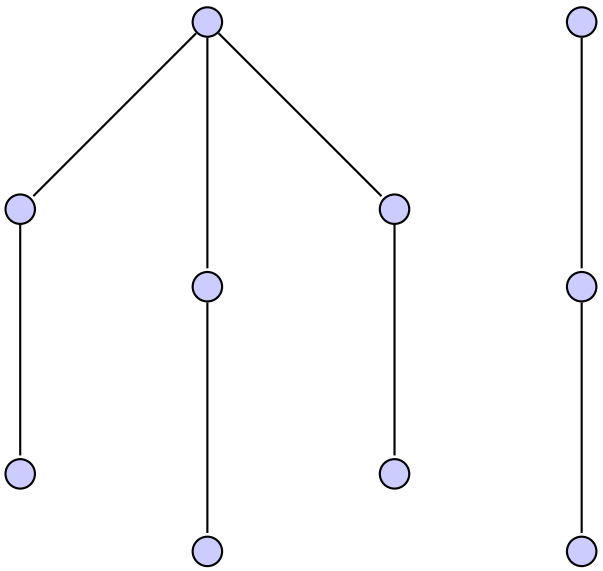

Figure 4.1: An example where the boxicity of the pruned graph G' is one less than the original graph G.

graph. There are also several graph colouring problems where parameterizations with tree-depth should be studied for the same reasons. (cf. [33, 69]).

CHAPTER 5

Elimination distance

The study of parameterized algorithmics for graph problems has uncovered a large variety of structural parameters of graphs. In Chapter 3 we have seen several examples of *distance parameters*, specific kinds of structural graph parameters that measure the distance of a graph G to a graph class $\mathcal{C}$ in some way. Among these, the simplest distance parameter is 'deletion distance to $\mathcal{C}$': it counts the number of vertices that one must delete from a graph G to obtain a graph in $\mathcal{C}$ (cf. Section 3.1).

In this chapter we introduce for any graph class $\mathcal{C}$ the graph parameter 'elimination distance to $\mathcal{C}$'. Elimination distance generalises the notion of deletion in the parameter 'deletion distance to $\mathcal{C}$'. This is made precise below, but intuitively elimination distance to a class $\mathcal{C}$ can be understood as a 'parallelized deletion distance' that allows deleting a vertex from every connected component simultaneously at each deletion step until every component is contained in a given class $\mathcal{C}$. It is based on elimination trees or tree-depth decompositions and can also be thought of as a natural generalisation of the parameter tree-depth, which was introduced in Chapter 4. In fact, tree-depth can be characterised as the elimination distance to the class of edgeless graphs, so it is a specific case of elimination distance. (See Section 5.4 below for a discussion of the relationship).

In Section 5.1 we formally introduce the notion of elimination distance to a class $\mathcal{C}$ in all generality and study the relationship to other parameters.

Recall from Section 3.2 that the vertex cover number of a graph G is the size of a minimal vertex cover of G. Alternatively, it is easy to see that it can be characterised as the deletion distance to the class of edgeless graphs: if we delete every vertex from the graph that is in the vertex cover, there are no edges in the graph left. So vertex cover number can be seen as a special case of deletion distance to a class $\mathcal{C}$.

A vertex cover can be also interpreted as a stricter version of a tree-depth decomposition, where the tree-depth decomposition has to be a path. Or, equivalently, just as in our description of elimination distance above, the notion of deletion in tree-depth can be understood as a 'parallelized deletion distance' that generalises the notion of deletion in deletion distance: instead of deleting a fixed number of vertices, in a fixed number of steps we recursively delete vertices from each component of the graph until the graph has no edges left.

In a sense, the graph parameter introduced in this chapter, elimination distance to a class $\mathcal{C}$, can be thought of as a natural generalisation of both deletion distance (generalising the notion of deletion) and tree-depth (allowing for more general classes than the class of edgeless graphs). The elimination distance to any class $\mathcal{C}$ of a graph G gives a lower

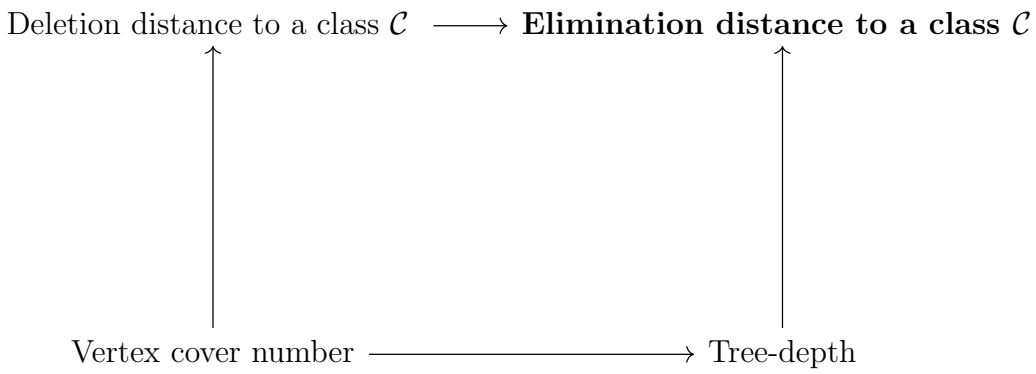

Figure 5.1: Illustration of the relationship between the parameters. An arrow means 'is an upper bound for'.

bound for both the tree-depth of G and the deletion distance of G to $\mathcal{C}$. It thus occupies an interesting new space in the hierarchy of parameters demonstrated in Figure 5.1.

Tree-width is another parameter that gives a lower bound for both tree-depth and vertex cover number. In that sense, elimination distance to a class $\mathcal{C}$ is an alternative generalisation of vertex cover number and tree-depth. As a notion of distance it is more natural than tree-width, because it is defined in terms of vertex deletions instead of the 'width' of a tree decomposition. The relationship of elimination distance to $\mathcal{C}$ to tree-width depends on the choice of $\mathcal{C}$: for example, if $\mathcal{C}$ is the class of graphs with degree bounded by d (for $d \geq 3$), then $\mathcal{C}$ has unbounded tree-width and the same is a fortiori true of graphs with elimination distance k to $\mathcal{C}$. On the other hand, if $\mathcal{C}$ is just the class of edgeless graphs, then elimination distance to $\mathcal{C}$ is tree-depth. The class of all paths has unbounded tree-depth, but constant tree-width.

Elimination distance offers a number of algorithmically interesting characterisations. It can be defined through recursive deletions, as described, but also order theoretically by defining an order on the vertices of the graph that tells us which vertices should be 'eliminated' first. There is also a more combinatorial way: If a graph has elimination distance k to a class $\mathcal{C}$, then it can be split in a graph with a tree-depth decomposition of height k, and a graph where every component is contained in $\mathcal{C}$ and connected to at most one branch of the tree-depth decomposition. This allows for interesting algorithmic applications exploiting the tree structure of the tree-depth decomposition. Just as for tree-depth, this can also be translated into an order theoretic definition. Lastly the class of graphs with elimination distance k to a class $\mathcal{C}$ can also be obtained by repeatedly closing $\mathcal{C}$ under the addition of apex vertices and taking the closure under disjoint unions. We discuss these alternative characterisation in Section 5.2.

In Section 5.3 we establish that if a class $\mathcal{C}$ has bounded expansion, then the class of graphs $\mathcal{C}_k$ with elimination distance k to $\mathcal{C}$ also has bounded expansion. This is done using one of the alternative characterisations established before.

Every choice of the graph class $\mathcal{C}$ gives us a different parameter. This makes the parameter very versatile, because the elimination distance to $\mathcal{C}$ has different properties for different classes $\mathcal{C}$. In Section 5.4 we consider the case where $\mathcal{C}$ is the class of edgeless graphs and establish that this gives us an alternative characterisation of tree-depth.

In Section 5.5 we explore the case where $\mathcal{C}$ is a class of graphs of bounded degree. Along the way we establish a different characterisation of this particular parameter that

is interesting in itself: we define the notion of an *elimination order to degree d*, which is similar to the elimination order to $\mathcal{C}$ introduced in Section 5.2, but makes use of the locality afforded by bounded degree classes. It also defines a partial order on the vertices, now indicating the order in which they need to be eliminated to reduce the degree to d, but is defined in a different way.

As discussed in the introduction, there are generally two reasons why the study of a parameter can be interesting in parameterized complexity. Either the parameter itself is of interest, or we want to understand the complexity of other problems with regard to the parameter. Elimination distance to a class $\mathcal{C}$ is an interesting parameter in both these ways and Chapter 6 and Chapter 7 give examples for each.

In Chapter 6 we show that Graph Isomorphism is FPT parameterized by elimination distance to bounded degree. This generalises known results, in particular the result by Bouland et al [23] that shows it is FPT parameterized by tree-depth and generalised tree-depth. Here the parameter gives us a wider class of graphs where the Graph Isomorphism problem is known to be tractable, and improves our understanding of the problem.

In Chapter 7, we show that given a finite set of minors that characterise a minor-closed class $\mathcal{C}$, we can find a finite set of minors that characterizes the class of graphs with elimination distance to $\mathcal{C}$. This gives an explicit FPT-algorithm to check whether a given graph has a certain elimination distance to $\mathcal{C}$.

5.1 Elimination distance to a class $\mathcal{C}$

In this section we formally introduce the parameter in its full generality. We relax the notion of 'deletion distance' so that rather than considering the sequential deletion of k vertices, we consider the recursive deletion of vertices in a tree-like fashion. To be precise, we say that a graph G has *elimination distance* $k+1$ from a class $\mathcal{C}$ if, in each connected component of G we can delete a vertex so that the resulting graph has elimination distance k to $\mathcal{C}$.

We want to make sure that deleting vertices gets us closer to the target class $\mathcal{C}$, and that is why we restrict the classes $\mathcal{C}$ to ones that are closed under taking subgraphs. Formally, elimination distance is defined as follows:

Definition 5.1.1. Let $\mathcal{C}$ be a class of graphs closed under taking subgraphs. The *elimination distance to* $\mathcal{C}$ of a graph G is defined as follows:

$$ed_{\mathcal{C}}(G) := \begin{cases} 0, & \text{if } G \in \mathcal{C}; \\ 1 + \min\{ed_{\mathcal{C}}(G \setminus v) \mid v \in V(G)\}, & \text{if } G \notin \mathcal{C} \text{ and } G \text{ is connected;} \\ \max\{ed_{\mathcal{C}}(H) \mid H \text{ a connected component of } G\}, & \text{otherwise.} \end{cases}$$

Remark. Observe that if a graph G has elimination distance 1 to a class $\mathcal{D}$, and all graphs in $\mathcal{D}$ have elimination distance k to a class $\mathcal{C}$, then G has elimination distance $k+1$ to $\mathcal{C}$. This fact is going to be used in some induction proofs.

In the following we see the relationship of the parameter to the other distance parameters deletion distance to a class $\mathcal{C}$ (Proposition 5.1.2) and tree-depth (Proposition 5.4.1).

For deletion distance, the following is a straightforward consequence that can be easily seen from the definition. In the worst case we have to delete all vertices from a single

connected component without being able to split the graph into several components. In that case deletion distance and elimination distance are the same.

Proposition 5.1.2. *Let $\mathcal{C}$ be a class of graphs. If a graph G has deletion distance k to $\mathcal{C}$, then G has elimination distance at most k to $\mathcal{C}$.*

Recall the notion of sparsity called 'bounded expansion' from Definition 2.3.31. In Section 5.3 we establish that if a class $\mathcal{C}$ has bounded expansion, then the class of graphs $\mathcal{C}_k$ with elimination distance k to $\mathcal{C}$ also has bounded expansion. In order to prove this, we first we need to make use of a different characterisation of elimination distance introduced in in Section 5.2.

5.2 Alternative characterisations of elimination distance

In this section we introduce a number of algorithmically interesting characterisations of elimination distance. First we give a characterisation given by an order on the vertices that tells us which vertices should be removed first. Next we establish a more combinatorial way that splits the graph in a part with a tree-depth decomposition of height bounded by the parameter, and one where every component is contained in $\mathcal{C}$ and connected to at most one branch of the tree-depth decomposition. This allows for interesting algorithmic applications exploiting the tree structure of the tree-depth decomposition. The actual definition is also given in order-theoretic terms. Lastly we establish a characterisation using apex graphs and closures under disjoint unions.

5.2.1 Elimination order to $\mathcal{C}$

We now introduce an equivalent characterisation of this parameter by defining an order on the vertices. If G is a graph that has elimination distance k to a class $\mathcal{C}$, then we can associate a certain tree order $\leq$ with it as defined below. The idea is that from every connected component S of G we can delete a vertex v_S so that the elimination distance to $\mathcal{C}$ is reduced. We define $v_S \leq u$ for all vertices $u \in S$ and vertices in distinct components of $S \setminus v_S$ are made incomparable with respect to $\leq$. We proceed in this fashion until all components are contained in $\mathcal{C}$, and make all vertices in these incomparable (and indeed $\leq$-maximal). Note that all vertices that are not $\leq$-maximal must be comparable to their neighbours. In the following we give the definition inspired by this and prove it gives us an equivalent notion to the one given in Definition 5.1.1:

Definition 5.2.1. Let G be a graph and let $\mathcal{C}$ be a class of graphs closed under taking subgraphs. A tree order $\leq$ on $V(G)$ is an *elimination order to* $\mathcal{C}$ for G if for each $v \in V(G)$ we have that:

- if there is a vertex $u > v$, then there is no vertex w that is a neighbour of v and incomparable to v, i.e. there is no vertex w such that $uw \in E(G)$ and $v \not\leq w$ and $w \not\leq v$.
- if v is $\leq$-maximal, then the maximal subgraph S of G that only contains vertices that are $\leq$-maximal and contains v is contained in the class $\mathcal{C}$. Moreover any two

vertices that are less than a vertex in S must be comparable, i.e. for any two vertices $w, w' \in S$, we have that if $u < w$, $u' < w'$ then $u \leq u'$ or $u' \leq u$.

Remark. Note that this definition naturally extends the notion of an elimination order defined in Definition 2.4.9.

The following proposition implies Proposition 4.2.3, which states that the height of an elimination order (see Definition 2.4.9) gives us an alternative characterisation of tree-depth.

Proposition 5.2.2. *A graph G has $ed_{\mathcal{C}}(G) \leq k$ if, and only if, there is an elimination order to $\mathcal{C}$ of height at most k for G.*

Proof. Suppose first that $ed_{\mathcal{C}}(G) \leq k$. We proceed by induction on k. If $k = 0$, then all components of G are contained in $\mathcal{C}$ and we define the elimination order $\leq$ to be the identity relation on $V(G)$. Then every $v \in V(G)$ is maximal, and the connected component S of the subgraph of G induced by the $\leq$-maximal vertices of G containing v is thus contained in the class $\mathcal{C}$.

Suppose $k > 0$ and the statement is true for smaller values. If G is not connected, we apply the following argument to each component. So in the following we assume that G is connected. Thus, there is a vertex $a \in V(G)$ such that the components $C_1, \ldots, C_r$ of $G \setminus a$ all have $ed_{\mathcal{C}}(C_i) \leq k-1$. So by the induction hypothesis each C_i has an elimination order $\leq_i$ to $\mathcal{C}$ of height at most $k-1$ with the properties in Definition 5.2.1.

Let

$$\leq := \{(a, w) \mid w \in V(G)\} \cup \bigcup_i \leq_i .$$

Then $\leq$ is clearly a tree order for G. Note that there is no vertex w that is a neighbour of a and incomparable to a.

Conversely assume there is an elimination order $\leq$ to $\mathcal{C}$ of height d for G. We again proceed by induction on k. If $k = 0$, then $\leq$ is empty and all vertices v in G are $\leq$-maximal. Thus all components of G are contained in $\mathcal{C}$ as required.

Now, suppose $k > 0$. If v is an $\leq$-minimal element then for any u for which there is a path from v to u, we must have $v \leq u$. Thus, if G is connected, there is a unique $\leq$-minimal element v. Note that $\leq$ restricted to a component C of $G \setminus v$ has height $k-1$ and thus by the induction hypothesis we have that $ed_{\mathcal{C}}(C) \leq k-1$. If G is not connected, this argument can be applied to each component of G. □

5.2.2 Separation into two parts

In the following we demonstrate that the previous characterisation can also be interpreted in a different way, giving us a second alternative characterisation of elimination order to $\mathcal{C}$. We derive that characterisation by separating the maximal elements from the non-maximal elements: that way, we can split a graph with an elimination order to some class $\mathcal{C}$ in two parts; one with an elimination order defined on it, and one where every connected component is contained in $\mathcal{C}$. Furthermore, any two vertices from the former part that are neighbours of vertices from the same component of the latter part must be comparable in the elimination order. This means that if G is a graph that has elimination distance k to a class $\mathcal{C}$, we can associate an elimination order $\leq$ for a subgraph H of G of height k with G, so that each component of $G \setminus V(H)$ is contained in $\mathcal{C}$, and is connected to H along just one branch (this is defined more formally below).

Proposition 5.2.3. *Let $\mathcal{C}$ be a class of graphs closed under taking subgraphs. Let G be a graph and let $\leq$ be an elimination order to $\mathcal{C}$ for G of height k. If A is the set of vertices in $V(G)$ that are not $\leq$-maximal, then:*

1. *$\leq$ restricted to A is an elimination order of height $k-1$ for $G[A]$;*
2. *every component of $G \setminus A$ is contained in $\mathcal{C}$;*
3. *and if C is the vertex set of a component of $G \setminus A$, and $u, v \in A$ are $\leq$-incomparable, then either $E(u, C) = \emptyset$ or $E(v, C) = \emptyset$.*

Proof. By Definition 5.2.1 $\leq$ is a tree-order. Moreover, since any $v \in A$ is non-maximal, also by Definition 5.2.1, there is no vertex that is a neighbour of v and incomparable to v. Hence if there is an edge between any $u, v \in A$, either $u < v$ or $v < u$, and (1) follows.

Since $G \setminus A$ contains the $\leq$-maximal elements, it follows from the definition of an elimination order to $\mathcal{C}$ that every component of $G \setminus A$ is contained in $\mathcal{C}$, establishing (2).

To show (3), let C be the vertex set of a component of $G \setminus A$ and let $u, v \in A$ be such that $E(u, C) \neq \emptyset$ and $E(v, C) \neq \emptyset$. Then there are $a, b \in C$ such that $au, bv \in E(G)$. By Definition 5.2.1, since u is non-maximal and a is a neighbour of v, they must be comparable. Since a is maximal, we must have $u < a$. Similarly, $v < b$. So again by Definition 5.2.1 we must have either $u \leq v$ or $v \leq u$ and we conclude that u and v are comparable, proving (3). □

We also have a converse to the above in the following sense.

Proposition 5.2.4. *Let $\mathcal{C}$ be a class of graphs closed under taking subgraphs. Suppose G is a graph with $A \subseteq V(G)$ a set of vertices and $\leq_A$ an elimination order of $G[A]$ of height k, such that:*

1. *every component of $G \setminus A$ is contained in $\mathcal{C}$;*
2. *if C is the vertex set of a component of $G \setminus A$, and $u, v \in A$ are incomparable, then either $E(u, C) = \emptyset$ or $E(v, C) = \emptyset$.*

Then, $\leq_A$ can be extended to an elimination order to $\mathcal{C}$ for G of height $k+1$.

Proof. Let

$$\begin{aligned} \leq := &\leq_A \cup \{(v, v) \mid v \in (V(G) \setminus A)\} \\ &\cup \{(u, v) \mid u \in A, v \in C, C \text{ a component of } G \setminus A, \\ &\qquad E(w, C) \neq \emptyset \text{ for some } u \leq w\}. \end{aligned}$$

Then it is easily seen that $\leq$ is a tree order on G. Indeed, $\leq_A$ is, by assumption, a tree order on A and for any $v \in V(G) \setminus A$, assumption (2) guarantees that $\{w \mid w \leq v\}$ is linearly ordered.

Let $v \in V(G)$. If there is a vertex $u > v$, then by our construction $v \in A$. Since $\leq_A$ is an elimination order, v is comparable to all its neighbours in A. Let $w \in V(G) \setminus A$ be a neighbour of v not contained in A, and let C be the component of $G \setminus A$ containing w. Then by the construction we have $u < w$ because $w \in E(v, C)$ and thus $E(v, C) \neq \emptyset$.

Now suppose v is $\leq$-maximal, and let C be the connected component of $G \setminus A$ containing v. By assumption (1), we have that C is contained in $\mathcal{C}$. Let $w, w' \in C$ and let

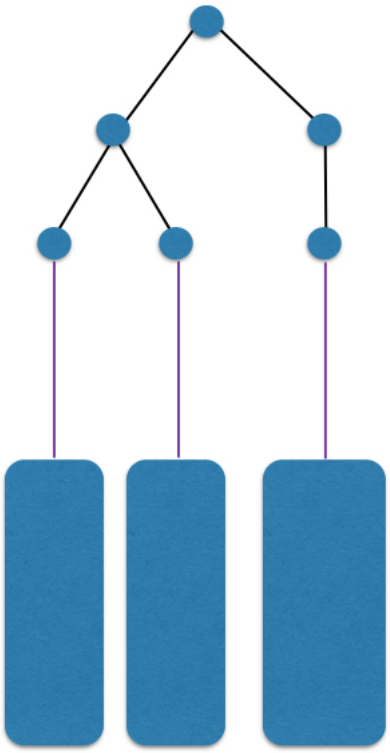

Figure 5.2: Illustration of an elimination order to $\mathcal{C}$ of a graph G. The tree on top is a tree-depth decomposition of a subgraph of G. The remaining components (at the bottom) are all contained in $\mathcal{C}$ and connected to at most one branch of the tree-depth decomposition.

u, u' be vertices of G such that $u < w$ and $u' < w'$. By the construction of $\leq$, there must be vertices $a \geq u$ and $a' \geq u'$ such that $E(a, C) \neq \emptyset$ and $E(a', C) \neq \emptyset$. So by the contrapositive of assumption (2) we have that a and a' are comparable. But $\leq$ is a tree order, so we must have that u and u' are comparable, completing the proof. □

Remark (1). In the remainder of the dissertation, given a graph G and an elimination order $\leq$ to $\mathcal{C}$ for some class $\mathcal{C}$ we call the subgraph of $V(G)$ induced by the non-maximal elements of the order $\leq$ the *non-maximal subgraph of* G *under* $\leq$.

Remark (2). In the proof of Proposition 5.2.4 above, a suitable tree order on a subset A of $V(G)$ is extended to an elimination order to $\mathcal{C}$ of G by making all vertices not in A maximal in the order. This is a form of construction we use repeatedly when we are working with these definitions in Chapter 6.

Proposition 5.2.3 and Proposition 5.2.4 establish a third characterisation of elimination distance to a class $\mathcal{C}$ via a separation of the graph. Note that, although the definition here is given in terms of an elimination order, we could also state it in terms of a tree-depth decomposition: then the graph is split into a graph of tree-depth k, and one where every component is contained in $\mathcal{C}$ and connected to at most one branch of the tree-depth decomposition. This is illustrated in Figure 5.2

This separation can be exploited in algorithms, and we see an example of how it can be used in Section 5.3, and later in Chapter 6.

5.2.2.1 Characterisation via games

Another immediate consequence of the characterisation via the separation in two parts, is that elimination distance to a class $\mathcal{C}$ can also be characterised via games. Recall from Section 4.2 that tree-depth can be characterised via a Cop and Robber game, where the Robber player loses when she cannot move anymore. Similarly, the generalised tree-depth of a graph (Section 4.4.1) is defined via a Cop and Robber game, where the Robber player loses when she is confined to a path (with cops at the endpoints) or a cycle that is disconnected (by cops) from the rest of the graph.

From the above it follows that the same Cop and Robber game can be adapted to characterise elimination distance to a class $\mathcal{C}$. This is done by changing the winning condition of the Cop player to confining the Robber player to a connected component that is part of $\mathcal{C}$. The Cop player can then play the same strategy as for tree-depth on the non-maximal subgraph of G under $\leq$.

5.2.3 Characterisation using apex graphs and closures under disjoint unions

In this section we establish a third alternative characterisation of the class of graphs with elimination distance k to a class $\mathcal{C}$, using apex graphs and closures under disjoint unions.

First we give the necessary definitions and establish notation. The closure of a binary relation R on a set X is the transitive relation R+ on set X such that R+ contains R and R+ is minimal

Definition 5.2.5. Let G, G' be graphs. The disjoint union of G and G', denoted $G \sqcup G'$, is the graph with vertex set

$$V(G) \times \{0\} \cup V(G') \times \{1\}$$

and edge set

$$\{(u,0)(v,0) \mid uv \in E(G)\} \cup \{(u,1)(v,1) \mid uv \in E(G')\}.$$

Definition 5.2.6. Let $\mathcal{C}$ be a class of graphs. The *closure of $\mathcal{C}$ under taking disjoint unions* is the minimal class of graphs $\overline{\mathcal{C}}$ that contains $\mathcal{C}$, and has the additional property that for any two graphs $G, G' \in \overline{\mathcal{C}}$ there is $H \in \overline{\mathcal{C}}$ with $H \cong G \sqcup G'$.

The special case of a graph with deletion distance 1 to a class $\mathcal{C}$ is called an *apex graph* over $\mathcal{C}$ in this chapter.

Definition 5.2.7. We say that a graph G is an *apex graph* over a class $\mathcal{C}$ of graphs if there is a vertex $v \in V(G)$ such that the graph $G \setminus \{v\} \in \mathcal{C}$. The class of all apex graphs over $\mathcal{C}$ is denoted $\mathcal{C}^{\text{apex}}$. We say $\mathcal{C}^{\text{apex}}$ is an *apex graph class over $\mathcal{C}$*.

Remark. We will sometimes refer to such a vertex v as *apex vertex*, or just *apex*.

The characterisation of elimination distance introduced in this section is in terms of the iterated closure of $\mathcal{C}$ under the operation of disjoint unions and taking the class of apex graphs. We introduce a piece of notation for this in the next definition.

Definition 5.2.8. For a class of graphs $\mathcal{C}$, let $\mathcal{C}_0 := \mathcal{C}$, and $\mathcal{C}_{i+1} := \overline{\mathcal{C}_i}^{\text{apex}}$.

We show next that the class $\mathcal{C}_k$ is exactly the class of graphs at elimination distance k from $\mathcal{C}$.

Proposition 5.2.9. *Let $\mathcal{C}$ be a class of graphs and $k \geq 0$. Then $\mathcal{C}_k$ is the class of all graphs with elimination distance at most k to $\mathcal{C}$.*

Proof. We prove this by induction. Only the graphs in $\mathcal{C}$ have elimination distance 0 to $\mathcal{C}$, so the statement holds for $k = 0$.

Suppose the statement holds for $k \geq 0$. If $G \in \mathcal{C}_{k+1}$, then G is a disjoint union of graphs $G_1, \ldots, G_s$ from ${\mathcal{C}_k}^{\text{apex}}$, so we can remove at most one vertex from each of the G_i and obtain a graph in $\mathcal{C}_k$. Thus the elimination distance of G to $\mathcal{C}_k$ is 1, and by induction the elimination distance to $\mathcal{C}$ is $k + 1$.

Conversely, if G has elimination distance $k+1$ to $\mathcal{C}$, then we can remove a vertex from each component of G to obtain a graph G' with elimination distance k to $\mathcal{C}$. Using the induction hypothesis each component of G' is in $\mathcal{C}_k$, and thus $G \in \mathcal{C}_{k+1}$. □

5.3 Elimination distance & classes of bounded expansion

In this section we use one of the alternative characterisations established in the last section to prove that if $\mathcal{C}$ is a class of graphs with bounded expansion (cf. Definition 2.3.31), then the class of graphs with fixed elimination distance to $\mathcal{C}$ also has bounded expansion.

Proposition 5.3.1. *Let $\mathcal{C}$ be a class of graphs with bounded expansion and let $d \geq 0$. Then the class of graphs with elimination distance d to $\mathcal{C}$ also has bounded expansion.*

Proof. Let $\mathcal{C}$ be a class of graphs with bounded expansion. We use the characterisation of bounded expansion classes via low tree-depth colourings (see Definition 2.3.32).

Recall from Theorem 2.3.33 that $\mathcal{C}$ has bounded expansion if, and only if, for every integer p there is an integer N_p such that for any graph $G \in \mathcal{C}$ we have $\chi_p(G) \leq N_p$. We use the characterisation of elimination distance defined above in Section 5.2.2 to split G into two parts and find a low tree-depth colouring for each of them.

Fix an integer p. Then there is an integer N such that for any graph $G \in \mathcal{C}$ we have $\chi_p(G) \leq N$. Let G' be a graph with elimination distance d to $\mathcal{C}$. By Proposition 5.2.3 there is an elimination order $\leq$ on G' such that $\leq$ is an elimination order of height d on the graph T induced by the non-$\leq$-maximal vertices. All components of the graph U induced by the $\leq$-maximal vertices are contained in $\mathcal{C}$. Thus there is a tree-depth colouring of U using at most N colours. Observe that it is not a problem to reuse colours from one component in another, because the components are not connected.

We colour the non-$\leq$-maximal vertices as follows: We use d fresh colours to colour every level (see Definition 2.4.8) of $\leq$ in one of the d colours. Thus every colour induces a subgraph of T with tree-depth just 1, and every combination of $j \leq d$ colours induces a subgraph with tree-depth j.

Combining the colourings of both parts gives us a colouring of G' that uses $N + d$ colours. Note that if S is a set of colours, and S' is a set of colours obtained by adding to S one of the d colours used for the non-$\leq$-maximal vertices, then the tree-depth of the

subgraph induced by S' is at most one more than the tree-depth of the subgraph induced by S. Thus every set of colours S with $|S| \leq p$ induces a subgraph of G' with tree-depth at most $|S|$, and therefore $\chi_p(G') \leq N + d$.

So for every integer p, and every graph G' with elimination distance d to $\mathcal{C}$, there is an integer N_p ($N + d$ above) such that $\chi_p(G') \leq N_p$. This shows that the class of graphs $\mathcal{C}_d$ with elimination distance d to $\mathcal{C}$ has bounded expansion. □

5.4 Elimination distance to the class of edgeless graphs

The class of edgeless graphs is the class of all the graphs where every vertex is isolated. Here we consider the elimination distance to that class and show that this gives us yet another alternative characterisation of tree-depth, underlining the versatility of the parameter.

If $\mathcal{C}$ is the class of edgeless graphs, then Definition 5.1.1 just yields the definition of the tree-depth of G (cf. Definition 4.0.4). So we have the following.

Proposition 5.4.1. *Let $\mathcal{C}$ be the class of edgeless graphs and let $k > 0$. Let G be a graph with at least one vertex. Then G has elimination distance $k - 1$ to $\mathcal{C}$ if, and only if, G has tree-depth k.*

Proof. We prove this by induction on k. Let $k = 1$. A graph G has elimination distance 0 to $\mathcal{C}$ if all its components are contained in $\mathcal{C}$. Since $\mathcal{C}$ is the class of edgeless graphs this means none of the components contain an edge. So the tree-depth of G is 1. Conversely, if $td(G) = 1$, then all the vertices of G are isolated, and thus $ed_{\mathcal{C}}(G) = 0$.

Let $k > 1$ and assume the statement holds for smaller values. If G has elimination distance $k-1$ to $\mathcal{C}$, there is a vertex $v \in V(G)$ such that $G \setminus \{v\}$ has elimination distance $k - 2$ to $\mathcal{C}$. By the induction assumption $G \setminus \{v\}$ has tree-depth $k - 1$ and thus G has tree-depth k. The converse is similar: If G has tree-depth k, then there is a vertex $w \in V(G)$ such that $G \setminus \{w\}$ has tree-depth $k - 1$. By the induction assumption $G \setminus \{w\}$ has elimination distance $k - 2$ to $\mathcal{C}$ and thus, by definition, $ed_{\mathcal{C}}(G) = k - 1$. □

Remark. We just exclude the empty graph because in that case both tree-depth and elimination distance to the class of edgeless graphs are 0.

From that perspective elimination distance is a natural generalisation of tree-depth. In general elimination distance to a class $\mathcal{C}$ seems to be a much more powerful parameter than tree-depth; for example we show in Chapter 8 that tree-depth is first-order definable. It is doubtful that the same holds for elimination distance to any class $\mathcal{C}$, or even to any non-trivial class – but it remains an open question.

From Proposition 5.4.1 it also follows that the tree-depth of a graph is an upper bound for the elimination distance to any class $\mathcal{C}$:

Proposition 5.4.2. *Let $\mathcal{C}$ be a (non-empty) class of graphs closed under taking subgraphs. If a graph G has tree-depth k, then G has elimination distance at most k to $\mathcal{C}$.*

Proof. Let $\mathcal{C}$ be a class of graphs closed under taking subgraphs that contains at least one graph. Let G be a graph with $td(G) = k$. If $k = 0$, G is the empty graph and since $\mathcal{C}$ is closed under taking subgraphs we have $G \in \mathcal{C}$ and thus G has elimination distance 0 to $\mathcal{C}$.

If $k > 0$, by Proposition 5.4.1 we have that G has elimination distance $k-1$ to the class of edgeless graphs. Thus G has elimination distance at most k to the class containing only the empty graph. Since $\mathcal{C}$ contains the class containing only the empty graph, this proves that G has elimination distance to $\mathcal{C}$ at most k. □

5.4.1 Elimination distance & generalised tree-depth

In Section 4.4 we discussed the parameter generalised tree-depth, introduced by Bouland et al. [23]. Generalised tree-depth generalises both tree-depth and max leaf number in the sense that it is a lower bound for both. What is the relationship of this parameter to elimination distance?

Recall from Section 4.4.1 that the generalised tree-depth of a graph is defined via a Cop and Robber game, where the Robber player loses when she is confined to a path (with cops at the endpoints) or a cycle that is disconnected (by cops) from the rest of the graph. We will see that this gives us almost the elimination distance to paths and cycles (i.e. graphs of degree at most 2). However, since it is not enough to isolate the robber on a path, and the robber is only caught after a cop is placed at each end point, there is an additional restriction to consider.

Consider for example a graph G that is a disjoint union of paths. Clearly G has elimination distance 0 to the class $\mathcal{C}_2$ of graphs of degree at most 2. However, the rules of the game that defines generalised tree-depth force the Cop player to place two cops on the path where the Robber player puts the robber. The generalised tree-depth of G is thus 2. Note however that this is the worst case: It is never more than 2 and can be equal. So generalised tree-depth is 'almost' the elimination distance to $\mathcal{C}_2$.

In general if we have a graph G with elimination distance k to $\mathcal{C}_2$ there is an elimination order $\leq$ to $\mathcal{C}_2$ of height k on G by Proposition 5.2.3. The Cop player can use the order as a strategy so that after k rounds, there are only components of maximal degree at most 2 left. If there are paths with no cops on them at this stage the Cop player might need two more rounds to win the game. Conversely, the strategy of the Cop player defines an elimination order to $\mathcal{C}_2$. It follows that the elimination distance to $\mathcal{C}_2$ is a lower bound for generalised tree-depth, and generalised tree-depth can be at most 2 more than elimination distance to $\mathcal{C}_2$:

Proposition 5.4.3. *Let $\mathcal{C}_2$ be the class of graphs of degree at most 2. Then for any graph G we have the following:*

$$\underset{\mathcal{C}_2}{ed}(G) \leq gtd(G) \leq \underset{\mathcal{C}_2}{ed}(G) + 2.$$

In the next section we explore the more general case where $\mathcal{C}$ is a class of bounded degree. We choose a different way to define an elimination order and show that it gives us an alternative characterisation of elimination distance to classes of bounded degree.

5.5 Elimination distance to bounded degree classes

In this section we study elimination distance to graph classes of bounded degree. The class of graphs with degree bounded by a constant d is interesting, because many problems become tractable when restricted to it. A very prominent example is graph isomorphism, which is an important application and the subject of Chapter 6. Figure 1.1 shows the relationship of elimination distance to bounded degree to other distance parameters.

By defining $\mathcal{C}$ to be graphs of degree bounded by a constant d, we obtain a special case of Definition 5.1.1. To be less verbose, we define the *elimination distance to degree d* of a graph G, abbreviated $ed_d(G)$, to mean the elimination distance to the class of graphs with maximal degree d.

In this section we introduce another characterisation of this parameter; if a graph has elimination distance k to degree d, we define an order equivalent to the elimination order to $\mathcal{C}$, where $\mathcal{C}$ is the class of graphs with maximal degree d. So this definition is similar to that given in Definition 5.2.1, but there is an important difference: because maximal degree is a local property we do not need to refer to components here and instead state the definition in local terms of neighbourhoods of incomparable vertices. This is the definition we work with in Chapter 6.

Just as in Definition 5.2.1, the idea is that if G is a connected graph we can choose a vertex v so that deleting it strictly reduces the elimination distance to degree d. We then make $v \leq u$ for all vertices u and vertices in distinct components of $G \setminus v$ are made incomparable with respect to $\leq$. We proceed in this fashion until the remaining vertices have degree at most d (and are incomparable, and indeed $\leq$-maximal). Note that any component of the remaining vertices has maximal degree d, and the vertices of $G \setminus C$ that are neighbours of vertices in C must be linearly ordered by $\leq$ and they all precede the vertices of C in the order $\leq$. Thus, if we associate with every vertex v, the set S_v of neighbours of v that are $\leq$-incomparable with v we can note the following: S_v has at most d elements for any v; if S_v is not empty, then v must be $\leq$-maximal; and if $u \in S_v$ then u and v must have the same $\leq$-predecessors. This can be taken as the defining property of an elimination order to degree d, as below.

Definition 5.5.1. A tree order $\leq$ on $V(G)$ is an *elimination order to degree d* for G if for each $v \in V(G)$ the set

$$S_v := \{u \in V(G) \mid uv \in E(G) \text{ and } u \not\leq v \text{ and } v \not\leq u\}$$

satisfies either:

- $S_v = \emptyset$; or
- v is $\leq$-maximal, $|S_v| \leq d$, and for all $u \in S_v$, we have $\{w \mid w < u\} = \{w \mid w < v\}$.

Remark. Note that if $S_v = \emptyset$ for all $v \in V(G)$, then an elimination order to degree d is just an elimination order, in the sense of Definition 2.4.9.

Proposition 5.5.2. *A graph G has $ed_d(G) \leq k$ if, and only if, there is an elimination order to degree d of height k for G.*

Proof. Suppose first that $ed_d(G) \leq k$. We proceed by induction on k. If $k = 0$, then G has no vertex of degree larger than d and we define the elimination order $\leq$ to be the identity relation on $V(G)$. Then every $v \in V(G)$ is maximal, we have $|S_v| \leq d$, and for all $u \in S_v$ we have $\{w \mid w < u\} = \emptyset = \{w \mid w < v\}$.

Suppose $k > 0$ and the statement is true for smaller values. If G is not connected, we apply the following argument to each component. So in the following we assume that G is connected. Thus, there is a vertex $a \in V(G)$ such that the components $C_1, \ldots, C_r$ of $G \setminus a$ all have $ed_d(C_i) \leq k - 1$. So by the induction hypothesis each C_i has a tree order

$\leq_i$ to degree d of height at most $k-1$ with the properties in Definition 5.5.1. For each $v \in V(C_i)$ define

$$S_v^i := \{u \in V(C_i) \mid uv \in E(G) \text{ and } u \not\leq_i v \text{ and } v \not\leq_i u\}.$$

Let

$$\leq := \{(a, w) \mid w \in V(G)\} \cup \bigcup_i \leq_i .$$

Then $\leq$ is clearly a tree order for G. Note that $S_a = \emptyset$. Let $v \in V(G) \setminus a$ be a vertex different from a, say $v \in V(C_i)$. Note $S_v^i = S_v$. If $S_v \neq \emptyset$, then v is $\leq_i$-maximal, and thus also $\leq$-maximal. Moreover, $|S_v^i| = |S_v| \leq d$. Lastly for any $u \in S_v$:

$$\{w \mid w < u\} = \{a\} \cup \{w \mid w <_i u\} = \{a\} \cup \{w \mid w <_i v\} = \{w \mid w < v\}.$$

Conversely assume there is an elimination order $\leq$ to degree d of height k for G. We again proceed by induction on k. If $k = 0$, then $\leq$ is empty and all vertices v in G are $\leq$-maximal. Thus, S_v contains all neighbours of v and therefore v has at most d neighbours. In other words, $\Delta(G) \leq d$ as required.

Now, suppose $k > 0$. If v is an $\leq$-minimal element then for any u for which there is a path from v to u, we must have $v \leq u$. Thus, if G is connected, there is a unique $\leq$-minimal element v. Note that $\leq$ restricted to a component C of $G \setminus v$ has height $k-1$ and thus by the induction hypothesis we have that $ed_d(C) \leq k-1$. If G is not connected, this argument can be applied to each component of G. □

Recall that Proposition 5.2.3 and Proposition 5.2.4 allow us to split a graph with an elimination order to degree d in two parts: one of low degree, and one with an elimination order defined on it.

The alternative characterisation in terms of the elimination order to degree d established above, and the ability to split the graph, are algorithmically very useful. As an application, in Chapter 6, we use these to establish that Graph Canonisation (and thus Graph Isomorphism) is FPT parameterized by elimination distance to bounded degree. This is done by constructing a *canonical* elimination order to degree d of G, based on an elimination order of a graph we call the *torso* of G, which contains the high-degree vertices of G, along with some additional edges.

chapter 6

Graph Isomorphism

The Graph Isomorphism (GI) problem is the problem of determining, given a pair of graphs G and H (as adjacency matrices), whether they are isomorphic, i.e., whether there is a bijective function $\varphi : V(G) \to V(H)$ such that $uv \in E(G)$ if and only if $\varphi(u)\varphi(v) \in E(H)$. The precise definitions can be found in Section 2.3.1.

More informally, GI asks whether the two given graphs are structurally equal, i.e. if we can rename the vertices of G to obtain H, or whether given a drawing of G we can rearrange the nodes in a way that gives us H. For example, the following is a mapping from the graph in Figure 6.1 to the graph in Figure 6.2 that preserves the edge relation in both directions:

$$
\begin{aligned}
a &\mapsto 1 \\
b &\mapsto 6 \\
c &\mapsto 8 \\
d &\mapsto 3 \\
g &\mapsto 5 \\
h &\mapsto 2 \\
i &\mapsto 4 \\
j &\mapsto 7
\end{aligned}
$$

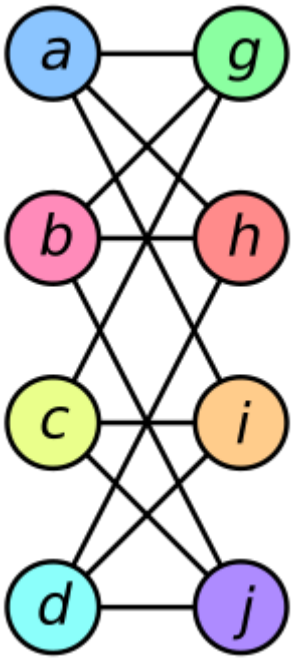

Figure 6.1: A graph.

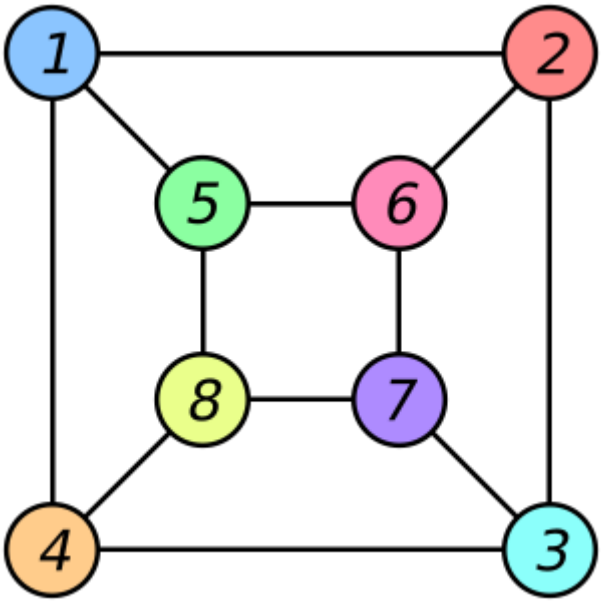

Figure 6.2: Another graph.

The problem is formally defined as follows:

Graph Isomorphism
Input: Two graphs G and H.
Problem: Are G and H isomorphic?

GI is clearly in NP (an isomorphism is a succinct certificate). For a problem in NP it has an unusual status in complexity theory: it is neither known to be in P nor known to be NP-complete – one of the few natural problems for which this is the case; see the discussion below.

A naïve approach to solve GI would try out all of the $n!$ possible mappings between $V(G)$ and $V(H)$ and would take exponential time. Despite considerable effort[1], progress on the problem has been slow. However, while the problem has not been satisfactorily resolved, the intense study of the problem produced a large number of algorithmic techniques and new concepts in complexity theory. Until recently the best known algorithm was due to Babai and Luks [12] and runs in time $2^{\sqrt{n \log n}}$. However, it seems the situation has changed: Very recently Babai [11] announced an algorithm that solves graph isomorphism in quasi-polynomial time, i.e. in time $O(2^{\log^c n})$ for some constant $c > 1$. Both these results are mostly based on group theoretic approaches.

This puts GI almost in the class P, but obstacles remain. The problem is very closely related to the Group Isomorphism problem, which is in fact a special case. Group Isomorphism asks whether two groups $(G, \cdot)$ and $(H, \circ)$, given in the form of their Cayley tables, are isomorphic, i.e. if there is a bijection $\varphi : G \to H$ such that for all $a, b \in G$ we have $\varphi(a) \circ \varphi(b) = \varphi(a \cdot b)$. In the 1970s Tarjan[2] and Lipton et al. [116] independently observed that the Group Isomorphism problem can be solved in time $n^{\log n + O(1)}$ – exploiting the fact that a group of size n is generated by $\log n$ generators. However, despite significant efforts to improve this 'trivial' bound, and despite the much richer structure available for groups, there has not been any real progress on the Group Isomorphism problem. This seems to be the current frontier for the general Graph Isomorphism problem.

Despite the lack of success in showing that the problem is contained in P, it seems unlikely that it is NP-hard. GI is known to be contained in co-AM and in level L_2^p of the low hierarchy [13, 146], which means that if it were NP-hard, then the polynomial hierarchy would collapse to its second level, which most complexity theorists do not believe to be likely. This is the reason for the widespread belief that GI is in fact contained in P – a major open problem at the moment.

GI is also one of the few 'natural' candidates[3] for NP-intermediate problems. The class NP-intermediate contains the problems in NP that are neither in P nor NP-hard. Ladner [111] used a diagonalization based technique to show that such problems exist if $\mathsf{P} \neq \mathsf{NP}$; in fact, he proves there are infinitely many. However, despite this fact there are not many natural problems that are potential members of NP-intermediate. Most natural problems in NP can either be solved in polynomial time (and are thus in P) or can be shown to be NP-hard.

[1] The problem has been called "Isomorphism Disease" because of the time wasted by researchers [84, 139].

[2] Tarjan communicated the result in private correspondence.

[3] The other problem that is usually mentioned in this context is Factoring. There are a few more candidates for NP-intermediate problems depending on where one decides to draw the line between natural and artificial.

Graph Isomorphism has a large number of applications in practice. One of the problems that initiated the practical study of the problem was the need to identify canonical names for molecules to find related work on chemical compounds [62]. Other applications include the analysis of circuits in hardware design [136], optical character recognition [156], the semantic web [34], improving the PageRank algorithm [20, 105], autonomous repair of sensor networks [37], graph mining of social networks [150], de-anonymizing social networks [14], and even fighting terrorism [90] and paleontology [144]. The most significant application is possibly the elimination of symmetries in instances of the boolean satisfiability problem (SAT) [8], that is used to speed up industrial SAT solvers. High-performance SAT solvers have an increasingly important role in solving combinatorial problems in industry.

Despite its uncertain status in complexity theory, GI can be considered solved for most of these practical applications. Powerful graph isomorphism software, such as nauty by McKay [126, 127], Traces by Piperno [127, 137], saucy by Darga et al. [46] and conauto by Lopez-Presa [119] can quickly solve most instances that show up in applications. All these programs are based on the same idea: they compute a canonical labeling of the graph, employing several heuristics to speed up the process.

Despite the fact that researchers have not found a polynomial-time algorithm for GI on general graphs, polynomial-time algorithms are known for a variety of special classes of graphs. Most of the published research on the problem over the last 40 years is concerned with this question. Many of these lead to natural parameterizations of GI by means of structural parameters of the graphs which can be used to study the problem from the point of view of parameterized complexity. For instance, it is known that GI is in XP (cf. Definition 2.2.7) parameterized by the genus of the graph [70, 128], by maximum degree [12, 120] and by the size of the smallest excluded minor [138], or more generally, the smallest excluded topological minor [93].

For each of the parameters mentioned in the last paragraph it remains an open question whether Graph Isomorphism is FPT. On the other hand, GI has been shown to be FPT when parameterized by eigenvalue multiplicity [60], tree distance width [158], the maximum size of a simplical component [151, 153] and minimum feedback vertex set [108]. Bouland et al. [23] showed that the problem is FPT when parameterized by the tree-depth of a graph and extended this result to a parameter they termed generalised tree depth (cf. Section 5.4.1) that also proves that the problem is FPT parameterized by max leaf number. In a recent advance on this, Lokshtanov et al. [118] have shown that graph isomorphism is also FPT parameterized by tree-width.

The result we present in this chapter extends the results of Bouland et al. and is incomparable with that of Lokshtanov et al. We show that graph canonisation is FPT parameterized by elimination distance to degree d, for any constant d. The structural graph parameter we use is an instance of elimination distance introduced in Chapter 5, which is a general paradigm that is also of interest in the context of other graph problems. It should be noted that the class of all graphs with degree bounded by d (for $d \geq 3$) has unbounded tree-width and the same is *a fortiori* true of graphs with elimination distance k to degree d. It should also be noted that generalised tree depth in is an upper bound for the special case of elimination distance to degree 2 (cf. Section 5.4.1).

To motivate the choice of elimination distance, and the choice of the class of graphs of bounded degree, in the case of GI, consider the simplest notion of distance for a graph G to a class where GI is tractable: the number k of vertices of G that must be deleted

to obtain a graph with no edges (cf. Section 3.1). This is, of course, just the size of a minimal vertex cover in G (cf. Section 3.2) and is a parameter that has been extensively studied (see for instance [67]). Indeed, it is also quite straightforward to see that GI is FPT when parameterized by vertex cover number.

Consider two ways this observation might be strengthened. The first is to relax the notion of what we consider to be tractable, i.e. to consider a greater class of graphs where GI is still tractable. For instance, as there is, for each d, a polynomial time algorithm deciding GI among graphs with maximum degree d, we may take this as our base case, i.e. as the class of graphs we measure the distance from. We then parameterize G by the number k of vertices that must be deleted to obtain a subgraph of G with maximum degree d. This yields the parameter deletion distance to bounded degree, which we consider in Section 6.2 below.

Alternatively, we can relax the notion of "distance" to that of "elimination distance" introduced in Chapter 5, so that rather than considering the sequential deletion of k vertices, we consider the recursive deletion of vertices in a tree-like fashion.

In our main result of this chapter, established in Section 6.3, we combine these two approaches by parameterizing G by the elimination distance to the class of graphs with maximal degree d (or just elimination distance to degree d). We show that, for any fixed d, this gives a structural parameter on graphs for which graph canonisation is FPT.

We begin by recalling some result for isomorphism on bounded degree graphs in the next section.

6.1 Isomorphism on bounded-degree graphs

In this section we collect some well known results about isomorphism tests and canonisation of bounded degree graphs that we use below. Luks [120] shows that isomorphism of bounded-degree graphs is decidable in polynomial time. It is well-known that this result extends, by an easy reduction, to *coloured* graphs of bounded-degree. Though this reduction is folklore, for the sake of completeness, we present it here explicitly.

Proposition 6.1.1. *The isomorphism problem for coloured graphs of maximum degree d can be reduced to the isomorphism problem of graphs of maximum degree $d+2$ in polynomial time.*

Proof. Let G, G' be graphs and let $c, c' : V(G) \to \{1, \ldots, k\}$ be colourings of G, G' respectively for some $k \in \mathbb{N}$.

We define H to be the graph whose vertices include $V(G)$ and, additionally, for each $v \in V(G)$, $c(v)+1$ new vertices $u^v_1, \ldots, u^v_{c(v)+1}$. The edges of H are the edges $E(G)$ plus additional edges so that the vertices v and $u^v_1, \ldots, u^v_{c(v)+1}$ form a simple cycle of length $c(v)+2$. We obtain H' in a similar way from G'.

We claim that $G \cong G'$ if, and only if, $H \cong H'$. Clearly, if $G \cong G'$ and φ is an isomorphism witnessing this, it can be extended to an isomorphism from H to H' by mapping u^v_i to $u^{\varphi(v)}_i$. For the converse, suppose $H \cong H'$ and let $\varphi : H \to H'$ be an isomorphism. We use it to define an isomorphism φ' from G to G'. Note that, if $v \in V(G)$ is not an isolated vertex of G, then it has degree at least 3 in H. Since $\varphi(v)$ has the same degree, it is in $V(G')$, and we let $\varphi'(v) = \varphi(v)$. If v is an isolated vertex of G, then its component in H is a simple cycle of length $c(v)+2$. The image of this component under φ is a simple cycle of H' which must contain exactly one vertex v' of $V(G')$. We let

$\varphi'(v) = v'$. It is easy to see that there is an edge between v_1, v_2 in G if, and only if, there is an edge between $\varphi'(v_1)$ and $\varphi'(v_2)$ in G'. To see that φ' also preserves colours, note that φ must map the cycle containing $u^v_{c(v)}$ to the cycle containing $u^{\varphi'(v)}_{c(\varphi'(v))}$ and therefore $c(v) = c(\varphi'(v))$.

Note that if G and G' are graphs of maximum degree d, then H, H' are graphs of maximum degree $d + 2$. □

As Luks [120] proves that isomorphism of bounded degree graphs can be decided in polynomial time, we have the following:

Theorem 6.1.2. *We can test in polynomial time whether two (coloured) graphs with maximum degree bounded by a constant are isomorphic.*

In 1983 Babai and Luks [12] give a polynomial-time canonisation algorithm for bounded degree graphs. Just as above we can reduce canonisation of coloured bounded degree graphs to the bounded degree graph canonisation problem.

Theorem 6.1.3. *Let $\mathcal{C}$ be a class of (coloured) bounded degree graphs closed under isomorphism. Then there is a canonical form F for $\mathcal{C}$ that allows us to compute $F(G)$ in polynomial time.*

6.2 Deletion distance to bounded degree

We first study the notion of deletion distance to bounded degree and establish in this section that graph isomorphism is FPT with this parameter. Though the result in this section is subsumed by the more general one in Section 6.3.2 and also by a result of Kratsch and Schweitzer [108], it provides a useful warm-up. It also provides a specific algorithm for the problem, while Kratsch and Schweizer [108, Theorem 1] solve the more general problem of graph isomorphism given two graphs that are k vertex deletions away from a graph class characterised by excluded induced subgraphs.

The notion of deletion distance to bounded degree is a particular instance of the general notion of distance to triviality introduced by Guo et al. [95]. So from that perspective, and in the context of graph isomorphism, we have chosen triviality to mean graphs of maximum degree at most d.

In the present warm-up we only give an algorithm for the graph isomorphism problem, though the result easily holds for canonisation as well (and this follows from the more general result in Section 6.3.2).

Recall the notion of deletion distance to a class $\mathcal{C}$ introduced in Definition 3.1.1. We introduce some notation that allows us to be less verbose in this section.

Definition 6.2.1. We say that a graph G has *deletion distance k to degree d* if it has deletion distance k to the class of graphs of maximum degree d. We call a set of vertices $\{v_1, \ldots, v_k\}$ such that $G \setminus \{v_1, \ldots, v_k\}$ has maximum degree d a *d-deletion set*.

We make the following observations:

Remark. To say that G has deletion distance 0 from degree d is just to say that G has maximum degree d. Also note that if $d = 0$, then a d-deletion set is just a vertex cover and the minimum deletion distance the vertex cover number of G.

We show that isomorphism is fixed-parameter tractable on graphs parameterized by k with fixed degree d; in particular we give a procedure that computes in linear time a set U of vertices of size polynomial in k so that any deletion set must be found in U if one exists. This then allows for exhaustive search over all potential isomorphisms. We call U a *bounding set* for the deletion set. The construction is reminiscent of a kernelization result in [64] for the problem of determining whether a graph has deletion distance k to degree d, though that does not yield a bounding set containing all deletion sets.

Theorem 6.2.2. *For any graph G and integers $d, k > 0$, we can identify in linear time a subgraph G' of G, a set of vertices $U \subseteq V(G')$ with $|U| = O(k(k+d)^2)$ and a $k' \leq k$ such that: G has deletion distance k to degree d if, and only if, G' has deletion distance k' to d and, moreover, if G' has deletion distance at most k', then any minimum size d-deletion set for G' is contained in U.*

Proof. Let $A := \{v \in V(G) \mid \deg(v) > k+d\}$. Now, if R is a minimum size d-deletion set for G and G has deletion distance at most k to degree d, then $|R| \leq k$ and the vertices in $V(G \setminus R)$ have degree at most $k+d$ in G. So $A \subseteq R$. This means that if $|A| > k$, then G must have deletion distance greater than k to degree d and in that case we let $G' := G, k' := k$ and $U = \emptyset$.

Otherwise let $G' := G \setminus A$ and $k' := k - |A|$. We have shown that every d-deletion set of size at most k must contain A. Thus G has deletion distance k to degree d if, and only if, G' has deletion distance k' to degree d.

Let $S := \{v \in V(G') \mid \deg_{G'}(v) > d\}$ and $U := S \cup N_{G'}(S)$. Let $R' \subseteq V(G')$ be a minimum size d-deletion set for G'. We show that $R' \subseteq U$. Let $v \notin U$. Then by the definition of U we know that $\deg_{G'}(v) \leq d$ and all of the neighbours of v have degree at most d in G'. So if $v \in R'$, then $G \setminus (R' \setminus \{v\})$ also has maximal degree d, which contradicts the assumption that R' is of minimum size. Thus $v \notin R'$.

Note that the vertices in $G' \setminus (R' \cup N(R'))$ have the same degree in G' as in G and thus all have degree at most d. So $S \subseteq R' \cup N(R')$ and thus $|U| \leq k' + k'(k+d) + k'(k+d)^2 = O(k(k+d)^2)$.

Finally, the sets A and U defined as above can be found in linear time, and G', k' can be computed from A in linear time. □

Remark. It may be noted that the set U is canonical in the sense that if there is an isomorphism from G to a graph H, this must take U to the corresponding set in H. But, this is not essential to our argument below. What is important is that U contains *all* possible deletion sets for G. In particular, if $U = \emptyset$ and $k' > 0$, then there are no d-deletion sets of size at most k'.

Next we see how the bounding set U can be used to determine whether two graphs with deletion distance k to degree d are isomorphic by reducing the problem to isomorphism of coloured graphs of degree at most d.

In the following suppose we are given two graphs G and H, with d-deletion sets $S := \{v_1, \ldots, v_k\}$ and $T := \{w_1, \ldots, w_k\}$ respectively. Further suppose that the map $v_i \mapsto w_i$ is an isomorphism on the induced subgraphs $G[S]$ and $H[T]$. We can then test if this map can be extended to an isomorphism from G to H using Theorem 6.1.3. To be precise, we define the coloured graphs G' and H' which are obtained from $G \setminus S$ and $H \setminus T$ respectively, by colouring vertices. A vertex $u \in V(G')$ gets the colour $\{i \mid v_i \in N_G(u)\}$, i.e. the set of indices of its neighbours in S. Vertices in H' are similarly coloured by the

sets of indices of their neighbours in T. It is clear that G' and H' are isomorphic if, and only if, there is an isomorphism between G and H, extending the fixed map between S and T. The coloured graphs G' and H' have degree bounded by d, so Theorem 6.1.3 gives us a polynomial-time isomorphism test on these graphs.

Now, given a pair of graphs G and H which have deletion distance k to degree d, let A and B be the sets of vertices of degree greater than $k+d$ in the two graphs respectively. Also, let U and V be the two bounding sets in the graphs obtained from Theorem 6.2.2. Thus, any d-deletion set in G contains A and is contained in $A \cup U$ and similarly, any d-deletion set for H contains B and is contained in $B \cup V$. Therefore to test G and H for isomorphism, it suffices to consider all k-element subsets S of $A \cup U$ containing A and all k-element subsets T of $B \cup V$ containing B, and if they are d-deletion sets for G and H, check for all $k!$ maps between them whether the map can be extended to an isomorphism from G to H. As d is constant this takes time $O^*\left(\binom{k^3}{k}^2 \cdot k!\right)$, which is $O^*\left(2^{7k \log k}\right)$. (See the remark after Definition 2.2.3 for an explanation of the O^* notation.)

6.3 Elimination distance to bounded degree

We now consider elimination distance to bounded degree and show in this section that Graph Isomorphism is FPT with respect to this parameter. The result in this section subsumes the result in Section 6.2.

Just as deletion distance in the previous section, the notion of elimination distance to bounded degree can also be interpreted as a particular instance of the notion of distance to triviality introduced by Guo et al. [95]. From that perspective, and in the context of graph isomorphism, we have again chosen triviality to mean graphs of bounded maximum degree. However, now we consider the more general *elimination distance* as the distance.

The main idea of the proof is to identify canonical structures in the graph. It is easy to see that in a graph G that has elimination distance k to degree d, all vertices of sufficiently high degree have to be 'eliminated', i.e. they have to be non-maximal elements of the elimination order. However, it may be the case that in an optimal elimination order some low-degree vertices may also need to be non-maximal elements of the elimination order. A key idea in our proof is to show that G contains a canonically defined set of vertices C (which we call its torso), including the high-degree vertices so that if G has elimination distance k to degree d, then we can reduce the maximum degree to d by 'eliminating' just vertices in C, and moreover this yields an elimination order whose height is bounded by a function of k and d. This is established in Section 6.3.1.

With this characterisation established, we are able to present the canonisation algorithm in Section 6.3.2. The central technique used here is inspired by Lindell's tree canonisation algorithm [115]. We define an order $\sqsubseteq$ on isomorphism types of coloured graphs equipped with an elimination order recursively on the height of this order, with the base case being a canonical order on coloured graphs of bounded degree. We then show that given a graph G, separated into its torso C and a low-degree part, we can construct an elimination order for G (i.e. a tree-depth decomposition of C) which is $\sqsubseteq$-minimal. We use the result of Bouland et al. [23] on canonisation of bounded tree-depth graphs to bound the branching in the search and establish that canonisation is FPT.

6.3.1 Canonical separation into high and low degree parts

The aim of this section is to show that if a graph G has elimination distance k to degree d, then there is an elimination order to degree d whose height is still bounded by a function of d and k and in which the set of non-maximal elements is *canonical*. To be precise, we identify a graph which we call the d-degree torso of G, which contains all the vertices of G of degree more than d and has additional edges to represent paths between these vertices that go through the rest of G. We show that this torso necessarily has tree-depth bounded by a function of k and d and an elimination order witnessing this can be extended to an elimination order to degree d of G.

The result is established through a series of lemmas. A pattern of construction that is repeatedly used here is that we define a certain set A of vertices of G and construct an elimination order of $G[A]$. It is then shown that extending the order by making all vertices in $V(G) \setminus A$ maximal yields an elimination order to degree d of G. Necessarily, in this extended order, all the non-maximal elements are in A.

We first do this for graphs of degree bounded by $k + d$. To be precise, the following lemma establishes that if G has elimination distance k to degree d and moreover the maximum degree of G is at most $k + d$, then we can construct an alternative elimination order to degree d on G in which all the vertices of degree greater than d are included in the non-maximal subgraph and the height of the new elimination order to degree d is still bounded by a function of k and d.

Lemma 6.3.1. *Let G be a graph with maximum degree $\Delta(G) \leq k + d$. Let $\leq$ be an elimination order to degree d of height k of G with non-maximal subgraph H, and let $A = V(H) \cup \{v \in V(G) \mid \deg_G(v) > d\}$. Then G has an elimination order $\sqsubseteq$ to degree d of height at most $k(k + d + 1)$ for which A is the set of non-maximal elements.*

Proof. Let G, H, A and $\leq$ be as in the statement of the lemma. We adapt $\leq$ to an elimination order $\sqsubseteq$ of $G[A]$.

For each $w \in A \setminus V(H)$ let C_w be the component of $G \setminus V(H)$ that contains w. Note that $N(C_w) \neq \emptyset$, because $\deg(w) > d$, so at least one vertex in H must be adjacent to w. By Definition 5.5.1, the vertices in $N(C_w)$ are linearly ordered by $\leq$. We write $b(w)$ to denote the unique $\leq$-maximal element of $N(C_w)$ for each $w \in A \setminus V(H)$. For each $b \in V(H)$, let $W_b := \{w \in A \setminus V(H) \mid b(w) = b\}$, and let $\sqsubseteq_b$ be an arbitrary linear order on W_b.

For any $u, v \in V(G)$, define $u \sqsubseteq v$ if one of the following holds:

- $u = v$;
- $u, v \in H$ and $u \leq v$;
- $u \in H$, $v \in G \setminus A$ and $u \leq v$;
- $u \in H$, $v \in A \setminus V(H)$ and $u \leq b(v)$;
- $u \in A \setminus V(H)$, $v \in G \setminus A$ and $b(u) \leq v$;
- $u, v \in A \setminus V(H)$, $b(u) = b(v)$ and $u \sqsubseteq_b v$.

It follows from the construction that $\sqsubseteq$ restricted to A is an elimination order of $G[A]$, and that $\sqsubseteq$ is an elimination order to degree d of G.

For each $b \in V(H)$, the set $\{v \in H \mid v \leq b\}$ has at most k elements, by the assumption on the height of the order $\leq$. Since G has maximum degree $k+d$ and $W_b \subseteq N(\{v \in H \mid v \leq b\})$, we have that W_b has at most $k(k+d)$ vertices. Since the height of any $\sqsubseteq$-chain is at most the height of a $\leq$-chain plus $|W_b|$, we conclude that the height of $\sqsubseteq$ is at most $k(k+d+1)$. □

The lemma above allows us to re-arrange the elimination order so that it includes all vertices of large degree. In contrast, the next lemma gives us a means to re-arrange the elimination order so that all vertices of small degree are made maximal in the order. This is again achieved while keeping the height of the elimination order bounded by a function of k and d. Note that while Lemma 6.3.1 only applies to graphs of maximal degree bounded by $k+d$, the following result applies to general graphs.

Lemma 6.3.2. *Let G be a graph. Let A be the set containing all vertices of degree greater than d, i.e. $A = \{v \in V(G) \mid \deg_G(v) > d\}$, and let $\leq$ be an elimination order to degree d of G of height k with non-maximal subgraph H, such that H contains A. Then, there is an elimination order to degree d of G of height at most $k((k+1)d)^{2^k}+1$ for which A is the set of non-maximal elements.*

Proof. Let G, H, A and $\leq$ be as in the statement of the lemma. We assume that G is connected – if not, we can apply the argument to each component of G. We construct an elimination order $\sqsubseteq$ of $G[A]$ from $\leq$, making sure that it has height at most $k((k+1)d)^{2^k}$, and satisfying the conditions of Proposition 5.2.4. This then extends to an elimination order to degree d of G by making all vertices not in A maximal, as in that Proposition.

Let $J := H \setminus A$. For $v \in V(J)$, let K_v be the set of vertices $w \in A$ such that:

1. $v \leq w$;
2. there is a path from v to w through $G \setminus A$; and
3. for any u with $u < v$, there is no path from u to w through $G \setminus A$.

Note that because $\leq$ is a tree order and due to the third condition, the sets K_v are pairwise disjoint. Let $\overline{K} := A \setminus (\bigcup_{v \in V(J)} K_v)$ be the set of vertices in A that are not contained in K_v for any v.

For each $v \in V(J)$, let $\sqsubseteq_v$ be an arbitrary linear order on K_v. The idea behind the construction below is that we replace v in the elimination order by K_v, ordered by $\sqsubseteq_v$. Formally, for any $u, w \in V(G)$, define $u \sqsubseteq w$ if one of the following holds:

- $u = w$;
- $u \in K_v$, $w \in G \setminus A$ and $v \leq w$;
- $u \in \overline{K}$, $w \in G \setminus A$ and $u \leq w$;
- $u, w \in K_v$ and $u \sqsubseteq_v w$;
- $u \in K_v$, $w \in K_{v'}$ and $v < v'$;
- $u \in \overline{K}$, $w \in K_v$ and $u \leq v$;
- $u \in K_v$, $w \in \overline{K}$ and $v \leq w$;

- $u, w \in \overline{K}$ and $u \leq w$.

We first show that $\sqsubseteq$ is an elimination order for $G[A]$. The construction ensures $\sqsubseteq$ is a tree order. Let $u, w \in A$. We show that if $u \leq w$, then either $u \sqsubseteq w$ or $w \sqsubseteq u$. We go through all possible cases: If $u = w$, we have $u \sqsubseteq w$. If there is some $v \in V(J)$ such that $u, w \in K_v$, then $u \sqsubseteq w$ or $w \sqsubseteq u$. Suppose $u \in K_v$, $w \in K_{v'}$ for $v, v' \in V(J)$. By definition of K_v, $v \leq u$ and similarly $v' \leq w$. Since we also have $u \leq w$, by the tree order property either $v \leq v'$ or $v' \leq v$ and thus $u \sqsubseteq w$ or $w \sqsubseteq u$. If $u \in \overline{K}$ and $w \in K_v$, then both $u, v \leq w$, so either $u \leq v$ or $v \leq u$, and thus either $u \sqsubseteq w$ or $w \sqsubseteq u$. The case where $u \in K_v$, $w \in \overline{K}$ is symmetric. Finally, if both $u, w \in \overline{K}$, then $u \sqsubseteq w$. Thus if $uw \in E(G)$, we have $u \leq w$ or $w \leq u$ and therefore $u \sqsubseteq w$ or $w \sqsubseteq u$. Hence $\sqsubseteq$ is an elimination order for $G[A]$.

Let Z be a component of $G \setminus A$. By assumption A is the set of all vertices of degree greater than d in G, and therefore Z has maximum degree d.

Suppose $u, v \in A$ are two vertices that are connected to Z, i.e. $E_G(u, V(Z)) \neq \emptyset \neq E_G(v, V(Z))$. We show that either $u \sqsubseteq v$ or $v \sqsubseteq u$. Note that there is a path P through $Z \subseteq G \setminus A$ from u to v, i.e. all vertices in P, except for the endpoints, lie outside of A. If P contains no vertices from J, then the connected component Z' of $G \setminus V(H)$ containing $P \setminus \{u, v\}$ satisfies $E_G(u, V(Z')) \neq \emptyset \neq E_G(v, V(Z'))$ and thus $u \leq v$ or $v \leq u$, and therefore by the above $u \sqsubseteq v$ or $v \sqsubseteq u$. Otherwise, P contains vertices from J. Let w be a $\leq$-minimal vertex in $V(P) \cap V(J)$. Then there is a path outside of A from w to u, and also to v (both part of P). Moreover, if neither $u \leq v$ nor $v \leq u$, then $w \leq u$ and $w \leq v$. Thus u and v are in K_w (or in $K_{w'}$ for some $w' < w$), and therefore $u \sqsubseteq v$ or $v \sqsubseteq u$.

It remains to show that the size of K_v is bounded by $k((k+1)d)^{2^k}$ for all $v \in V(J)$. Let G' be the graph obtained from G by adding an edge between two vertices $s, t \in V(J)$ whenever there is a path through $G \setminus V(H)$ between s and t. This increases the degree of vertices in $V(J)$ by at most kd, because each of these vertices is connected to at most d components of $G \setminus V(H)$ and each of these is connected to at most k vertices in H. Now there is a path between two vertices $s, t \in J$ in G outside of A if, and only if, there is a path between s and t in $G'[V(J)]$. Moreover, $\leq$ is also an elimination order for $G'[V(J)]$. So, as $G'[V(J)]$ has tree-depth at most k, by Lemma 4.1.1 it does not contain a path of length more than 2^k. Since each vertex on the path has degree at most $(k+1)d$, we can reach at most $((k+1)d)^{2^k}$ vertices in A on paths only containing vertices outside of A. Thus $|K_v| \leq ((k+1)d)^{2^k}$ and the height of $\sqsubseteq$ is bounded by $k|K_v| \leq k((k+1)d)^{2^k}$. □

Next we introduce the notion of d-degree torso and prove that it captures the properties that we require of an elimination order to degree d, that is, we show that an elimination order of the torso can be extended to an elimination order to degree d of the whole graph.

Definition 6.3.3. Let G be a graph, let $d > 0$ and let H be the subgraph of G induced by the vertices of degree larger than d. The *d-degree torso of G* is the graph obtained from H by adding an edge between two vertices $u, v \in H$ if there is a path through $G \setminus V(H)$ from u to v in G, i.e. a path where all vertices on the path, except for the endpoints, lie outside of H.

Lemma 6.3.4. *Let G be a graph and let C be the d-degree torso of G. Let $H = G[V(C)]$ and let $\leq$ be an elimination order for H. Then $\leq$ is an elimination order for C of height h if, and only if, $\leq$ can be extended to an elimination order to degree d for G of height $h + 1$.*

Proof. Let G, C, H and $\leq$ be as above.

Suppose $\leq$ is an elimination order for C. Since C is a supergraph of H, this means that $\leq$ is an elimination order for H. Let Z be a component of $G \setminus V(H)$. Since C contains all vertices of degree greater than d, Z has maximum degree d. If $E(Z, u) \neq \emptyset$ and $E(Z, v) \neq \emptyset$ for two vertices $u, v \in H$, then there is a path through $Z \subseteq G \setminus V(H)$ connecting u and v, so by the definition of the d-degree torso $uv \in E(C)$ and thus u, v are $\leq$-comparable. We can extend $\leq$ to a tree order $\leq'$ on $V(G)$ where all the vertices from $V(G) \setminus V(H)$ are maximal.

Conversely assume that $\leq$ can be extended to an elimination order to degree d for G. Let $uv \in E(C)$. If $uv \in E(H)$, then u and v must be $\leq$-comparable. Otherwise $uv \notin E(H)$, so there is a path through $G \setminus V(H)$ from u to v in G, i.e. both u and v are connected to a component Z of $G \setminus V(H)$ and thus comparable. Therefore $\leq$ is an elimination order for C. □

The next lemma establishes an upper bound on the tree-depth of the torso of a graph when the maximum degree is bounded.

Lemma 6.3.5. *Let G be a graph with elimination distance k to degree d and maximum degree $\Delta(G) \leq k + d$. Let C be the d-degree torso of G and let $\leq$ be a minimum height elimination order for C. Then $\leq$ has height at most $k(k+d+1)((k(k+d+1)+1)d)^{2^{k(k+d+1)}}$.*

Proof. Let $\sqsubseteq$ be a minimum height elimination order to degree d of G. Since G has elimination distance to degree d at most k, the height of $\sqsubseteq$ is at most k. Let H be the non-maximal subgraph of G under $\sqsubseteq$ and define

$$A = V(H) \cup \{v \in V(G) \mid \deg_G(v) > d\}.$$

By Lemma 6.3.1, the graph $G[A]$ has an elimination order $\preceq$ of height at most $k(k+d+1)$ that can be extended to an elimination order to degree d for G.

Let $A' = \{v \in A \mid \deg_G(v) > d\}$. By Lemma 6.3.2, the graph A' has an elimination order $\leq$ of height at most $k(k+d+1)((k(k+d+1)+1)d)^{2^{k(k+d+1)}}$ that can be extended to an elimination order to degree d for G.

Lastly note that $A' = V(C)$, so that by Lemma 6.3.4, $\leq$ is an elimination order for C. □

We are now ready to prove the main result of this section, which establishes that an elimination order of the d-degree torso can be extended to an elimination order to degree d of the whole graph, with height bounded by a function of only k and d.

Theorem 6.3.6. *Let G be a graph that has elimination distance k to degree d. Let $\leq$ be a minimum height elimination order of the d-degree torso G. Then $\leq$ can be extended to an elimination order to degree d of G of height at most*

$$k((k+1)(k+d))^{2^k} + k(1+k+d)(k(1+k+2d))^{2^{k(1+k+d)}} + 1.$$

Proof. We show that the d-degree torso of G has an elimination order of height at most $k((k+1)(k+d))^{2^k} + k(1+k+d)(k(1+k+2d))^{2^{k(1+k+d)}}$. The theorem then follows by Lemma 6.3.4.

Let C be the $(k+d)$-degree torso of G. We first show that the tree-depth of C is bounded by $k((k+1)(k+d))^{2^k}$. To see this, let $\sqsubseteq$ be an elimination order to degree d of

G of minimum height with non-maximal subgraph H. Note that H contains all vertices of degree greater than $k+d$, because vertices in $G \setminus V(H)$ are adjacent to at most k vertices in H.

Let $A = \{v \in V(H) \mid \deg_G(v) > k+d\}$. By Lemma 6.3.2, the graph $G[A]$ has an elimination order $\preceq$ of depth at most $h := k((k+1)(k+d))^{2^k}$ that can be extended to an elimination order to degree $k+d$ of G of height $h+1$. Note that $A = V(C)$, so by Lemma 6.3.4, the order $\preceq$ is an elimination order for C. Let $\preceq'$ denote its extension to G.

Let Z be a component of $G \setminus A$ and let C_Z be the d-degree torso of Z. By Lemma 6.3.5, there is an elimination order $\preceq_Z$ for C_Z of height at most $k(k+d+1)((k(k+d+1)+1)d)^{2^{k(k+d+1)}}$. Let v_Z be the $\preceq$-maximal element in C such that there is a $w \in C_Z$ with $v_Z \preceq' w$. Define

$$\leq' := \preceq \cup \bigcup_Z \preceq_Z \cup \bigcup_Z \{(v,w) \mid v \preceq' v_Z, w \in C_Z\}.$$

Observe that $C \cup \bigcup_Z C_Z$ has as a subgraph the d-degree torso of G. Thus $\leq'$ is an elimination order for the d-degree torso of G. The height of $\leq'$ is bounded by

$$td(C) + \max\{td(C_Z)\}_Z$$

which in turn is bounded by

$$k((k+1)(k+d))^{2^k} + k(1+k+d)(k(1+k+2d))^{2^{k(1+k+d)}}.$$

□

6.3.2 Graph canonisation parameterized by elimination distance to bounded degree

In this section we show that Graph Canonisation, and thus Graph Isomorphism, is FPT parameterized by elimination distance to bounded degree. The algorithm is based on Lindell's algorithm [115] that decides isomorphism of trees by establishing an ordering on them. We extend a version of this algorithm used by Bouland et al. [23] to decide isomorphism on graphs of bounded tree-depth.

Recall from Definition 2.3.16 that a canonical form F for a class of graphs $\mathcal{C}$ is an endomorphism of $\mathcal{C}$ that assigns a graph G to a graph isomorphic to G and that assigns two isomorphic graphs $G \cong H$ to the same graph, i.e. $G \cong F(G) = F(H)$. The Graph Canonisation for a class $\mathcal{C}$ is the problem of computing the canonical form of a given graph from $\mathcal{C}$. The Graph Canonisation problem for a class $\mathcal{C}$ is at least as hard as Graph Isomorphism on that class, because it is easy to see that we can use an algorithm that solves the Graph Canonisation problem to solve Graph Isomorphism. However, it is not known whether the reverse also holds.

The central idea is to define a total pre-order, i.e. a total, reflexive and transitive binary relation, $\sqsubseteq$ on graphs $(G, \leq)$ equipped with an elimination order to degree d, with the property that $(G, \leq) \sqsubseteq (G', \leq')$ and $(G', \leq) \sqsubseteq (G, \leq')$ if, and only if, $(G, \leq)$ is isomorphic to $(G', \leq')$. We then exhibit an algorithm that constructs, given a graph G, the elimination order to degree d on G that makes $(G, \leq)$ $\sqsubseteq$-minimal. This serves as a canonical representative of G. Before defining $\sqsubseteq$ formally, we need some additional terminology.

Definition 6.3.7. Let G be a graph. We denote the number of connected components of G by $\#G$.

The definition of $\sqsubseteq$ is by induction on the height of the order $\leq$. Inductively, we need to consider coloured graphs and so formally, $\sqsubseteq_k$ is a pre-order on triples $(G, \chi, \leq)$ where $\chi : V(G) \to X$ is a colouring of G and $\leq$ is an elimination order to degree d of G.

Definition 6.3.8. For a graph G, with a colouring $\chi : V(G) \to X$ and a vertex $v \in V(G)$, we write χ_v for the colouring of $G \setminus v$ given by $\chi_v : V(G \setminus v) \to X \times \{0, 1\}$ where $\chi_v(u) = (\chi(u), 1)$ if $(u, v) \in E(G)$ and $\chi_v(u) = (\chi(u), 0)$ if $(u, v) \notin E(G)$.

By Theorem 6.1.3 we can find a canonical form of each coloured graph Z of degree at most d in polynomial time. This allows us to define a pre-order (see Definition 2.4.1) $\sqsubseteq_0$ on the graphs by taking the lexicographical order of the string-representation of the canonical forms. That order is clearly reflexive and transitive, but since isomorphic graphs share the same canonical form, it is not antisymmetric – so it is a pre-order and not a partial order. In other words, that pre-order $\sqsubseteq_0$ is such that for every pair of Z, Z' of such graphs, either $Z \sqsubseteq_0 Z'$ or $Z' \sqsubseteq_0 Z$ and $Z \cong Z'$ if, and only if, both $Z \sqsubseteq_0 Z'$ and $Z' \sqsubseteq_0 Z$.

In the following definition we extend this to a pre-order $\sqsubseteq$ on coloured graphs with an elimination order to degree d by induction on the height of the order.

Definition 6.3.9. Let G, G' be graphs with colourings χ and χ' and respective elimination orders $\leq, \leq'$ to degree d. We say that $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$ if one of the following holds:

1. $\text{height}(\leq) < \text{height}(\leq')$;
2. $\text{height}(\leq) = \text{height}(\leq') = 0$ and $(G, \chi) \sqsubseteq_0 (G', \chi')$;
3. $\text{height}(\leq) = \text{height}(\leq') > 0$ and $\#G < \#G'$;
4. $\text{height}(\leq) = \text{height}(\leq') > 0$, $\#G = \#G' = 1$ and $\chi(v) < \chi'(v')$ where v, v' are minimal elements of G, G' with respect to $\leq, \leq'$ respectively;
5. $\text{height}(\leq) = \text{height}(\leq') > 0$, $\#G = \#G' = 1$ and
$$(G \setminus v, \chi_v, \leq) \sqsubseteq (G' \setminus v', \chi'_{v'}, \leq'),$$
where v, v' are minimal elements of G, G' with respect to $\leq, \leq'$ respectively;
6. $\text{height}(\leq) = \text{height}(\leq') > 0$, $s = \#G = \#G' > 1$, and there is an enumeration $G_1, \ldots, G_s$ of the components of G and $G'_1, \ldots, G'_s$ of the components of G' such that
   (a) whenever $i \leq j$, $(G_i, \chi, \leq) \sqsubseteq (G_j, \chi, \leq)$ and $(G'_i, \chi', \leq') \sqsubseteq (G'_j, \chi', \leq')$; and
   (b) if for any i, $(G_i, \chi, \leq) \not\sqsubseteq (G'_i, \chi', \leq')$, then there is a $j < i$ such that $(G'_j, \chi', \leq') \not\sqsubseteq (G_j, \chi, \leq)$.

To see that this inductive definition is well-founded, note that the recursive use of $\sqsubseteq$ is on graphs either equipped with an elimination order to degree d of strictly smaller height (clause (5)) or with strictly fewer components (clause (6)). The next two propositions are aimed at showing that this definition establishes a linear pre-order on coloured graphs with an elimination order that classifies them up to isomorphism.

Proposition 6.3.10. *For any pair $(G, \chi, \leq)$, $(G', \chi', \leq')$ of coloured graphs with an elimination order to degree d, at least one of $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$ or $(G', \chi', \leq') \sqsubseteq (G, \chi, \leq)$ holds.*

Proof. The proof is by induction on the height k of $\leq$ and for a fixed k, by induction on the number of components. If $k = 0$, then either height$(\leq') > 0$ and $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$ by clause (1) of the definition, or height$(\leq') = 0$ and at least one of $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$ or $(G', \chi', \leq') \sqsubseteq (G, \chi, \leq)$ must hold by clause (2) and the definition of $\sqsubseteq_0$.

Now suppose $k > 0$. If one of the first four conditions of Definition 6.3.9 is satisfied, we are done. So suppose they fail and suppose $\#G = \#G' = 1$. Then, if v and v' are the minimal elements of G, G' with respect to $\leq, \leq'$, the height of these orders on $G \setminus v$ and $G' \setminus v'$ is strictly less than k. By induction hypothesis, either $(G \setminus v, \chi_v, \leq) \sqsubseteq (G' \setminus v', \chi'_{v'}, \leq')$ or $(G' \setminus v', \chi_{v'}, \leq') \sqsubseteq (G \setminus v, \chi_v, \leq)$, so the claim follows.

Finally, suppose height$(\leq)$ = height$(\leq') > 0$ and $s = \#G = \#G' > 1$ and we argue that in all such cases, clause (6) ensures the result. By the induction hypothesis and the previous case, for any pair of components G_i and G_j of G, one of $(G_i, \chi, \leq) \sqsubseteq (G_j, \chi, \leq)$ or $(G_j, \chi, \leq) \sqsubseteq (G_i, \chi, \leq)$ must hold and similarly for the components of G'. Thus, the components of each can be ordered so that whenever $i \leq j$, $(G_i, \chi, \leq) \sqsubseteq (G_j, \chi, \leq)$ and $(G'_i, \chi', \leq') \sqsubseteq (G'_j, \chi', \leq')$. Also, by the induction hypothesis, we have for each i, either $(G_i, \chi, \leq) \sqsubseteq (G'_i, \chi', \leq')$ or $(G'_i, \chi', \leq') \sqsubseteq (G_i, \chi, \leq)$. If the first case holds for all i, then $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$. If the second case holds for all i, then $(G', \chi', \leq') \sqsubseteq (G, \chi, \leq)$. If neither holds for all i, consider the least i such that either $(G_i, \chi, \leq) \not\sqsubseteq (G'_i, \chi', \leq')$ or $(G'_i, \chi', \leq') \not\sqsubseteq (G_i, \chi, \leq)$. In the former case, we have $(G', \chi', \leq') \sqsubseteq (G, \chi, \leq)$ and in the latter $(G_i, \chi, \leq) \sqsubseteq (G'_i, \chi', \leq')$. In all cases, the proposition is established. □

This proposition implies that it is always possible to enumerate the components $G_1, \ldots, G_s$ of G in $\sqsubseteq$-order. The next proposition shows that this order is, essentially, unique.

Proposition 6.3.11. *Let G, G' be graphs with colourings χ and χ' and let $\leq, \leq'$ be elimination orders to degree d on G, G' respectively.*

Then both $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$ and $(G', \chi', \leq') \sqsubseteq (G, \chi, \leq)$ if, and only if, $(G, \chi, \leq) \cong (G', \chi, \leq')$.

Proof. Suppose $(G, \chi, \leq) \cong (G', \chi, \leq')$ and suppose, without loss of generality that $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$. We aim to show that also $(G', \chi', \leq') \sqsubseteq (G, \chi, \leq)$, and we do this by induction on the height k of $\leq$ and for a fixed k, by cases on the number of components. If $k = 0$ then since $(G, \chi) \cong (G', \chi')$, the conclusion follows from the definition of $\sqsubseteq_0$ and clause (2) of Definition 6.3.9. If $k > 0$ and $\#G = 1$ then the induction hypothesis and the fact that $(G \setminus v, \chi_v, \leq) \cong (G' \setminus v', \chi'_{v'}, \leq')$ together imply that $(G \setminus v, \chi_v, \leq) \sqsubseteq (G' \setminus v', \chi'_{v'}, \leq')$ and $(G' \setminus v', \chi'_{v'}, \leq') \sqsubseteq (G \setminus v, \chi_v, \leq)$, so the proposition holds by clause (4) of Definition 6.3.9. Finally, suppose that $k > 0$ and $\#G > 1$ and let $G_1, \ldots, G_s$ enumerate the components of G in $\sqsubseteq$ order. By the isomorphism between $(G, \chi, \leq)$ and $(G', \chi', \leq')$ and the induction hypothesis, it follows that for the corresponding enumeration $G'_1, \ldots, G'_s$ of components of G' in $\sqsubseteq$ order, we have that $(G_i, \chi, \leq) \cong (G'_i, \chi, \leq')$ for each i. Then by clause (6) of Definition 6.3.9, both $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$ and $(G', \chi', \leq') \sqsubseteq (G, \chi, \leq)$ hold.

For the other direction, suppose $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$ and $(G', \chi', \leq') \sqsubseteq (G, \chi, \leq)$ If height$(\leq)$ < height$(\leq')$, then $(G', \chi', \leq') \not\sqsubseteq (G, \chi, \leq)$, so assume that height$(\leq)$ = height$(\leq') = k$ and we proceed by induction on k. If $k = 0$, we have by clause (2)

that $(G,\chi) \sqsubseteq_0 (G',\chi')$ and $(G',\chi') \sqsubseteq_0 (G,\chi)$ which, by definition of $\sqsubseteq_0$ means that $(G,\chi) \cong (G',\chi)$, and the two orders are both trivial.

Suppose then that $k > 0$ and assume first that G and G' are both connected. Then both G, G' have minimal elements v, v' with respect to $\leq, \leq'$. By the induction hypothesis we have $(G \setminus v, \chi_v \leq) \cong (G' \setminus v', \chi_{v'}, \leq')$, so let $\sigma : V(G \setminus v) \to V(G' \setminus v')$ be an isomorphism witnessing this. Extend this to $\hat{\sigma} : V(G) \to V(G')$ by setting $\hat{\sigma}(v) = v'$ and we claim that this is an isomorphism between $(G, \chi, \leq)$ and $(G', \chi', \leq')$. It is clear that $\hat{\sigma}$ takes the order $\leq$ to $\leq'$ since v and v' are minimal for these respective orders. Also, $\hat{\sigma}$ preserves colours since the colourings χ_v and $\chi_{v'}$ refine the colourings χ and χ' respectively and by clause (4) we have $\chi(v) = \chi'(v')$. Finally, we need to verify that $\hat{\sigma}$ preserves the edge relation. Since σ preserves all edges, we only need to verify that $uv \in E(G)$ if, and only if, $\hat{\sigma}(u)v' \in E(G')$. But, this follows from the fact that $\chi_v(u) = \chi'_{v'}(\sigma(u))$.

Now, if G and G' have different number of connected components, then we cannot have both $(G, \chi, \leq) \sqsubseteq (G', \chi', \leq')$ and $(G', \chi', \leq') \sqsubseteq (G, \chi, \leq)$. So, we are left with the case where $\#G = \#G' = s > 1$. Let $G_1 \sqsubseteq \cdots \sqsubseteq G_s$ be the connected components of G and let $G'_1 \sqsubseteq \cdots \sqsubseteq G'_s$ be the connected components of G'. Then by definition of $\sqsubseteq$ we have that both $(G_i, \chi, \leq) \sqsubseteq (G'_i, \chi', \leq')$ and $(G'_i, \chi', \leq') \sqsubseteq (G_i, \chi, \leq)$ for each $i \leq s$. By the argument for connected graphs above, we have $(G_i, \chi, \leq) \cong (G'_i, \chi', \leq')$ for each i and we are done. □

Algorithm 1 Recursive construction of minimal elimination order to degree d.

Input: A graph G, a subgraph $C \subseteq G$, and $\ell, d \in \mathbb{N}$.

Output: A $\sqsubseteq$-minimal elimination order to degree d of G, where all the sub-maximal vertices are from C.

```
1: procedure FindMinimalOrder(G, C, ℓ, d)
2:     if #G > 1 then
3:         return ⋃_i FindMinimalOrder(G_i, C[G_i ∩ C], ℓ, d), where the G_i are the connected components of G.
4:     else if Δ(G) ≤ d then
5:         return ∅.
6:     else
7:         find the set of potential roots R of a minimal elimination order on C.
8:         for w ∈ R do
9:             colour(u) := colour(u) ∪ {ℓ} for all neighbours u of w
10:            <_w := FindMinimalOrder(G \ w, C \ w, ℓ + 1, d)
11:            <_w := <_w ∪ {(w, u) | u ∈ V(G) \ {w}}.
12:        return <_w that makes (G, <_w) ⊑-minimal.
```

In order to define an algorithm for constructing a minimal elimination order to degree d, we need one further ingredient. For a graph G of tree-depth k, we say that a vertex v is a *potential root* of a minimal elimination order on G if $td(G \setminus v) < k$. We know from Bouland et al. [23] that there is a computable function f such that the number of potential roots in any graph of tree-depth k is at most $f(k)$. Moreover, the set of all such vertices can be found by an algorithm running in time $f(k)n^3$ where n is the number of vertices in G.

Recall from Definition 2.4.8 that the level of an element v in a tree order $\leq$ is the length of the chain $\{w \in V(G) \mid w \leq v\}$.

This now allows us to define Algorithm 1 which computes a minimal elimination order to degree d. The algorithm is started with the graph G, its d-degree torso C, the level ℓ initially set to 0 and the degree d. It then recursively chooses a potential root and returns the order that is minimal with respect to $\sqsubseteq$.

With this, we are able to establish our main result.

Theorem 6.3.12. Graph Canonisation *is* FPT *parameterized by elimination distance to degree d, for any d.*

Proof. Suppose we are given a graph G with $|V(G)| = n$. We first compute the d-degree torso C of G in $O(n^4)$ time. Using Algorithm 1 we compute a $\sqsubseteq$-minimal elimination order $<$ to degree d on G. By the analysis from Bouland et al. [23, Theorem 11], this takes time $O(h(k)n^3log(n))$ for some computable function h. Note for the base case that we can compute the canonical representation of a component Z with $\Delta(Z) \leq d$ by Theorem 6.1.3 in polynomial time (where the degree of the polynomial depends on d).

Note that the order $\sqsubseteq$ also induces a natural order on the vertices of G and we can thus enumerate the vertices of G in a canonical way in linear time. □

Since we can use the Graph Canonisation problem to solve Graph Isomorphism, this implies the following:

Corollary 6.3.13. Graph Isomorphism *is* FPT *parameterized by elimination distance to degree d, for any d.*

CHAPTER 7

Graph classes characterised by excluded minors

This chapter establishes that the problem of determining elimination distance to any minor-closed class $\mathcal{C}$ is FPT, and provides an explicit algorithm for the problem if we are given a set of forbidden minors characterising $\mathcal{C}$.

A minor of a graph is obtained by deleting vertices and edges, and by contracting edges. A graph class is minor-closed if whenever some graph is a minor of a graph in the class, that graph must also be contained in the class. (The definitions can be found in Section 2.3.2.)

For example the class of planar graphs, i.e. graphs that can be embedded into the plane without edges crossing, is clearly minor-closed. Wagner [155] showed that the class of planar graphs is exactly the class characterized by not having the complete graph K_5 and the complete bipartite graph $K_{3,3}$ as minors. A more general statement, for a long time known as Wagner's Conjecture, was that every minor-closed class is characterized by such a finite set of 'forbidden minors'.

In a series of twenty papers spanning over 500 pages from 1983 to 2004 Robertson and Seymour developed their theory of graph minors and concluded with a proof of Wagner's conjecture, which is now called the Robertson-Seymour Theorem (cf. Theorem 2.3.21). The theorem states that the class of all undirected graphs, partially ordered by the graph minor relationship, is in fact well-quasi-ordered. This is equivalent to the statement that every minor-closed class of graphs is characterised by a finite set of 'forbidden minors'.

Further examples of minor-closed classes include the class of all trees, the class of all forests, and for every value of k, the class of graphs with genus bounded by k (i.e. a generalisation of planar graphs), the class of graphs with tree-depth bounded by k, the class of graphs with path-width (or tree-width) bounded by k. For some of these the explicit characterisation is known.

The Robertson-Seymour Theorem also has a very interesting consequence for the study of algorithmics. Robertson and Seymour [142] show that there is a polynomial-time algorithm[1] that checks, given a graph G, whether a fixed graph H is a minor of G. This implies that for any fixed minor-closed class $\mathcal{C}$ of graphs, there is a polynomial time

[1]The runtime Robertson and Seymour originally showed was cubic, but it was recently improved to quadratic by Kawarabayashi et al. [104]. Also note that when checking whether $H \preceq G$, the runtime is polynomial in the size of the graph G and has exponential dependence on the size of H.

algorithm to test membership: given a graph G, just iterate over all the 'forbidden minors' and check if one of them is a minor of G. If one of them is, $G \notin \mathcal{C}$, otherwise $G \in \mathcal{C}$. However, there is a huge constant cost in the runtime of the algorithm which makes it not useful in practice.

In the area of parameterized complexity the technique based on the Robertson-Seymour algorithm is often used to show that a problem is FPT, usually motivating further research into the problem revealing more efficient algorithms. The technique can be applied for a graph parameter k if the graphs with parameter $\leq k$ are minor-closed. For example, since both the classes of tree-depth $\leq k$ and tree-width $\leq k$ are minor-closed, testing whether tree-depth or tree-width is at most k is FPT.

The catch is that this technique is non-constructive: if we do not know what the 'forbidden minors' are, we know that there is a polynomial-time algorithm, but we cannot use it. Although many graph classes are known to be minor-closed, the set of 'forbidden minors' characterising them is usually not known. Since knowing the 'forbidden minors' is a prerequisite for applying the algorithm from the Robertson-Seymour Theorem, this makes the search for 'forbidden minors' an important research question.

In this chapter we show that for certain graph classes the 'forbidden minors' can be computed. Recall the parameter *elimination distance to a class* $\mathcal{C}$ that we introduced in Chapter 5. It is a distance parameter based on elimination trees or tree-depth decompositions that is a lower bound for both tree-depth (cf. Section 5.4) and deletion distance (cf. Section 3.1).

Here we show that if a class of graphs $\mathcal{C}$ is minor-closed and its set of 'forbidden minors' is known, then for every $k > 0$, we can also compute the set of 'forbidden minors' that characterise the class of graphs with elimination distance k to $\mathcal{C}$. By showing that we can compute the 'excluded minors' in this particular case, we establish that there is an explicit algorithm that can check whether a graph has elimination distance k to $\mathcal{C}$ if the 'excluded minors' for $\mathcal{C}$ are available to the algorithm.

Our result also implies that checking whether a given graph G has elimination distance k to a minor-closed class $\mathcal{C}$ is fixed-parameter tractable, answering a question posed in [28].

A large number of graph classes can be characterised by excluded minors, and an even larger number can be characterised by the elimination distance to such graph classes. Thus our result here gives a general method for a large number of problems and is thus an *algorithmic meta-theorem*.

In Chapter 8 we establish another algorithmic meta-theorem on the tractability of graph problems that are both slicewise first-order definable and slicewise nowhere dense. Note that our results discussed in Chapter 8 do not apply here: While a proper minor-closed class is always nowhere dense, it is not generally first-order definable (for instance, neither the class of acyclic graphs nor the class of planar graphs is), and elimination distance to such a class is also not known to be first-order definable. Thus, our results from Chapter 8 do not apply here.

7.1 Elimination distance to excluded minors

In this section we show that for any $k > 0$, the class of graphs with elimination distance k to a minor-closed class $\mathcal{C}$ is also minor-closed, and, moreover, if we have access to the list of 'forbidden minors' characterising $\mathcal{C}$, we can find the list of 'forbidden minors'

characterising the class of graphs with elimination distance k to $\mathcal{C}$. We first pin down the exact computational problem that we are trying to solve and give an outline of the proof strategy.

Recall the notion of elimination distance (Definition 5.1.1) from Chapter 5. For convenience we restate the definition here.

Definition 7.1.1 (Definition 5.1.1)**.** Let $\mathcal{C}$ be a class of graphs closed under taking subgraphs. The *elimination distance to* $\mathcal{C}$ of a graph G is defined as follows:

$$ed_{\mathcal{C}}(G) := \begin{cases} 0, & \text{if } G \in \mathcal{C}; \\ 1 + \min\{ed_{\mathcal{C}}(G \setminus v) \mid v \in V(G)\}, & \text{if } G \notin \mathcal{C} \text{ and } G \text{ is connected;} \\ \max\{ed_{\mathcal{C}}(H) \mid H \text{ a connected component of } G\}, & \text{otherwise.} \end{cases}$$

We build on work of Adler et al. [6] to show that from a finite list of the 'forbidden minors' characterising $\mathcal{C}$, we can compute the set of 'forbidden minors' characterising the graphs with elimination distance k to $\mathcal{C}$. Adler et al. show how to do this for apex graphs, from which one immediately obtains the result for graphs that are k deletions away from $\mathcal{C}$. To extend this to elimination distance k, we show how we can construct the forbidden minors for the closure of a minor-closed class under disjoint unions.

In the following we show that determining the elimination distance of a graph to a minor-closed class $\mathcal{C}$ is FPT when parameterized by the elimination distance. More generally, we formulate the following parameterized problem where the forbidden minors of $\mathcal{C}$ are also part of the parameter.

Recall from Definition 2.3.20 that we write $M(\mathcal{C})$ for the set of minimal excluded minors of $\mathcal{C}$, i.e. the set of graphs in the complement of $\mathcal{C}$ such that for each $G \in M(\mathcal{C})$ all proper minors of G are in $\mathcal{C}$.

Elimination Distance to Excluded Minors
Input: A graph G, a natural number $k \in \mathbb{N}$ and a finite set of graphs M
Parameter: $k + \sum_{H \in M} \lvert H \rvert$
Problem: Does G have elimination distance k to the class $\mathcal{C}$ characterised by $M(\mathcal{C}) = M$?

In Section 7.2 we apply the characterisation of elimination distance k to a class $\mathcal{C}$ established in Section 5.2.3. We first use this characterisation to establish that for any minor-closed class $\mathcal{C}$, the class of graphs which have elimination distance k to $\mathcal{C}$ is also a minor-closed class. We then show how to compute a set of excluded minors for the class of graphs which have elimination distance k to $\mathcal{C}$ from a set of excluded minors for $\mathcal{C}$.

7.2 Computing the excluded minors of $\mathcal{C}^{\text{apex}}$ and $\overline{\mathcal{C}}$

In this section we first show that for any minor-closed class $\mathcal{C}$ the class of graphs which have elimination distance k to $\mathcal{C}$ is also a minor-closed class. Then we give a procedure to compute a set of excluded minors for the class of graphs which have elimination distance k to $\mathcal{C}$ from a set of excluded minors for $\mathcal{C}$.

Recall the alternative characterisation of elimination distance k to a class $\mathcal{C}$ in terms of apex graphs and closures under disjoint unions that we established in Section 5.2.3.

First we show that if $\mathcal{C}$ is a minor-closed class of graphs then so is the class of graphs $\mathcal{C}_k$ with elimination distance k to $\mathcal{C}$ for any $k \geq 0$.

Proposition 7.2.1. *Let $\mathcal{C}$ be a minor-closed class. Then $\mathcal{C}_k$ is also minor-closed.*

Proof. Indeed, it is well-known that $\mathcal{C}^{\text{apex}}$ is minor-closed for any minor-closed $\mathcal{C}$: contracting any edge, or removing any edge or vertex, leads to another graph in $\mathcal{C}^{\text{apex}}$. For, if $G \in \mathcal{C}^{\text{apex}}$ with apex $v \in V(G)$, then any graph obtained from $G \setminus \{v\}$ by an edge contraction or removal of a vertex is contained in $\mathcal{C}$, and thus any graph obtained from G by a contraction of an edge not incident to v, or a removal of a vertex other than v, gives us a graph in $\mathcal{C}^{\text{apex}}$. It is easy to see that the same holds for any edge removal or contraction of an edge incident to v. And if v itself is removed, any other vertex may be chosen as the apex.

So it is just left to show that $\overline{\mathcal{C}}$ is also minor-closed. But it is clear that if H is a minor of a graph G that is the disjoint union of graphs $G_1, \ldots, G_s$, then H itself is the disjoint union of (possibly empty) minors of $G_1, \ldots, G_s$. Thus, the class of graphs of elimination distance at most k to a minor-closed class $\mathcal{C}$ is itself minor-closed. □

We next show how to construct a set of excluded minors of $\mathcal{C}_k$ from the corresponding set for $\mathcal{C}$.

To obtain $M(\mathcal{C}_k)$, we need to iteratively compute $M(\mathcal{C}^{\text{apex}})$ and $M(\overline{\mathcal{C}})$ from $M(\mathcal{C})$. Adler et al. [6] show that from the set of minimal excluded minors $M(\mathcal{C})$ of a class $\mathcal{C}$, we can compute $M(\mathcal{C}^{\text{apex}})$ (and thus, in fact, the minimal set of excluded minors for the class of graphs with deletion distance k to $\mathcal{C}$ for any fixed value k).

Theorem 7.2.2 ([6], Theorem 5.1)**.** *There is a computable function that takes the set of graphs $M(\mathcal{C})$ characterising a minor-closed class $\mathcal{C}$ to the set $M(\mathcal{C}^{apex})$.*

We next aim to show that from $M(\mathcal{C})$ we can also compute $M(\overline{\mathcal{C}})$. Together with Theorem 7.2.2 this implies that from $M(\mathcal{C})$ we can compute $M(\mathcal{C}_k)$, the set of minimal excluded minors for the class of graphs with elimination distance k to $\mathcal{C}$.

We begin by characterising minor-closed classes that are closed under disjoint unions in terms of the connectedness of their excluded minors.

Lemma 7.2.3. *Let $\mathcal{C}$ be a class of graphs closed under taking minors. Then $\mathcal{C}$ is closed under taking disjoint unions if, and only if, each graph in $M(\mathcal{C})$ is connected.*

Proof. Let $\mathcal{C}$ be a minor-closed class of graphs, and let $M(\mathcal{C}) = \{H_1, \ldots, H_s\}$ be its set of minimal excluded minors.

Suppose each of the graphs in $M(\mathcal{C})$ is connected. Let $H \in M(\mathcal{C})$ and let $G = G_1 \oplus \cdots \oplus G_r$ be the disjoint union of graphs $G_1, \ldots, G_r \in \mathcal{C}$. Because H is connected, we have that $H \preceq G$ if, and only if, $H \preceq G_i$ for one $1 \leq i \leq r$. So, since all the $G_i \in \mathcal{C}$, we have $H \not\preceq G$ and thus $G \in \mathcal{C}$. This shows that $\mathcal{C}$ is closed under taking disjoint unions.

Conversely assume one of the graphs $H \in M(\mathcal{C})$ is not connected and let $A_1, \ldots, A_t$ be its components. Then $A_1, \ldots, A_t \in \mathcal{C}$, since each A_i is a proper minor of H, and H is minor-minimal in the complement of $\mathcal{C}$. However, $A_1 \oplus \cdots \oplus A_t = H \notin \mathcal{C}$. □

An important tool is the *connection closure* of a graph G, which defines an edge-minimal connected supergraph of G.

Definition 7.2.4. For a graph G with connected components $G_1, \ldots, G_r$, let $\mathcal{H}$ denote the set of connected graphs H with $V(H) = V(G)$ such that the subgraph of H induced by $V(G_i)$ is exactly G_i. We define the *connection closure* of G to be the set of all minimal (under the subgraph relation) graphs in $\mathcal{H}$. The connection closure of a set S of graphs is the union of the connection closures of the graphs in S.

Next we observe each of the graphs in the connection closure of a graph has the same number of edges.

Lemma 7.2.5. *Let G be a graph with e edges and m connected components. Then any graph in the connection closure of G has exactly $e + m - 1$ edges.*

Proof. Let H be a graph in the connection closure of G. Then H has G as a subgraph, so e of its edges come from G. In addition, H is connected, so it has at least $m-1$ additional edges corresponding to a tree on m vertices connecting the m components. Because H is minimal (under the subgraph relation), it does not have more than $m - 1$ additional edges. Thus H has exactly $e + m - 1$ edges. □

Lemma 7.2.6. *Let $\mathcal{C}$ be a minor-closed class of graphs. Then $M(\overline{\mathcal{C}})$ is the set of minor-minimal graphs in the connection closure of $M(\mathcal{C})$.*

Proof. Let $\mathcal{C}$ be a minor-closed class of graphs, with $M(\mathcal{C})$ its set of minimal excluded minors, and let $\hat{M}$ be the connection closure of $M(\mathcal{C})$.

Let G be a graph such that $\hat{H} \not\preceq G$ for all $\hat{H} \in \hat{M}$. Suppose for a contradiction that G is not a disjoint union of graphs from $\mathcal{C}$. Then there is a component G' of G that is not in $\mathcal{C}$ and therefore there is a graph $H \in M(\mathcal{C})$ such that $H \preceq G'$. We show that one of the graphs in the connection closure of H is a minor of G'.

Let $\{w_1, \ldots, w_s\}$ be the vertex set of H and consider the image $T_1, \ldots, T_s$ of the minor map from H to G'. Let T be a minimal subtree of G' that contains all of the T_i. Such a tree must exist since G' is connected. Let $\hat{H}$ be the graph with the same vertex set as H, and an edge between two vertices w_i, w_j whenever either $w_i w_j \in E(H)$ or when there is a path between T_{w_i} and T_{w_j} in T that is disjoint from any T_{w_k} with $w_i \neq w_k \neq w_j$. We claim that $\hat{H}$ is in the connection closure of H. By construction, $\hat{H}$ is connected and contains all components of H as disjoint subgraphs, so we only need to argue minimality. $\hat{H}$ has no vertices besides those in H so no graph obtained by deleting a vertex would contain all components of H as subgraphs. To see that no edge of $\hat{H}$ is superfluous, we know from Lemma 7.2.5 it has exactly $e+m-1$ edges and thus no proper subgraph could be connected and have all components of H as disjoint subgraphs. By the construction $\hat{H} \preceq G' \preceq G$, so by the transitivity of the minor relation we have that $\hat{H} \preceq G$.

Conversely let G be an arbitrary graph and assume that $\hat{H} \in \hat{M}$ and $\hat{H} \preceq G$. Because $\hat{H}$ is connected, there is a connected component G' of G such that $\hat{H} \preceq G'$. Now there must be a graph $H \in M(\mathcal{C})$ such that $\hat{H}$ is in the connection closure of H, and since H is a subgraph of $\hat{H}$, $H \preceq \hat{H}$. Then, by the transitivity of the minor relation, $H \preceq G'$ and thus $G' \notin \mathcal{C}$. Therefore G is not a disjoint union of graphs from $\mathcal{C}$. □

7.3 Computing the excluded minors of $\mathcal{C}_k$

Now we can put everything together and prove our main theorem:

Theorem 7.3.1. *There is a computable function which takes a set M of excluded minors characterising a minor-closed class $\mathcal{C}$ and $k \geq 0$ to the set $M(\mathcal{C}_k)$.*

Proof. The proof is by induction. For $k = 0$, the set of minimal excluded minors of $\mathcal{C}_0$ is $M(\mathcal{C}_0) = M(\mathcal{C})$, which is given. For $k > 0$, we have that $\mathcal{C}_k = \overline{\mathcal{C}_{k-1}{}^{\text{apex}}}$. By the induction hypothesis we can compute $M(C_{k-1})$, by Theorem 7.2.2 we can compute $M(\mathcal{C}_{k-1}{}^{\text{apex}})$ and using Lemma 7.2.6 we can compute the connection closure of $M(\mathcal{C}_{k-1}{}^{\text{apex}})$ to obtain $M(\overline{\mathcal{C}_{k-1}{}^{\text{apex}}}) = M(\mathcal{C}_k)$. □

So by the Robertson-Seymour Theorem (cf. Theorem 2.3.21) we have the following:

Corollary 7.3.2. *The problem* Elimination Distance to Excluded Minors *is fixed-parameter tractable.*

CHAPTER 8

Slicewise first-order and slicewise nowhere-dense classes

In this chapter we establish a general technique, usually called an algorithmic meta-theorem, that allows us to show that for various classes $\mathcal{C}$ of sparse graphs, and several measures of distance to such classes, the problem of determining the distance of a given graph G to $\mathcal{C}$ is fixed-parameter tractable. More precisely, we establish that any problem that is both *slicewise nowhere dense* and *slicewise first-order definable* (these terms are defined precisely below in Definition 8.2.1 and Definition 8.2.2) is FPT. This includes problems such as deletion distances, and more general edit distances to several sparse graph classes.

In particular, this technique implies that if we have a class $\mathcal{C}$ that is first-order definable and nowhere dense and the distance measure we are interested in is also first-order definable (that is to say, for each k there is a formula that defines the graphs at distance k from $\mathcal{C}$), then the problem of determining the distance is FPT. More generally, if we have a parameterized problem (Q, κ) that is slicewise nowhere dense and slicewise first-order definable, and a measure of distance to it is definable in the sense that for any values of k and d, there is a first-order formula defining the graphs at distance d to the class $\{G \mid G \in Q \text{ and } \kappa(G) \leq k\}$, then the problem of deciding whether a graph has distance at most d to this class is FPT parameterized by $k + d$.

As a consequence of this it follows that if $\mathcal{C}$ is a nowhere dense class of graphs that is definable by a first-order formula, then the parameterized problem of determining the distance of a graph G to $\mathcal{C}$ is FPT, for various notions of distance that can be themselves so defined.

As an application, we give new proofs of a number of existing results in the literature, which we explore in Section 8.3. We show that certain natural distance measures to sparse graph classes are FPT. In particular, we get that various forms of edit distance to classes of bounded-degree graphs are FPT (a result established by Moser and Thilikos [129] for deletion distance, by Mathieson [125] for several editing distances, and by Golovach [88] by more direct methods).

A simple example is deletion distance to graph classes of bounded degree. Consider, for each $d > 0$, the class of graphs with maximum degree at most d. These classes are both first-order definable and nowhere dense. So our result implies that the problem of determining whether a graph has degree at most d is FPT parameterized by d – however,

it is easy to see that this problem is in fact in P. Here we are concerned with distance parameters and we show in Section 8.3 that several natural distance parameters, such as deletion distance are first-order definable. Moreover, we will see that the problem of determining whether a graph has a certain deletion distance to a graph class with bounded maximal degree is slicewise nowhere dense. This means that the problem 'Can we remove k vertices to obtain a graph with maximum degree less than d?' is fixed-parameter tractable.

Another interesting application is obtained by considering elimination distance of a graph G to the class $\mathcal{C}$ of edgeless graphs. Recall from Section 5.4 that this is nothing other than the *tree-depth* of G. While it is an open question whether elimination distance to a class $\mathcal{C}$ is (or is not) in general first-order definable, it is in the particular case where $\mathcal{C}$ is the class of empty graphs. Thus, we obtain as an application of our method the result that tree-depth is FPT, a result previously known from other algorithmic meta theorems (see [133, Theorem 17.2]).

The method of establishing that a parameterized problem is FPT by establishing that it is slicewise nowhere dense and slicewise first-order definable appears to be a powerful method of some generality which will find application beyond these examples.

Our work builds on a recent result of Grohe et al. [92] which shows that the problem of evaluating first-order formulas on any *nowhere dense* class of graphs is FPT with the size of the formula as parameter. We extract from their proof of this result a general statement about the fixed-parameter tractability of definable sparse graph classes. Our method is an adaptation of the main algorithm in [92]. Since the proof is essentially a modification of their central construction we state the main results they prove and explain how the proofs can be adapted for our purposes.

Section 8.1 gives an overview of the key elements of the construction from [92] and the elements from it which we need to extract for our result. Section 8.2 then gives our main result and Section 8.3 derives some consequences for distance measures.

8.1 Evaluating formulas on nowhere dense classes

The key result of [92] is:

Theorem 8.1.1 (Grohe *et al.* [92], Theorem 1.1)**.** *For every nowhere dense class $\mathcal{C}$ and every $\epsilon > 0$, every property of graphs definable in first-order logic can be decided in time $O(n^{1+\epsilon})$ on $\mathcal{C}$.*

We first give a sketch of the algorithm from Theorem 8.1.1 with an emphasis on the changes needed for our purposes. We refer to [92] for several definitions and results.

The algorithm developed in the proof of Theorem 8.1.1 uses a locality-based approach, similar to that used by Frick and Grohe [75] to show that first-order evaluation is FPT on graphs of local bounded tree-width and developed in [50] for application to graph classes with locally excluded minors. The idea is that any first-order formula φ is, by Gaifman's theorem, equivalent to a Boolean combination of *local* formulae, that is formulae that assert the existence of neighbourhoods satisfying certain conditions. In classes of sparse graphs where the size (or other parameter) of neighbourhoods of a given radius can be bounded, this yields an efficient evaluation algorithm.

In nowhere dense classes of graphs, we cannot in general bound the size of neighbourhoods. For example, the class of apex graphs is nowhere dense, but a graph may contain

a vertex whose neighbourhood is the whole graph. However, nowhere dense classes are *quasi-wide*, which means that we can remove a small (i.e. parameter-dependent) set of vertices (the bottleneck) to ensure that there are many vertices that are far away from each other. This is a notion of sparseness for graphs introduced in [50] (cf. Definition 2.3.25 and Definition 2.3.26; for the connection to nowhere-dense classes, see [133]). Grohe *et al.* [92] use this approach to iteratively transform the input graph into a coloured graph where key bottleneck vertices are removed and vertices are coloured to keep relevant information. At the same time the formula φ to be evaluated is also transformed so that it can be evaluated on the modified graph. This procedure terminates on nowhere dense classes of graphs within a constant number of steps.

A key data structure used in the algorithm is a neighbourhood cover, i.e. a collection of connected subgraphs, called clusters, so that each neighbourhood of a vertex is contained in one of the clusters. The radius of a cover is the maximum radius of any of its clusters. The degree of a vertex in a cover is the number of clusters the vertex is contained in. An important result from [92] is that graphs from a nowhere dense class allow for small covers and that such a cover can be efficiently computed.

Theorem 8.1.2 (Grohe *et al.* [92], Theorem 6.2)**.** *Let $\mathcal{C}$ be a nowhere dense class of graphs. There is a function f such that for all $r \in \mathbb{N}$ and $\epsilon > 0$ and all graphs $G \in \mathcal{C}$ with $n \geq f(r, \epsilon)$ vertices, there exists an r-neighbourhood cover of radius at most $2r$ and maximum degree at most n^ϵ and this cover can be computed in time $f(r, \epsilon) \cdot n^{1+\epsilon}$. Furthermore, if $\mathcal{C}$ is effectively nowhere dense, then f is computable.*

In this theorem, f is a function of r and ϵ and depends on the class $\mathcal{C}$ in the sense that it is determined, for an effectively nowhere dense $\mathcal{C}$ by its parameter function (see Definition 8.2.1 for a definition of parameter function). To be precise, the algorithm needs to order the vertices of G to witness a weak colouring number of less than n^ϵ. The weak colouring number is an invariant of the graph that is guaranteed to be low for graphs from a nowhere dense class. The time bound $f(r, \epsilon) \cdot n^{1+\epsilon}$ is obtained using an algorithm for this from Nešetřil and Ossona de Mendez [132].

While the algorithm of [92] assumes that the input graph G comes from the class $\mathcal{C}$, we can say something more. For a fixed nowhere dense class $\mathcal{C}$, where we know the parameter function h, we can, given G, r and ϵ, compute a bound on the running time of the algorithm from Theorem 8.1.2. By running the algorithm to this bound, we have the following as a direct consequence of the proof of Theorem 8.1.2.

Lemma 8.1.3. *There is a computable function f and an algorithm A which given any graph G with n vertices and for any $r \in \mathbb{N}$ and $\epsilon > 0$ either computes an r-neighbourhood cover of radius at most $2r$ and maximum degree at most n^ϵ or determines that $G \notin \mathcal{C}$.*

At the core of the proof of Theorem 8.1.1 is the Rank-Preserving Locality Theorem. Given a neighbourhood cover $\mathcal{X}$ in a graph G, the algorithm iteratively removes bottleneck vertices and adds colours to the neighbourhoods of removed vertices to obtain a coloured graph denoted $G \star_{\mathcal{X}}^{q+1} q$. Here q is an integer parameter obtained from the first-order formula φ that we wish to evaluate in G. At the same time, φ is transformed into a formula $\hat{\varphi}$ that is *(a)* in the expanded signature of the $G \star_{\mathcal{X}}^{q+1} q$; and *(b)* in a logic FO^+ which enriches FO by allowing us to assert distances between vertices without the need for quantifiers. This ensures that the *local* sentence $\hat{\varphi}$ has the same quantifier rank as φ, giving us the Rank-Preserving Locality Theorem below. In the following statement,

a $(q+1,r)$-*independence sentence* is a formula asserting the existence of a distance-r-independent set of size $q+1$ of a particular colour.

Theorem 8.1.4 (Rank-Preserving Locality Theorem, Grohe *et al.* [92], Theorem 7.5)**.** *For every $q \in \mathbb{N}$ there is an r such that for every FO-formula $\varphi(x)$ of quantifier rank q there is an FO^+-formula $\hat{\varphi}(x)$, which is a Boolean combination of $(q+1,r)$-independence sentences and atomic formulas, such that for any graph G every r-neighbourhood cover $\mathcal{X}$ of G, and every $v \in V(G)$,*

$$G \models \varphi(v) \iff G \star_{\mathcal{X}}^{q+1} q \models \hat{\varphi}(v).$$

Furthermore, $\hat{\varphi}$ is computable from φ, and r is computable from q.

An important tool for constructing $G \star_{\mathcal{X}}^{q+1} q$ is a game characterisation of nowhere dense classes. The game has three parameters: ℓ, m, r. In the (ℓ, m, r) Splitter game two players Connector and Splitter play against each other. In each round Connector chooses a vertex u, and Splitter has to respond with a set A of vertices of size at most m in the r-neighbourhood of u. In the next round the graph is the neighbourhood of u with the vertices from A removed. If the graph is empty, Splitter wins. If Connector survives for more than ℓ rounds, she wins. Grohe *et al.* [92, Theorem 4.2] prove that if $\mathcal{C}$ is a nowhere dense class, then there are ℓ, m such that Splitter has a winning strategy on the $(\ell, m, 2r)$ Splitter game on every graph in $\mathcal{C}$.

The Splitter's strategy on a graph G (which can be efficiently computed) is the essential tool in the construction of $G \star_{\mathcal{X}}^{q+1} q$. The inductive procedure used to compute $G \star_{\mathcal{X}}^{q+1} q$ from G is outlined in [92, Proof of Theorem 8.1]. We note that the termination of the algorithm depends on the length of the game – which is bounded by a constant since $\mathcal{C}$ is nowhere dense. The strategy to compute Splitter's moves is described in [92, Remark 4.3]. Since the run time of the algorithm to compute $G \star_{\mathcal{X}}^{q+1} q$ only depends on q and the length of the Splitter game and we can compute this in advance, we can once again extract the fact that if we start with an arbitrary graph G, we can efficiently *either* transform it into $G \star_{\mathcal{X}}^{q+1} q$ *or* determine that it is not in the class $\mathcal{C}$. This is summed up in the following lemma.

Lemma 8.1.5. *Let $\mathcal{C}$ be a nowhere dense class of graphs. For every $\epsilon > 0$ there is an algorithm that runs in time $O(f(q) \cdot n^{1+\epsilon})$ for some function f, and which given a graph G returns $G \star_{\mathcal{X}}^{q+1} q$ or determines that $G \notin \mathcal{C}$.*

Theorem 8.1.4 reduces the problem of evaluating a formula of first-order logic to deciding a series of distance-r-independent set problems. So, the final ingredient is to show that this is tractable. Formally, the problem is defined as follows:

Distance Independent Set
Input: A graph G and $k, r \in \mathbb{N}$.
Parameter: $k + r$
Problem: Does G contain an r-independent set of size k?

The problem is shown to be FPT on nowhere dense classes of graphs [92, Theorem 5.1]. Since the runtime of the algorithm depends on the length of the Splitter game and Splitter's strategy, and this can be bounded in advance, [92, Theorem 5.1] can be restated as follows:

Lemma 8.1.6. *Let $\mathcal{C}$ be a nowhere dense class of graphs. Then there is an algorithm A and a computable function f such that for every $\epsilon > 0$ A runs in time $f(\epsilon, r, k) \cdot |V(G)|^{1+\epsilon}$ and either solves the* Distance Independent Set *problem or determines that $G \notin \mathcal{C}$.*

This is all we need to evaluate $\hat{\varphi}$ on $G \star_{\mathcal{X}}^{q+1} q$, which is equivalent to evaluating φ on G by Theorem 8.1.4.

8.2 Deciding first-order definable nowhere dense problems

The main result of [92] establishes that checking whether $G \models \varphi$ is FPT when parameterized by φ provided that G comes from a known nowhere dense class $\mathcal{C}$. Thus, the formula is arbitrary, but the graphs come from a restricted class. In Section 8.1 above we give an account of this proof from which we can extract the observation that the algorithm can be modified to work for an arbitrary input graph G with the requirement that the algorithm *may* simply reject the input if G is not in $\mathcal{C}$. This suggests a tractable way of deciding $G \models \varphi$ provided that φ defines a nowhere dense class. Now the graph is arbitrary, but the formula comes from a restricted class. We formalise the result in the following theorem:

We use the model theory based notions of parameterized complexity introduced in Section 2.2.4 below. Additionally we define the notions of slicewise nowhere dense and slicewise first-order definable. The idea of slicewise definability of problems in a logic was introduced by Flum and Grohe [71].

We first introduce the notion of slicewise nowhere dense. The idea is that each slice is nowhere dense, i.e. for each value k of the parameter, the subclass of problem instances with parameter value bounded above by k is nowhere dense.

Definition 8.2.1. Let σ be a graph signature. We say that a parameterized graph problem (Q, κ) is *slicewise nowhere dense* if there is a computable function h from pairs of integers to graphs such that for all $i \in \mathbb{N}$, we have if $G \in Q$ and $\kappa(G) \leq i\}$ then for all r we have $h(i, r) \not\preceq_r G$. We call h the *parameter function* of Q.

The idea of slicewise definability is formalised in a similar way: a problem is slicewise first-order definable, if, for each value k of the parameter, there is a first-order formula that defines the problem instances bounded above by k.

Definition 8.2.2 (Flum and Grohe [71])**.** Let σ be a signature. A parameterized problem (Q, κ) is *slicewise first-order definable* if there is a computable function $\varphi : \mathbb{N} \to \mathrm{FO}[\sigma]$ such that a σ-structure A with $\kappa(A) \leq i$ is in Q if, and only if, $A \models \varphi(i)$.

Given these two definitions we can state the main theorem of this chapter:

Theorem 8.2.3. *Let (Q, κ) be a problem that is slicewise first-order definable and slicewise nowhere dense. Then (Q, κ) is fixed-parameter tractable.*

Proof. In the following, for ease of exposition, we assume that an instance of the problem consists of a graph G and $\kappa(G) = i$ for some positive integer i.

Step 1: Compute φ and the parameter function. Since (Q, κ) is slicewise first-order definable, we can compute from i a first-order formula $\varphi = \varphi(i)$ which defines the

class of graphs $C_i = \{H \mid H \in Q \text{ and } \kappa(H) \leq i\}$. Moreover, since (Q, κ) is slicewise nowhere dense, we can compute from i an algorithm that computes the parameter function h for C_i.

Step 2: Obtain $\hat{\varphi}$ from φ. By the Rank-Preserving Locality Theorem (Theorem 8.1.4), we can compute from φ the formula $\hat{\varphi}$ and a radius r.

Step 3: Find a small cover $\mathcal{X}$ for G. By Lemma 8.1.3, we can either find a cover $\mathcal{X}$ for G, or reject if the algorithm determines that $G \notin C_i$.

Step 4: Simulate Splitter game to compute $G \star_{\mathcal{X}}^{q+1} q$. By Lemma 8.1.5 we obtain $G \star_{\mathcal{X}}^{q+1} q$ or reject if the algorithm determines that $G \notin C_i$.

Step 5: Evaluate $\hat{\varphi}$ on $G \star_{\mathcal{X}}^{q+1} q$. Finally we evaluate $\hat{\varphi}$ on $G \star_{\mathcal{X}}^{q+1} q$. To do this, we need to solve the distance independent set problem. We can do this by Lemma 8.1.6.

Since evaluating $\hat{\varphi}$ on $G \star_{\mathcal{X}}^{q+1} q$ is equivalent to evaluating φ on G this allows us to decide whether $G \in Q$.

□

8.3 Applications

In this Section we discuss some applications of Theorem 8.2.3 that demonstrate its power. We begin by considering simple edit distances.

8.3.1 Deletion distances

Recall from Definition 3.1.1 that a graph G has *deletion distance* k to a class $\mathcal{C}$ if there exists a set S of k vertices in G so that $G \setminus S \in \mathcal{C}$. Suppose (Q, κ) is a parameterized graph problem. We define the problem of deletion distance to Q as follows:

Deletion Distance to Q
Input: A graph G and $k, d \in \mathbb{N}$.
Parameter: $k + d$
Problem: Does G contain a set S of k vertices so that $\kappa(G \setminus S) \leq d$ and $G \setminus S \in Q$?

In many of the examples below, we define formulas of first-order logic by *relativisation*. For convenience, we define the notion here.

Definition 8.3.1. Let φ and $\psi(x)$ be first-order formulas, where ψ has a distinguished free variable x. The relativisation of φ by $\psi(x)$, denoted $\varphi^{[x.\psi]}$ is the formula obtained from φ by replacing all subformulas of the form $\exists v\, \varphi'$ in φ by $\exists v(\psi[v/x] \wedge \varphi')$, and all subformulas of the form $\forall v\, \varphi'$ in φ by $\forall v(\psi[v/x] \rightarrow \varphi')$. Here $\psi[v/x]$ denotes the result of replacing the free occurrences of x in ψ with v in a suitable way avoiding capture.

The key idea here is that $\varphi^{[x.\psi]}$ is true in a graph G if, and only if, φ is true in the subgraph of G induced by the vertices that satisfy $\psi(x)$. Note that the variable x that is free is ψ is bound in $\varphi^{[x.\psi]}$. Other variables that appear free in ψ remain free in $\varphi^{[x.\psi]}$. We stress this as it is needed in Proposition 8.3.5 where nested relativisations are used.

Proposition 8.3.2. *If (Q, κ) is slicewise nowhere dense and slicewise first-order definable then* Deletion Distance to Q *is* FPT.

Proof. It suffices to show that Deletion Distance to Q is also slicewise nowhere dense and slicewise first-order definable. For the latter, note that if φ_i is the first-order formula that defines the class of graphs $\mathcal{C}_i = \{G \mid \kappa(G) \leq i \text{ and } G \in Q\}$, then the class of graphs at deletion distance k to $\mathcal{C}_i$ is given by:

$$\exists w_1, \ldots, w_k \varphi_i^{[x.\theta_k]}$$

where $\theta_k(x)$ is the formula $\bigwedge_{1 \leq i \leq k} x \neq w_i$.

To see that Deletion Distance to Q is also slicewise nowhere dense, let h be the parameter function for Q. If the graph $h(i, r)$ has m vertices, then K_m is not a depth-r-minor of any graph in $\mathcal{C}_i$. Then a graph which has deletion distance k to $\mathcal{C}_i$ cannot have K_{m+k} as a depth-r-minor. Indeed, suppose $K_{m+k} \preceq_r G$ and $G \setminus S \in \mathcal{C}_i$ where S is a set of k vertices. Vertices from S can appear in the images of at most k vertices from K_{m+k} under the minor map. Thus, this minor map also witnesses that $K_m \preceq_r G \setminus S$, a contradiction. □

Consider as an example deletion distance k to maximum degree d.

Example 8.3.3. We can express that a vertex has no more than d different neighbours using the following formula:

$$\varphi_d(v) := \neg \exists v_1, \ldots, v_{d+1} \left(\left(\bigwedge_{i<j} v_i \neq v_j \right) \wedge \left(\bigwedge_i E(v, v_i) \right) \right).$$

So a graph G has maximum degree bounded by d if, and only if, $G \models \forall v \varphi_d(v)$. Let $\mathcal{C}_d$ be the class of graphs with maximum degree bounded by d. Then the following formula captures deletion distance k to maximum degree d:

$$\begin{aligned} \exists w_1, \ldots, w_k \forall v \left(\bigwedge_i v \neq w_i \right) \rightarrow \\ \left(\neg \exists v_1, \ldots, v_{d+1} \left(\left(\bigwedge_{i,j} v_i \neq w_j \right) \wedge \left(\bigwedge_{i<j} v_i \neq v_j \right) \right. \right. \\ \left. \left. \wedge \left(\bigwedge_i E(v, v_i) \right) \right) \right). \end{aligned}$$

Thus the technique introduced in this section is sufficient to show that deletion distance k to maximum degree d is fixed-parameter tractable parameterized by $k + d$.

Moser and Thilikos [129] showed that deleting k vertices to obtain a d-regular graph is fixed-parameter tractable parameterized by $k + d$. Since the class of d-regular graphs is also first-order definable and nowhere dense for any d, their result is also a consequence of Theorem 8.2.3.

8.3.2 Edit distances to graph classes defined by degree constraints

Instead of deletion distance (defined by deleting vertices), we can also consider more general graph editing distances (defined through more general edit operations on the graph), e.g. modifying the graph by adding or deleting edges. (See Definition 3.3.1 for a formal definition of edit distance.)

These more general editing operations are also first-order definable. For example, to obtain a formula that defines the graphs that are one edge addition away from a class of graphs $\mathcal{C}$ defined by the formula φ, we construct $\hat{\varphi}$ from φ by replacing all occurrences of $E(w_1, w_2)$ in φ by:

$$(w_1 = u \wedge w_2 = v) \vee (w_1 = v \wedge w_2 = u) \vee E(w_1, w_2).$$

Then the formula $\exists u \exists v \hat{\varphi}$ defines the class of graphs with edge addition distance 1 to $\mathcal{C}$, i.e. $G \models \exists u \exists v \hat{\varphi}$ if, and only if, G with an additional edge satisfies φ. This can easily be generalised to k edge additions: From a formula φ we can obtain a formula $\hat{\varphi}_k$ such that for any graph G we have that $G \models \hat{\varphi}$ if, and only if, there are pairs of vertices $u_1, v_1, \ldots, u_k, v_k \in V(G)$ such that G, with additional edges $u_1v_1, \ldots u_kv_k$, satisfies φ.

Similarly, given a formula ψ that defines a graph class $\mathcal{C}$, we can define a formula $\hat{\psi}$ by replacing all occurrences of $E(w_1, w_2)$ in ψ by:

$$(w_1 \neq u \vee w_2 \neq v) \wedge (w_1 \neq v \vee w_2 \neq u) \wedge E(w_1, w_2).$$

Then $\exists u \exists v \hat{\psi}$ defines the class of graphs with edge deletion distance 1 to $\mathcal{C}$. It should be clear that this can also be generalised to k edge deletions.

Thus, an analogue of Proposition 8.3.2 can be obtained for any edit distance where the allowed edit operations are a combination of vertex and edge deletions and additions. In the following we discuss this in more detail, where the class we are editing to is defined by degree constraints.

In his doctoral thesis Mathieson [123] studies the parameterized complexity of such graph editing problems with the aim of satisfying a variety of degree constraints. He defines the general template of Weighted Degree Constrained Editing (or just WDCE). In the following we explore one instance (that he refers to as WDCE_1^r) of a number of more general templates that also allow for weight functions of vertices and edges, as well as a different degree target for each vertex and a target for the sum of edge weights. This is just for the sake of simplicity here, the more general operations are also definable in first-order logic and give rise to nowhere dense graph classes. In the following we abbreviate the editing operations vertex deletion, edge deletion and edge addition as v, e and a respectively. Then for each non-empty $S \subseteq \{v, e, a\}$ define WDCE(S) as:

Weighted Degree Constrained Editing(S) (WDCE(S))
Input: A graph G, two integers k and d
Parameter: $k + d$
Problem: Can we obtain from G a graph G' using at most k editing operations from S only, such that all vertices of G' have degree d?

The problem is $\mathsf{W}[1]$-hard parameterized by k [89, 123], so we need to consider a joint parameterization. Mathieson [125] shows that the problem is fixed-parameter tractable

for any S and parameter $k + d$. Inspired by Stewart [149], Mathieson shows that the problem is first-order definable (with the size of the formula depending on k and d), by considering the incidence graph as a relational structure. (For the weighted version of the problem, he adds a unary relation for every possible weight.) Note that a graph G that can be edited to be regular in k steps can have maximal degree at most $k + d$ and G is therefore also nowhere dense. Thus Mathieson's fixed-parameter tractability results also follow directly from Theorem 8.2.3.

Building on this, Golovach [88] gives a concrete algorithm that edits a graph so that every vertex has a given degree at most d using at most k edge additions/deletions.

More recently, Mathieson [124] looks at more general versions of degree constraint problems. He considers three notions of regularity: *edge-degree-regular*, *edge-regular* and *strongly-regular*. He studies the problems of editing to these three notions of regularity.

The *edge-degree* of an edge uv is the sum of the degrees of the endpoints of $d(u) + d(v)$ and a graph is *edge-degree-regular* if all edges uv have the same edge-degree.

The two other notions combine the degrees of vertices and common neighbourhoods of endpoints of edges (and non-edges). A graph is (r, λ)-*edge-regular* if every vertex has degree r and every edge uv has $|N(u) \cap N(v)| = \lambda$. A graph is (r, λ, μ)-*strongly-regular* if it is (r, λ)-edge-regular and for every pair u, v of non-adjacent vertices we have $|N(u) \cap N(v)| = \mu$. This is the standard notion of a strongly regular graph as introduced by Bose [22].

Just as above we abbreviate the editing operations vertex deletion, edge deletion and edge addition as v, e and a respectively. We also abbreviate the three notions of regularity edge-degree regular, edge-regular and strongly regular as r_1, r_2, r_3 respectively. Then for each non-empty $S \subseteq \{v, e, a\}$ and each $r \in \{r_1, r_2, r_3\}$ define $\mathrm{RCE}(S, r)$ as:

> Regularity Constrained Editing(S, r) (RCE(S, r))
> **Input:** A graph G, integers k, d (and additionally an integer λ when $r = r_2$, and a pair of integers (λ, μ) when $r = r_3$)
> **Parameter:** $k + d$
> **Problem:** Can we obtain from G a graph G' using editing operations from S only, such that G' is r-regular (with the given parameters)?

Mathieson [124] shows that editing to these three notions of regularity is FPT parameterized by $k + d$, where d is the degree we are editing to and k is the number of allowed edits. This also follows from our meta-theorem: Just as in Example 8.3.3, it is easy to show that the class of graphs with edit distance k to an d-regular graph is first-order definable and has bounded degree, and is thus also nowhere dense. The additional constraints are also first-order definable, for example $|N(u) \cap N(v)| = \lambda$ can be expressed as follows:

$$\exists x_1 \ldots \exists x_\lambda . \left(\bigwedge_{i<j} x_i \neq x_j \bigwedge_i (E(x_i, u) \wedge E(x_i, v)) \right.$$
$$\left. \wedge \neg \exists y \left(\bigwedge_i (x_i \neq y) \wedge (E(u, y) \wedge E(v, y)) \right) \right)$$

So each of these problems is first-order definable and nowhere-dense, so it follows directly from Theorem 8.2.3 that these problems are fixed-parameter tractable with the combined parameter $k + d$.

8.3.3 Tree-depth

Recall that tree-depth (cf. Chapter 4) is a graph parameter that lies between the widely studied parameters vertex cover number and tree-width. It has interesting connections to nowhere dense graph classes, and can itself be interpreted as a distance measure (We showed that elimination distance, introduced in Chapter 5, can be thought of as a natural generalisation of tree-depth: tree-depth is just the elimination distance to the class of edgeless graphs. A graph has tree-depth k if, and only if, it has elimination distance k to the class of empty graphs; cf. Section 5.4). For convenience we repeat the usual definition (Definition 4.0.4) here:

Definition 8.3.4. The *tree-depth* of a graph G, written $td(G)$, is defined as follows:

$$td(G) := \begin{cases} 0, & \text{if } V(G) = \emptyset; \\ 1 + \min\{td(G \setminus v) \mid v \in V(G)\}, & \text{if } G \text{ is connected;} \\ \max\{td(H) \mid H \text{ a component of } G\}, & \text{otherwise.} \end{cases}$$

It is known that the problem of determining the tree-depth of a graph is FPT, with tree-depth as the parameter (see [133, Theorem 7.2]). We now give an alternative proof of this, using Theorem 8.2.3. First we show below that the class of graphs of tree-depth at most k is first-order definable. This is an interesting fact in itself and we are not aware if this has been established elsewhere.

Proposition 8.3.5. *For each $k \in \mathbb{N}$ there is a first-order formula φ_k such that a graph G has tree-depth k if, and only if, $G \models \varphi_k$.*

Proof. We use Lemma 4.1.1 which states the fact that in a graph of tree-depth less than k, there are no paths of length greater than 2^k. This allows us, in the inductive definition of tree-depth above, to replace the condition of connectedness (which is not first-order definable) with a first-order definable condition on vertices at distance at most 2^k.

Let $\mathrm{dist}_d(u, v)$ denote the first-order formula with free variables u and v that is satisfied by a pair of vertices in a graph G if, and only if, they have distance at most d in G. Note that the formula $\mathrm{dist}_d^{[x.x \neq w]}(u, v)$ is then a formula with three free variables u, v, w which defines those u, v which have a path of length d in the graph obtained by deleting the vertex w.

We can now define the formula φ_k by induction. Only the empty graph has tree-depth 0, so $\varphi_0 := \neg\exists v(v = v)$.

Suppose that φ_k defines the graphs of tree-depth at most k, let

$$\theta_k := (\forall u, v\, \mathrm{dist}_{2^{k+1}}(u, v)) \wedge \exists w(\varphi_k^{[x.x \neq w]}).$$

The formula θ_k defines the connected graphs of tree-depth at most $k+1$. Indeed, the first conjunct ensures that the graph is connected as no pair of vertices has distance greater than 2^{k+1} and the second conjunct ensures we can find a vertex w whose removal yields a graph of tree-depth at most k.

We can now define the formula φ_{k+1} as follows.

$$\varphi_{k+1} := (\forall u, v\, \mathrm{dist}_{2^{k+1}+1}(u, v) \rightarrow \mathrm{dist}_{2^{k+1}}(u, v)) \wedge \forall w \theta_k^{[x.\mathrm{dist}_{2^{k+1}}(w,x)]}.$$

The formula asserts that there are no pairs of vertices whose distance is strictly greater than 2^{k+1} and that for every vertex w, the formula θ_k holds in its connected component, namely those vertices which are at distance at most 2^{k+1} from w. □

It is well known that for any k, the class of graphs of tree-depth at most k is nowhere dense. It follows for example from the characterisation of nowhere dense classes in Theorem 2.3.33 via low tree-depth colourings (cf. Definition 2.3.32) that the class of graphs of tree-depth at most k has bounded expansion and is nowhere dense. Now, using Theorem 8.2.3, this fact together with Proposition 8.3.5 gives an alternative proof of the following:

Proposition 8.3.6. *The problem of determining, given a graph G and an integer k, whether G has tree-depth at most k is fixed-parameter tractable parameterized by k.*

While the proof of Proposition 8.3.2 shows that deletion distance to any slicewise first-order definable class is also slicewise first-order definable, Proposition 8.3.5 shows that elimination distance to the particular class of empty graphs is slicewise first-order definable. It does not establish this more generally for elimination distance to any slicewise nowhere dense class – that remains an open question. We conjecture that there is a slicewise nowhere dense class $\mathcal{C}$ for which elimination distance to $\mathcal{C}$ is not first-order definable.

CHAPTER 9

Distance parameters to sparse graph classes: two case studies

In this chapter we study the parameterized complexity of two more natural distance parameters, i.e. graph problems where the natural parameterization is a distance parameter.

First we consider the Clique Dominating Set problem on nowhere dense classes of graphs that are closed under induced subgraphs. The problem takes a graph G and numbers k, ℓ as input and asks whether k vertices are sufficient to dominate all the cliques of G of size ℓ. We show that Clique Dominating Set is FPT on nowhere dense classes of graphs, parameterized by k. The size of the smallest set that dominates all ℓ-cliques of a graph can be seen as a relaxed deletion distance to the class of graphs excluding K_ℓ as a subgraph.

The fixed-parameter tractability of Clique Dominating Set follows from a recent algorithmic meta-theorem of Grohe et al. [92]. We establish the result in a different way and believe our result is of independent interest.

The other problem we study in this chapter is the Anchored k-core problem. Recall from Definition 3.7.3 that the k-core of a graph G is the maximal subgraph of G in which all vertices have at least k neighbours. A b-anchored k-core (see Definition 9.2.1 below) relaxes the notion of the k-core and allows an additional number of b vertices in the core that may have degree less than k. The Anchored k-core problem asks, given a graph G, and integers $k, b, p \in \mathbb{N}$, whether G has a b-anchored k-core of size at least p. The size b of the 'anchor' can be seen as a distance measure to the the class of graphs that have a k-core of at least a certain size p.

The Anchored k-core problem is known to be W[2]-hard with respect to this distance parameter b, called the budget, so we consider joint parameterizations with another parameter: the tree-width of G. We show that the problem is FPT parameterized by the tree-width $tw(G)$ and budget b.

We also observe that the Anchored k-core problem can be expressed in monadic second-order logic (MSO), where the size of the MSO formula is bounded by a function of k and p. Using a generalisation of Courcelle's Theorem (Theorem 3.5.5) this implies that the Anchored k-core problem is also fixed-parameter tractable parameterized by the clique-width (Definition 3.5.2) of G and additional parameters k and p.

9.1 Clique Dominating Set

In this section we establish that Clique Dominating Set is FPT on nowhere-dense graph classes that are closed under induced subgraphs. The problem asks, given a graph G and two non-negative integers k and ℓ, whether there is a set of k vertices that dominates all the cliques of size ℓ. The parameterization we consider is the size of the clique dominating set k.

First we introduce some notation:

Definition 9.1.1. Let G be a graph. We say that a set of vertices $S \subseteq V(G)$ *dominates* a set of vertices $T \subseteq V(G)$ (or, that S is a *dominating set* for T) if there is a vertex $s \in S$ and a vertex $t \in T$ such that either $s = t$ or $st \in E(G)$.

Definition 9.1.2. Let G be a graph. An *ℓ-clique dominating set* for G is a set S that dominates all cliques of cardinality ℓ in G. The *ℓ-clique domination number* of G is the size of a minimal ℓ-clique dominating set for G.

The notion of *ℓ-clique domination number* is related to that of deletion distance (see Definition 3.1.1) in the following sense: Clearly, the deletion distance to the class of graphs excluding K_ℓ as a subgraph is an upper bound for the ℓ-clique domination number: if after deleting k vertices from the graph there are no cliques of size ℓ left, then the ℓ-clique domination number is at most k, because a subset of the k deleted vertices must be an ℓ-clique dominating set. However, the converse is not true: if we have an ℓ-clique dominating set of size k, then removing that set is not sufficient to remove all ℓ-cliques. As a simple example, consider a disjoint union of m copies of an ℓ-clique, all connected to a single additional apex vertex a. Of course this vertex a gives us a dominating set of size 1, but the deletion distance to the class of graphs excluding K_ℓ is m.

Thus the ℓ-clique domination number can be interpreted as a relaxed deletion distance to the class of graphs excluding K_ℓ as a subgraph. Given a graph G with ℓ-clique domination number k we can obtain a graph that does not contain K_ℓ as a subgraph by removing all the vertices in a minimal clique dominating set, and all the neighbours of these vertices. The class of graphs excluding K_ℓ as a subgraph is a sparse graph class.

Formally the Clique Dominating Set on a nowhere dense class $\mathcal{C}$ is defined as follows:

Clique Dominating Set
Input: A graph $G \in \mathcal{C}$, and non-negative integers k, ℓ
Parameter: k
Problem: Does G have ℓ-clique domination number at most k, i.e. is there a set of vertices $D \subseteq V(G)$ with $|D| \leq k$ such that every clique in G of size ℓ contains a vertex that is either in D or adjacent to a vertex in D?

The Clique Dominating Set problem was introduced by Flum and Grohe [72, p. 100] as an example of an A[2]-complete problem, which implies that it is W[2]-hard. Given its complexity status, it is unlikely to be FPT in general. That is why we are considering it here restricted to nowhere dense graph classes.

Flum and Grohe consider a version of the problem that is parameterized by $k + \ell$. However, we are considering the problem for graphs from a nowhere dense class $\mathcal{C}$, and thus there is an integer ℓ' such that $K_{\ell'}$ is not a subgraph of any graph in $\mathcal{C}$. So if $\ell > \ell'$ the

problems becomes trivial – the ℓ-clique dominating number is 0 for all G *in*$\mathcal{C}$. Otherwise ℓ is bounded by a constant. For that reason we consider a parameterization by just k.

The problem is first-order definable; it is expressed by the following formula:

$$\exists d_1 \ldots \exists d_k . \forall c_1, \ldots, c_\ell . \left(\bigwedge_{1 \leq i < j \leq \ell} E(c_i, c_j) \right) \rightarrow \left(\bigvee_{1 \leq i \leq k, 1 \leq j \leq \ell} d_i = c_j \vee E(d_i, c_j) \right)$$

A recent meta-theorem of Grohe et al. [92] establishes that first-order model checking is FPT on nowhere dense classes of graphs (parameterized by the size of the formula). So that result, together with the slicewise first-order definability of the problem, already implies that Clique Dominating Set is FPT on nowhere dense classes of graphs.

In this section we prove that Clique Dominating Set is FPT in a different way. Our approach is problem-specific, and uses an alternative characterisation of nowhere dense graph classes, and a powerful algorithmic result by Dawar and Kreutzer [52]. We thus believe our result is of independent interest.

In the following we use an alternative characterisation of nowhere dense graph classes called quasi-wideness (see Definition 2.3.26). Nešetřil and Ossona de Mendez [133] have shown that a class of graphs that is closed under taking induced subgraphs is nowhere dense if and only if it is quasi-wide (cf. Theorem 2.3.27).

Dawar and Kreutzer [52] extracted the algorithmic content of that theorem as a tool to prove that the Distance-d Dominating Set problem is fixed-parameter tractable on classes of graphs that are nowhere dense and closed under induced subgraphs; they proved the following:

Lemma 9.1.3 ([52, Lemma 9])**.** *Let $\mathcal{C}$ be a nowhere dense class of graphs and let h be the parameter function for $\mathcal{C}$, i.e. a function such that $K_{h(r)} \not\preceq_r G$ for all $G \in \mathcal{C}$. Then the following problem is solvable in time $O(|G|^2)$:*

Compute Quasi-Wide Parameters
Input: *$G \in \mathcal{C}, r, m \in \mathbb{N}, W \subseteq V(G)$ with $|W| > N(h(r), r, m)$*
Problem: *Compute a set $S \subseteq V(G), |S| \leq h(r) - 2$ and a set $A \subseteq W$ with $|A| \geq m$ such that A is r-scattered in $G - S$.*

Using this lemma we now prove that Clique Dominating Set problem is FPT on classes of graphs that are nowhere dense and closed under induced subgraphs.

Theorem 9.1.4. *The* Clique Dominating Set *problem is* FPT *on classes of nowhere dense graphs that are closed under induced subgraphs.*

Proof. Let $\mathcal{C}$ be a class of nowhere dense graphs and let $G \in \mathcal{C}$. We show that we can reduce the problem to a bounded number of subproblems with parameter $k - 1$.

Recall that since $\mathcal{C}$ is a nowhere dense class we can assume that ℓ is bounded by a constant. Thus we first can efficiently test whether a vertex is either contained in an ℓ-clique (i.e. its neighbourhood contains an $(\ell - 1)$-clique) or adjacent to an ℓ-clique (i.e. one of its neighbours is contained in an ℓ-clique). We remove all the vertices that are

neither contained in an ℓ-clique nor adjacent to one, obtaining an induced subgraph G' of G. Since $\mathcal{C}$ is closed under taking induced subgraphs, we have that $G' \in \mathcal{C}$.

Next we use Lemma 9.1.3. If $|V(G')| < N(h(3), 3, k+1)$, we just find a solution by brute force. Otherwise we obtain a subset $S \subseteq V(G')$ and a subset $A \subseteq V(G')$ with $|A| > k+1$ such that A is 3-scattered in $G' \setminus S$. Now every vertex $v \in A$ contains a clique in its neighbourhood $N_2^{G' \setminus S}(v)$, but for any $w \in A$, $w \neq v$, we have $N_3^{G' \setminus S}(v)(v) \cap N_3^{G' \setminus S}(v)(w) = \emptyset$. So if a vertex dominates one of the cliques, it cannot dominate any of the others. Thus $G' \setminus S$ does not contain an ℓ-clique dominating set of size k. So if there is an ℓ-clique dominating set for G' of size k it must contain a vertex from S.

Now for each $s \in S$, we remove s and $N(s)$ from G' to obtain the induced subgraph G'_s. We have that G has an ℓ-clique dominating set of size k if and only if one of the G'_s has an ℓ-clique dominating set of size $k-1$. Moreover, for each $s \in S$ we have $G'_s \in C$.

The size of S is bounded by the constant $h(3)$, and the number of reduction steps is bounded by k. Moreover, we can find an ℓ-clique in time $|G|^\ell$. So the overall time is bounded by $h(3)^k \cdot (|G|^2 \cdot |G|^\ell + |G|^2)$. □

Note that that this argument can be generalised to domination problems for other structures in a graph than ℓ-cliques. In fact, any problem variation where the structure can be efficiently tested for is FPT. For example the problem whether there is a set of k vertices that dominates all independent sets of size ℓ is fixed-parameter tractable parameterized by $k + \ell$.

9.2 The Anchored k-core problem

In this section we explore two different parameterizations of the Anchored k-core problem. Recall from Definition 3.7.3 that the k-core of a graph G is the largest induced subgraph H of G such that every vertex in H has at least k neighbours. The notion is closely related to that of the degeneracy of a graph: if G has a non-empty k-core, then G has degeneracy at least k, and, conversely, the degeneracy of G is the largest k for which G has a non-empty k-core.

The notion of k-core is important for applications. The k-core of a graph has been used in the context of social networks for some time now [148], where it is being used as a measure for the cohesion of the network. More recently, in the context of social networking websites, it was shown that the k-core of a social network graph is mostly congruent with the active user base: users contribute more content if at least a certain number of their friends do [31].

In the same context Bhawalkar et al. [16] used the idea of the k-core to suggest a simple model that measures how healthy a social networking website is: if a user is engaged with the site only if a sufficient number of friends, say k friends, are, then we can observe a process of 'unravelling' until only the k-core is left. This raises an interesting question: can the size of the k-core be increased by influencing the engagement of users? If it is possible to convince a certain amount of users to stay, even if their number of friends is below the threshold, can the 'unravelling' be stopped much earlier?

This is exactly the Anchored k-core problem, which was recently introduced by Bhawalkar *et al.* [16]. We are given a budget b, a degree threshold k and a target size p and the question is whether it is sufficient to convince b users to stay, so that after the

process of unravelling stops, where a user leaves if they have fewer than k friends, there are still p users left.

To formalize the idea of fixing vertices, we introduce the notion of an *anchored k-core*:

Definition 9.2.1. Let G be a graph and let $b, k \geq 0$. A *b-anchored k-core* of G is a subgraph H of G such that at most b vertices of H have degree less than k. We refer to the set of vertices of H with degree less than k as *anchor set*.

Note that the size b of the 'anchor' gives us an interesting notion of distance. It measures how far a graph is from unravelling: Instead of deleting vertices, we ask how many vertices have to be 'fixed' in order to guarantee a k-core of size at least p. This also gives us a lower bound for the graph editing problem of how many vertices we need to add in order to guarantee a k-core of size at least p.

Formally, the Anchored k-core problem is defined as follows.

Anchored k-core
Input: A graph G, positive integers $k, b, p \in \mathbb{N}$
Problem: Does G contain a b-anchored k-core of size at least p, i.e. is there a subgraph H of G of size $|H| \geq p$ such that at most b vertices in H have degree less than k?

Bhawalkar et al. [16] show that for $k = 2$, there is a polynomial-time algorithm. However, for $k \geq 3$ the problem becomes much harder and does not even admit a polynomial time approximation algorithm: for all $k \geq 3$, it is NP-hard to obtain an $O(n^{1-\epsilon})$ approximation for every positive constant $\epsilon > 0$, i.e. they prove that it is NP-hard to distinguish between instances of Anchored k-core instances where an optimal anchored k-core contains $\Omega(n)$ vertices and where $O(b)$ vertices in the anchored k-core suffice [16, Theorem 3].

So the problem is quite hard from the perspective of classical complexity theory. This motivates Bhawalkar et al. [16] to study the parameterized complexity of the problem. They consider the budget b as parameter, but it turns out that for every $k \geq 3$, the problem is W[2]-hard with respect to the parameter b.

The hardness of the problem parameterized by the budget motivates Bhawalkar et al. [16] to consider additional structural parameterisations. They successfully show that there is an algorithm that runs in time $O((3(k+1)^2)^w \cdot b^2 \cdot n)$, where w is the tree-width of the graph. The result is obtained using standard dynamic programming techniques on the tree decomposition of the graph. This shows that the problem is FPT parameterized by k and the tree-width of the input graph.

Chitnis et al. [36] establish further hardness results: they show Anchored k-core is W[1]-hard parameterized by p, even for $k = 3$. This improves the result of Bhawalkar et al. [16] since we can always assume that $p \geq b$. They also show that the problem remains NP-hard even on planar graphs for all $k \geq 3$, and even if the maximum degree of the graph is $k + 2$.

But what about parameterizing by the natural distance parameter b on restricted graph classes? Chitnis et al. [36] show that Anchored k-core is FPT on planar graphs parameterized by b for all $k \geq 7$.

In our analysis we consider further joint parameterizations, combining the distance parameter b with the structural distance parameters tree-width (Definition 3.4.2) and clique-width (Definition 3.5.2).

In Section 9.2.1 we study the Anchored k-core problem parameterized by b and tree-width and show that it is fixed-parameter tractable.

In Section 9.2.2 we show that the Anchored k-core is MSO-definable, with the size of the formula depending on k and b. Using a generalisation of Courcelle's Theorem to clique-width this implies that Anchored k-core is FPT parameterized by k, b and the clique-width of the graph.

Lastly we talk about some open questions in Section 9.2.3, in particular we discuss how the techniques in Section 9.2.1 can be used to extend the result to more general parameterizations, such as the combination of b and the degeneracy of the graph.

9.2.1 Anchored k-core parameterized by budget and tree-width

In this section we show that the Anchored k-core problem is FPT parameterized by the budget and tree-width of the graph. We do this by splitting the problem into two cases: we deal separately with the case where k is small and the case where k is large.

We generalise a lemma from Chitnis et al. [36] for the case where k is large. Chitnis et al. [36] observe that, for any $p \geq 0$, it is fairly straightforward to write down a first-order sentence that is true on graphs that have an anchored k-core of size at most p. (The definition literally translates to FO.) The size of the first-order formula depends on b, k and p, but since $b, k \leq p$, it is bounded by a function of p. So if we know that the graph must contain a small k-core, the problem reduces to evaluating first-order logic formulas.

A closer look at the proof by Chitnis et al. [36] reveals that the argument is not specific to planar graphs. They use the well known degeneracy (Definition 3.7.1) of planar graphs, i.e. the property of planar graphs that the number of edges is bounded by $3|V| - 6$, to show that the ratio of high degree vertices to all vertices is bounded:

Lemma 9.2.2. *Let G be a planar graph, $n = |G|, m = |\{v \in G \mid \deg(v) \geq k\}|$. Then, if $k \geq 7$, we have that $m/n < \frac{6}{7}$.*

Chitnis et al. [36] use the lemma to show that the Anchored k-core problem is FPT on planar graphs parameterized by b for all $k \geq 7$. This is an easy application of this lemma: For sufficiently large k ($k > 7$), the above lemma implies that the k-core must be small, and we can check it using a small first-order formula.

In the following we prove a more general version of Lemma 9.2.2. The property of planar graphs that is used in the proof is their degeneracy. So a similar statement can be made for any graph of bounded degeneracy. It again follows that for sufficiently large k, the k-core must be small, so we solve the problem by evaluating a small first-order formula. We can generalise the proof of Lemma 9.2.2 to graphs of bounded degeneracy as follows:

Lemma 9.2.3. *Let $G = (V, E)$ be a d-degenerate graph with no isolated vertices. Let $n = |V(G)|$ and $m = |\{v \in G \mid \deg(v) \geq k\}|$. Then, if $k > 2d$, we have that $m/n \leq 1 - \frac{1}{2d}$.*

Proof. Since G is d-degenerate, we have that $|E(G)| \leq dn$. So the following holds:

$$\begin{aligned} 2dn &\geq 2|E(G)| \\ &= \sum_{v \in V(G)} \deg(v) \\ &\geq (2d+1)m + n - m. \end{aligned}$$

Rearranging the terms gives $m/n \leq 1 - \frac{1}{2d}$. □

This is all we need to prove the main theorem of the section. The proof combines two techniques: We apply Lemma 9.2.3 to solve the case where k is large. In that case we use that the input graph is sparse and it therefore cannot contain a large k-core. Thus we can reduce the problem to a model checking problem with a formula of bounded size. For small k, we can use the algorithm described by Bhawalkar et al. [16] for solving the problem on graphs of bounded tree-width.

Theorem 9.2.4. *The* Anchored k-Core *problem is* FPT *parameterized by the tree-width t of the input graph and the budget b.*

Proof. We assume without loss of generality that the input graph G does not contain isolated vertices (otherwise we just remove them).

We distinguish the cases where k is small and large. If $k < 2t$, we can use the algorithm in [16], which runs in time $O((3(k+1)^2)^t \cdot b^2 \cdot n) = O(f(t)n^3)$.

If $k > 2t$, it follows from Lemma 3.7.2 that because G has tree-width t it is also t-degenerate. Assume that H is an optimal k-core with anchor set $B \subseteq H$. Since H is an induced subgraph of G, it is also t-degenerate and we get the following using the above lemma:

$$1 - \frac{|B|}{|H|} = \frac{|H| - |B|}{|H|} = \frac{|H \setminus B|}{|H|} < 1 - \frac{1}{2t}.$$

Rearranging the terms gives us $|H| < 2t|B| < 2tb$. So we know that an anchored k-core has size bounded by the parameter and express the problem in a first-order sentence of size bounded by a function of $2tb$. Since G has bounded tree-width we can evaluate the sentence in FPT time. □

In fact, the proof technique can be generalized to show that the problem is FPT parameterized by b on all t-sparse (t-degenerate) graphs for large enough values of the degree threshold k, i.e. whenever $k > 2t$. However, we do not know how to deal with small values of k, and this part of the problem remains open.

9.2.2 Anchored k-core parameterized by degree, size of the core and clique-width

Here we consider the Anchored k-core problem parameterized by the degree k, the size of the core p and the clique-width of the input graph. We show that the problem is FPT with a joint parameterization of these three parameters. This is a consequence of the fact that the problem can be expressed in MSO with the size of the formula bounded by a function of $k + p$, and a generalisation of Courcelle's theorem [42].

The main advantage that MSO gives us over first-order logic is that we can talk about sets without having to be specific about their size. So we can specify a lower bound for the k-core and an upper bound for the budget: For a second-order variable X and an integer x we can write a formula $|X| \geq x$ that is true if, and only if, the set X has at least x elements, with the size of the formula bounded by a function of x. Similarly, for a second-order variable X and an integer x we can write a formula $|X| \leq x$ that holds if, and only if, the set X has at most x elements with the size of the formula bounded by a function x. It is also clear that for every non-negative integer k there is a first-order formula δ_k^S such that $G \models \delta_k^S(v)$ if, and only if, $\deg_{G[S]}(v) \geq k$.

Also note that, if $b \geq p$, the problem becomes trivial, because we can just fix any b vertices to obtain a large enough anchored k-core. So every graph of size at least p is a positive instance. Thus, in the following we assume without loss of generality that $b < p$.

To match second-order syntax, the problem can be rephrased as follows: Is there a set of vertices K of size at least p and a subset B of K of size at most b such that all vertices in $K \setminus B$ have degree in $G[K]$ at most k? Or as a formula:

$$\exists K. \left(|K| \geq p \wedge \left(\exists B. |B| \leq b \wedge \forall v. \left(K(v) \wedge \not B(v) \rightarrow \delta_k^{K \setminus B}(v) \right) \right) \right)$$

So together with Theorem 3.5.5 this implies that the problem is FPT parameterized by clique-width, k and p:

Theorem 9.2.5. *The anchored k-core problem is* FPT *parameterized by the clique-width c, k and p.*

9.2.3 Open questions

An interesting question is whether the technique used for tree-width in Section 9.2.1 can be pushed further. Our Lemma 9.2.3 is a general statement about graphs of bounded degeneracy. We show that the problem is FPT parameterized by b and t on all t-degenerate graphs for large enough values of the degree k. Is there a way to deal with small values of k?

The most general result one could aim for with the techniques used here is to prove that Anchored k-core is FPT parameterized by degeneracy and budget. Bounded degeneracy is a very general notion of sparsity, and clases of bounded degeneracy include many other graph classes, for example graphs of bounded expansion. We know the problem for such classes is FPT for sufficiently large k. The open question is whether there is an FPT algorithm that can deal with the case where k is small – that would be enough to show fixed-parameter tractability with parameters b and t.

Another interesting area for the problem is the study of the directed version. Since it was designed with application to social networks in mind, that would be very natural, as many such networks are better modelled using directed graphs. Chitnis et al. [35] also discuss the directed version of the problem, and prove some hardness and tractability results. It seems possible that some of the approaches discussed here might be helpful in the direction version as well.

chapter 10

Further remarks and future research directions

In this dissertation we introduced the term *distance parameter* to capture the idea of a graph parameter that measures how far the graph is from being contained in some class. We showed that many common graph parameteres can be interpreted in that way, and presented several results to emphasise the usefulness of distance parameters.

Chapter 4 discussed the distance parameter *tree-depth* in some detail, and established fixed-parameter tractability and hardness results for a number of problems parameterized by tree-depth.

In Chapter 5 we introduced a new distance parameter, *elimination distance to a class* $\mathcal{C}$. The parameter measures distance in a way that generalises the notion of vertex deletion in a natural way by allowing deletions from every component in each deletion step. We show that the parameter tree-depth is a specific instance of elimination distance to a class $\mathcal{C}$, where $\mathcal{C}$ is the class of edgeless graphs. Indeed, elimination distance generalises deletion distance in the same way that tree-depth generalises vertex cover number.

We gave an application of the elimination distance in Chapter 6, where we demonstrate that Graph Isomorphism is FPT parameterized by elimination distance to bounded degree, extending a result of Bouland et al. [23].

In Chapter 7 we also studied the parameter itself. We showed that given a finite set of minors M that characterise a minor-closed class of graphs $\mathcal{C}$, we can find the minors that characterise the graphs with elimination distance k to $\mathcal{C}$. This yields an explicit algorithm for determining whether a graph has elimination distance k to a minor closed class $\mathcal{C}$ if the set M of minors excluded by $\mathcal{C}$ is available.

We believe that these applications demonstrate a lot of potential for elimination distance, and that there are likely many more applications for it.

Algorithmic meta-theorems are algorithmic results that apply not only to a single problem, but to a whole class of problems. Our result in Chapter 7 is an algorithmic meta-theorem, because it applies to all minor-closed graph classes. In Chapter 8 we established another algorithmic meta theorem: that any graph problem that is slicewise nowhere dense and slicewise first-order definable is FPT.

Lastly, in Chapter 9 we discussed two problems parameterized by their natural distance parameters, and showed several fixed-parameter tractability results.

This dissertation has demonstrated that distance is a useful lens through which to

view the parameterized complexity of graph problems. It certainly has the potential to help discover interesting research questions. By being a guide in finding the line between tractability and intractability for specific problems, it helps improve our understanding of the complexity theoretic landscape.

10.1 Future research directions

In the following we discuss some threads that the research in this dissertation has opened and that we think are worth pursuing in further research.

10.1.1 Elimination distance

We introduced the new parameter *elimination distance to a class* $\mathcal{C}$ in Chapter 5 and the notion seems to not only leave more room for the study of Graph Isomorphism, initiated in Chapter 6, but also offer promise for defining tractable parameterizations for many other graph problems. These are directions that bear further investigation.

10.1.1.1 Graph Isomorphism parameterized by elimination distances

An immediate question that arises from our work in Chapter 6 is what happens when we consider other classes of graphs for which Graph Isomorphism is known to be tractable.

For instance, what can we say about Graph Isomorphism when parameterized by elimination distance to planar graphs? Unfortunately techniques such as those deployed in Chapter 6 are unlikely to work in this case. Our techniques rely on identifying a canonical subgraph which defines an elimination tree into the class where GI is tractable, which would be the class of planar graphs in this case. As an obstacle, in the case of planar graphs, consider the class of graphs which are a subdivision of K_5 (i.e. which can be obtained from K_5 by replacing each edge by a path). It is easy to see that each of the graphs in that class is deletion distance 1 away from planarity. However, it is impossible to identify a canonical vertex to delete: the deletion of any vertex yields a planar graph. New techniques are needed to steer around this obstacle.

10.1.1.2 Elimination distance to bounded degree

Another open question raised by Chapter 5 and Chapter 6 is whether elimination distance to the class of graphs of maximal degree d is FPT. When d is 0, the elimination distance is just the tree-depth of a graph, and this case is covered by our result in Chapter 8. A graph of maximal degree 1 is just a union of disjoint edges and isolated vertices, so if $\hat{\mathcal{C}}$ is the class of such graphs, then we just have $ed_{\hat{\mathcal{C}}}(G) = td(G)$ or $ed_{\hat{\mathcal{C}}}(G) = td(G) + 1$. For values of $d > 1$, it is not clear whether elimination distance is first-order definable. Note that for $d = 2$ we have that the elimination distance is almost the generalised tree-depth of G (see Section 5.4.1) and the question is also open.

One possible strategy is to establish that elimination distance to $\mathcal{C}$ is first-order definable. If $\mathcal{C}$ is a graph class with bounded expansion, then so is the class of graphs with elimination distance k to $\mathcal{C}$ (cf. Section 5.3), and we might then apply our result from Chapter 8. Indeed, the question for which classes $\mathcal{C}$ elimination distance to $\mathcal{C}$ is first-order definable is an interesting question. Even more interesting would be negative results that show it is not definable.

But even if it is not possible to establish that elimination distance to a class $\mathcal{C}$ is first-order definable, it might still be FPT for another reason. So another, more general version of the first question is whether for any nowhere dense and first-order definable $\mathcal{C}$, elimination distance to $\mathcal{C}$ is FPT.

10.1.1.3 Elimination distance to edgeless graphs: tree-depth

The special case of elimination distance to the class of edgeless graphs is just the tree-depth of a graph. Tree-depth is the special case where the class we are considering the elimination to is the class of edgeless graphs. Although it has been studied under different names in the last twenty years now, it has not received as much attention as other parameters such as tree-width. The particular structure of the decompositions, and the fact that computing the decompositions is itself fixed-parameter tractable, lends itself to dynamic programming approaches. (See Section 4 for a more detailed discussion.)

What makes the study of tree-depth particularly interesting is that it lies between vertex cover number and tree-width as a parameter. Many problems are known to be fixed-parameter tractable parameterized by vertex cover number, but either known to be hard parameterized by tree-width or with an unknown complexity status. We discuss several such problems in Chapter 4, but there are many other problems that could be studied. We mention some examples in Section 4.6.

10.1.2 Algorithmic meta theorems

For a long time the study of algorithmic meta theorems has focused on evaluating first-order logic formulas on classes of sparse graphs. Evaluation of first-order logic on (non-trivial) graph classes closed under taking subgraphs is well understood now: Grohe et al. [92] showed that first-order evaluation is FPT on nowhere dense classes of graphs, and Dawar and Kreutzer [110] proved that it is not FPT on somewhere dense graph classes (unless $\mathsf{FPT} = \mathsf{AW}[*]$).

These results for nowhere dense classes are a great success, but a class that truly captures the tractable instances of first-order evaluation would have to include some somewhere dense graph classes as well and would need to satisfy certain closure properties, such as the closure under taking complements (see below).

This has motivated the study of graph classes that are not closed under taking subgraphs, e.g. graph classes that are just closed under taking induced subgraphs, or graph classes that include dense graphs? For example Bova et al. [24, 25] and Gajarsky et al. [77] have recently resolved the question for partially ordered sets (i.e. directed graphs that are a reflexive, antisymmetric, and transitive) of bounded width.

10.1.2.1 Graph classes including dense graphs

As mentioned above, the result by Grohe et al. [92] applies only to graph classes closed under taking subgraphs. A more ambitious goal would be to understand what kind of structure allows easy evaluation of first-order logic. Ideally we would be able to identify the class of graphs that precisely captures the graphs where evaluation is FPT. In fact, a class of structures that does this, would have to be closed under first-order interpretations. For example, the possibility of negating the atoms would mean the graphs have to be

closed under negations. This means that sparsity might not be a natural constraint for first-order logic.

So another interesting case that seems closely related to our methods, but is not an immediate consequence is that of classes that are given by first-order interpretations from nowhere dense classes of graphs. For instance, consider the problem of determining the deletion distance of a graph to a disjoint union of complete graphs. This problem, known as the cluster vertex deletion problem is known to be FPT (see [99]). The class of graphs that are disjoint unions of cliques is first-order definable but certainly not nowhere dense and so the method of Chapter 8 does not directly apply. However, this class is easily shown to be interpretable in the nowhere dense class of forests of height 1. Can this fact be used to adapt the methods of Chapter 8 to this class?

A further step would be to see if there is a nice way to characterise such classes. An interesting way to approach this would be to consider the connection of nowhere dense classes to stability theory noticed by Adler and Adler [5].

Another line of attack would be to use the idea of shrub-depth, recently introduced by Ganian et al. [80], and consider low shrub-depth colourings defined in an analogous way to low tree-depth colourings, which characterise nowhere dense classes of graphs. Such classes are a potential candidate for being exactly the graph classes where first-order logic is fixed-parameter tractable (as opposed to graph classes closed under taking minors / (induced) subgraphs). We expect this would yield a graph class with interesting algorithmic properties, that would be a natural generalisation of nowhere dense graph classes.

10.1.2.2 Order-invariant first-order logic on graphs of bounded degree

Another question worth considering is whether order-invariant first-order logic is fixed-parameter tractable on graphs of bounded degree. Order-invariant first-order logic is a variation of first-order logic that allows the use of an order relation in the formulas, but the truth value of a formula must not depend on the particular order (i.e. it either holds for all orders or for none). It was shown that order-invariant first-order logic is strictly more expressive than first-order logic [94]. A better understanding of this extension would be helpful in database theory, where the physical existence of the database in memory imposes an implicit order on the database. The database can be understood as a relational structure and the user of the database has no control over the order, so order-invariant queries are particularly interesting.

Grohe and Schwentick [94] showed that whether a tuple in a structure satisfies a query defined in order-invariant first-order logic only depends on a small neighbourhood of the tuple. This seems to suggest the possibility that order-invariant first-order logic might be fixed-parameter tractable on bounded degree graphs as well. However, the proof is very different from Gaifmans proof and does not involve the translation into a Boolean combination of local sentences, so a new approach would be needed.

10.1.3 Descriptive complexity

There are also several interesting open questions left regarding the descriptive complexity of graph classes featured in this dissertation. For example, it is an open question whether, on nowhere-dense graph classes, MSO_2 allows us to express more properties of graphs than just MSO. Recall that MSO is all of first-order logic extended with set variables

and quantification over set variables. MSO_2 also allows quantification over sets of edges. Courcelle [41] proved that MSO and MSO_2 have the same expressivity on k-degenerate graphs, and it would be interesting to extend this to nowhere-dense graph classes.

10.1.4 Subquadratic FPT algorithms

The growth of data sets processed in industry over the last decade has brought new algorithmic challenges: even algorithms that run in polynomial time, with a polynomial of moderate degree, have formidable runtimes on huge datasets. Motivated by these 'big data' developments, the threshold for tractability seems to have shifted: A problem need not be considered tractable on large datasets just because it has a polynomial-time algorithm. The line between tractability and intractability for such large input has moved to low degree polynomials; in fact, subquadratic computation is the 'new P' and problems that are provably at least quadratic are the new 'NP-hard' problems. The hardness results are usually conditional on stronger complexity theoretic assumptions such as the Strong Exponential Time Hypothesis (SETH) introduced by Impagliazzo, Paturi, and Zane [101].

The SETH was first used to prove conditional lower bounds for NP-hard problems (see e.g. [117] for a survey in the context of parameterized complexity). More recently lower bounds for problems known to be in P have been found, often showing that they require at least quadratic runtime (see e.g. [1, 3, 21, 26, 27, 143, 157]).

Just as for NP-hard problems, there is an interest to find ways around the hardness for problems that have been shown to be hard (if the SETH is true). Until recently this has mostly led to the study of approximation algorithms for such problems. Parameterized complexity theory has only very recently been applied ([2, 74, 86]).

The application of parameterized complexity seems very promising because it can both provide additional insights into what makes the problem hard (i.e. at least quadratic), and offer algorithms that work well in some circumstances (whenever the parameter is small). Giannopoulou et al. [86] have formulated a research program to systematically study polynomial-time fixed-parameter tractable algorithms. They have introduced new complexity classes. Given a polynomial-bounded function $p(n)$, they define the class FPT$(p(n))$, the class P-FPT$(p(n))$ as the problems that can be solved in time $O(k^t \cdot p(n))$ for some constant t, and lastly the class PL-FPT$(p(n))$, defined as P-FPT(n).

An interesting question in this context is whether it is possible to prove algorithmic meta theorems, comparable to what we present in Chapter 8, to show that certain problems are in PL-FPT. Given a fixed first-order formula, it can be evaluated in polynomial time on a graph and the problem of evaluating a given first-order formula is fixed-parameter tractable parameterized by the size of the formula on nowhere dense classes of graphs. Is there maybe a fragment of first-order logic that can be evaluated in linear FPT time with only polynomial parameter dependence?

What about the evaluation of a fixed first-order formula, and distance parameters to sparse graph classes? Could it be that the first-order model checking problem of a fixed formula, parameterized by the tree-width of the graph, is in PL-FPT? Or, much simpler, parameterized by the vertex cover number? Unfortunately this is not the case if the SETH holds, even for first-order formulas with just 3 variables. To see this, consider a graph construction from another hardness result: Abboud et al. [2] argue that there is no subquadratic fixed-parameter algorithm with subexponential parameter dependence for the Diameter problem unless the SETH is true. They use a reduction from CNF-

SAT to Diameter, discovered by Roditty and Vassilevska Williams [143]. A look at the reduction reveals that given a CNF-SAT instance with n variables and m clauses we obtain a graph with $O(2^{n/2+m+m})$ vertices and $O(2^{n/2} \cdot m)$ edges.

Abboud et al. [2] show that if there is an algorithm that solves the Diameter problem on a graph with b edges and some parameter k in time $O(p(k) \cdot b^{2-\epsilon})$, then CNF-SAT can be solved in time $O(p(k) \cdot poly(m,n) \cdot 2^{n-\frac{\epsilon n}{2}})$, which implies that the SETH is false if $p(k)$ is subexponential in m and n.

A closer look at the reduction in [143] also reveals that the constructed graph has tree-width m and in fact vertex cover number m. This implies that the Diameter problem, parameterized by tree-width or vertex cover number, is not in PL-FPT unless the SETH is false.

Moreover, the reduction does not require us to solve the whole Diameter problem. It is sufficient to distinguish the cases where the diameter is 2 or 3. Of course 'diameter $\neq 2$' can be easily expressed in first-order logic; all we need to say is that there is a pair of vertices that are not equal, not adjacent, and also not connected via a third vertex, or formally:

$$\exists x \exists y . x \neq y \wedge \neg E(x,y) \wedge \neg \left(\exists z . E(x,z) \wedge E(z,y)\right).$$

This implies that model checking a fixed formula from 3-variable first-order logic parameterized by tree-width or vertex cover number is not in PL-FPT unless the SETH is false.

Proposition 10.1.1. *Evaluating a fixed* 3*-variable first-order formula on graphs is not fixed-parameter tractable parameterized by the vertex cover number of the graph, unless the SETH is false.*

However, the research of the parameterized complexity of problems in P has just begun, and there are many interesting questions to study, including potential meta theorems.

Bibliography

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-Time Hardness of LCS and other Sequence Similarity Measures. *CoRR abs/1501.07053*, 2015.

[2] Amir Abboud, Virginia Vassilevska Williams, and Joshua R Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter. *CoRR abs/1506.01799*, 2015.

[3] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of Faster Alignment of Sequences. *ICALP*, 8572(Chapter 4):39–51, 2014.

[4] Abhijin Adiga, Jasine Babu, and L Sunil Chandran. Polynomial Time and Parameterized Approximation Algorithms for Boxicity. In Dimitrios M Thilikos and Gerhard J Woeginger, editors, *Parameterized and Exact Computation - 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings*, pages 135–146, 2012.

[5] Hans Adler and Isolde Adler. Nowhere dense graph classes, stability, and the independence property. *CoRR abs/1406.4718*, page 9, November 2010.

[6] Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, January 2008.

[7] Takuya Akiba, Christian Sommer, and Ken-ichi Kawarabayashi. Shortest-path queries for complex networks: exploiting low tree-width outside the core. In Elke A Rundensteiner, Volker Markl, Ioana Manolescu, Sihem Amer-Yahia, Felix Naumann, and Ismail Ari, editors, *15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings*, pages 144–155, New York, New York, USA, 2012. ACM Press.

[8] Fadi A Aloul. Symmetry in Boolean Satisfiability. *Symmetry*, 2(2):1121–1134, 2010.

[9] Amihood Amir, Moshe Lewenstein, and Ely Porat. Faster algorithms for string matching with k mismatches. *Journal of Algorithms*, 50(2):257–275, 2004.

[10] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, June 1991.

[11] László Babai. Graph Isomorphism in Quasipolynomial Time. *CoRR abs/1512.03547*, 2015.

[12] László Babai and Eugene M Luks. Canonical Labeling of Graphs. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 171–183, New York, NY, USA, 1983. ACM.

[13] László Babai and Shlomo Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[14] Lars Backstrom, Cynthia Dwork, and Jon M Kleinberg. Wherefore art thou R3579X?: anonymized social networks, hidden patterns, and structural steganography. *Communications of the ACM*, 54(12):133–141, 2011.

[15] Umberto Bertelè and Francesco Brioschi. On Non-serial Dynamic Programming. *Journal of Combinatorial Theory, Series B*, 14(2):137–148, 1973.

[16] Kshipra Bhawalkar, Jon M Kleinberg, Kevin Lewi, Tim Roughgarden, and Aneesh Sharma. Preventing Unraveling in Social Networks: The Anchored k-Core Problem. *SIAM Journal on Discrete Mathematics*, 29(3):1452–1475, 2015.

[17] Hans L Bodlaender. Some Classes of Graphs with Bounded Treewidth. Technical report, Utrecht University, 1988.

[18] Hans L Bodlaender, Jitender S Deogun, Klaus Jansen, Ton Kloks, Dieter Kratsch, Haiko Müller, and Zsolt Tuza. Rankings of Graphs. *SIAM Journal on Discrete Mathematics*, 11(1):168–181, 1998.

[19] Hans L Bodlaender, Thomas Wolle, and Arie M C A Koster. Contraction and Treewidth Lower Bounds. *Journal of Graph Algorithms and Applications*, 10(1):5–49, 2006.

[20] Paolo Boldi, Violetta Lonati, Massimo Santini, and Sebastiano Vigna. Graph fibrations, graph isomorphism, and PageRank. *ITA*, 40(2):227–253, 2006.

[21] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the Square - On the Complexity of Quadratic-Time Solvable Problems. *CoRR abs/1406.4718*, cs.CC, 2014.

[22] Raj C Bose. Strongly regular graphs, partial geometries and partially balanced designs. *Pacific Journal of Mathematics*, 13(2):389–419, 1963.

[23] Adam Bouland, Anuj Dawar, and Eryk Kopczyński. On Tractable Parameterizations of Graph Isomorphism. *Parameterized and Exact Computation*, 7535:218–230, 2012.

[24] Simone Bova, Robert Ganian, and Stefan Szeider. Model checking existential logic on partially ordered sets. In Thomas A Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS 2014, Vienna, Austria, July 14 - 18, 2014*, pages 21:1–21:10, New York, New York, USA, 2014. ACM Press.

[25] Simone Bova, Robert Ganian, and Stefan Szeider. Quantified conjunctive queries on partially ordered sets. *Theoretical Computer Science*, January 2016.

[26] Karl Bringmann. Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 661–670. IEEE, 2014.

[27] Karl Bringmann and Marvin Künnemann. Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping. *CoRR abs/1502.01063*, 2015.

[28] Jannis Bulian and Anuj Dawar. Graph Isomorphism Parameterized by Elimination Distance to Bounded Degree. *Parameterized and Exact Computation*, 8894(Chapter 12):135–146, 2014.

[29] Jannis Bulian and Anuj Dawar. Fixed-parameter Tractable Distances to Sparse Graph Classes. In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pages 236–247, 2015.

[30] Jannis Bulian and Anuj Dawar. Graph Isomorphism Parameterized by Elimination Distance to Bounded Degree. *Algorithmica*, pages 1–20, August 2015.

[31] Moira Burke, Cameron Marlow, and Thomas M Lento. Feed me: motivating newcomer contribution in social network sites. *CHI*, pages 945–954, 2009.

[32] Leizhen Cai. Fixed-Parameter Tractability of Graph Modification Problems for Hereditary Properties. *Information Processing Letters*, 58(4):171–176, 1996.

[33] Leizhen Cai. Parameterized Complexity of Vertex Colouring. *Discrete Applied Mathematics*, 127(3):415–429, 2003.

[34] Jeremy J Carroll. Signing RDF Graphs. *International Semantic Web Conference*, 2870(Chapter 24):369–384, 2003.

[35] Rajesh Hemant Chitnis, Fedor V Fomin, and Petr A Golovach. Parameterized Complexity of the Anchored k-Core Problem for Directed Graphs. In Anil Seth and Nisheeth K Vishnoi, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, December 12-14, 2013, Guwahati, India*, pages 79–90, 2013.

[36] Rajesh Hemant Chitnis, Fedor V Fomin, and Petr A Golovach. Preventing Unraveling in Social Networks Gets Harder. In Marie des Jardins and Michael L Littman, editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*, 2013.

[37] Peter I Corke, Stefan Hrabar, Ronald A Peterson, Daniela Rus, Srikanth Saripalli, and Gaurav S Sukhatme. Autonomous Deployment and Repair of a Sensor Network using an Unmanned Aerial Vehicle. *ICRA*, pages 3602–3608, 2004.

[38] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, September 2009.

[39] Brouno Courcelle. Graph Rewriting: An Algebraic and Logic Approach. In *Logic and Automata – History and Perspectives*, pages 193–242. Amsterdam University Press, Cambridge, MA, USA, 1990.

[40] Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1), March 1990.

[41] Bruno Courcelle and Joost Engelfriet. Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach. *Cambridge University Press 2012*, 2012.

[42] Bruno Courcelle, Johann A Makowsky, and Udi Rotics. Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width. *Theory of Computing Systems*, 33(2):125–150, April 2000.

[43] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[44] Margaret B Cozzens. Higher and Multi-Dimensional Analogues of Interval Graphs. *Dissertation Abstracts International Part B: Science and Engineering*, 42(4):1981, 1981.

[45] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. On Cutwidth Parameterized by Vertex Cover. *Algorithmica*, 68(4):940–953, November 2012.

[46] Paul T Darga, Mark H Liffiton, Karem A Sakallah, and Igor L Markov. Exploiting structure in symmetry detection for CNF. *DAC '04 Proceedings of the 41st annual Design Automation Conference*, pages 530–534, 2004.

[47] Anuj Dawar. Finite Model Theory on Tame Classes of Structures. In *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, pages 2–12, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[48] Anuj Dawar. Structure and Specification as Sources of Complexity. In Ravi Kannan and K Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTC 2009*, pages 407–416, 2009.

[49] Anuj Dawar. Homomorphism preservation on quasi-wide classes. *Journal of Computer and System Sciences*, 76(5):324–332, 2010.

[50] Anuj Dawar, Martin Grohe, and Stephan Kreutzer. Locally Excluding a Minor. *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 270–279, 2007.

[51] Anuj Dawar and Yuguo He. Parameterized Complexity Classes under Logical Reductions. In *Mathematical Foundations of Computer Science 2009*, pages 258–269. Springer Berlin Heidelberg, Berlin, Heidelberg, August 2009.

[52] Anuj Dawar and Stephan Kreutzer. Domination Problems in Nowhere-Dense Classes. In Ravi Kannan and K Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, pages 157–168. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009.

[53] Anuj Dawar and Stephan Kreutzer. Parameterized complexity of first-order logic. *Electronic Colloquium on Computational Complexity*, 131(131):1–18, 2009.

[54] Jitender S Deogun, Ton Kloks, Dieter Kratsch, and Haiko Müller. On Vertex Ranking for Permutations and Other Graphs. In Patrice Enjalbert, Ernst W Mayr, and Klaus W Wagner, editors, *STACS 94, 11th Annual Symposium on Theoretical Aspects of Computer Science, Caen, France, February 24-26, 1994, Proceedings*, pages 747–758, 1994.

[55] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, 2010.

[56] Rodney G Downey and Michael R Fellows. *Parameterized Complexity*. Springer Verlag, October 2012.

[57] Zdeněk Dvořák, Daniel Kráľ, and Robin Thomas. Deciding First-Order Properties for Sparse Graphs. *Annual IEEE Symposium on Foundations of Computer Science*, 0:133–142, 2010.

[58] Eduard Eiben, Robert Ganian, and Stefan Szeider. Solving Problems on Graphs of High Rank-Width. In *Algorithms and Data Structures*, pages 314–326. Springer International Publishing, Cham, August 2015.

[59] Paul Erdős and András Hajnal. On chromatic number of graphs and set-systems. *Acta Mathematica Academiae Scientiarum Hungarica*, 17(1-2):61–99, March 1966.

[60] Sergei Evdokimov and Ilia Ponomarenko. Isomorphism of coloured graphs with slowly increasing multiplicity of Jordan blocks. *Combinatorica*, 19(3):321–333, 1999.

[61] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *SIAM-AMS Proceedings*, 7:43–73, 1974.

[62] Jean-Loup Faulon. Isomorphism, Automorphism Partitioning, and Canonical Labeling Can Be Solved in Polynomial-Time for Molecular Graphs. *Journal of Chemical Information and Computer Sciences*, 38(3):432–444, 1998.

[63] 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. *Information and Computation*, 209(2):143–153, 2011.

[64] Michael R Fellows, Jiong Guo, Hannes Moser, and Rolf Niedermeier. A generalization of Nemhauser and Trotter's local optimization theorem. *Journal of Computer and System Sciences*, 77(6):1141–1158, 2011.

[65] Michael R Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, January 2009.

[66] Michael R Fellows, Daniel Lokshtanov, Neeldhara Misra, Matthias Mnich, Frances A Rosamond, and Saket Saurabh. The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number. *Theory of Computing Systems*, 45(4):822–848, 2009.

[67] Michael R Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A Rosamond, and Saket Saurabh. Graph Layout Problems Parameterized by Vertex Cover. *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, 5369(Chapter 28):294–305, 2008.

[68] Michael R Fellows and Frances A Rosamond. The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number. In S Barry Cooper, Benedikt Lowe, and Andrea Sorbi, editors, *Computation and Logic in the Real World, Third Conference on Computability in Europe, CiE 2007, Siena, Italy, June 18-23, 2007, Proceedings*, pages 268–277, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[69] Jiří Fiala, Petr A Golovach, and Jan Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theoretical Computer Science*, 412(23):2513–2523, May 2011.

[70] Ion S Filotti and Jack N Mayer. A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus. In *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 236–243, New York, New York, USA, April 1980. ACM.

[71] Jörg Flum and Martin Grohe. Fixed-Parameter Tractability, Definability, and Model-Checking. *SIAM Journal on Computing*, 31(1):113–145, January 2002.

[72] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer Science & Business Media, May 2006.

[73] Fedor V Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 470–479. IEEE, 2012.

[74] Fedor V Fomin, Daniel Lokshtanov, Michal Pilipczuk, Saket Saurabh, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *CoRR abs/1511.01379*, 2015.

[75] Markus Frick and Martin Grohe. Deciding First-Order Properties of Locally Tree-Decomposable Graphs. In *Automata, Languages and Programming*, pages 331–340. Springer Berlin Heidelberg, Berlin, Heidelberg, January 1999.

[76] Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM (JACM)*, 48(6):1184–1206, November 2001.

[77] Jakub Gajarský, Petr Hlinený, Daniel Lokshtanov, Jan Obdrzálek, Sebastian Ordyniak, M S Ramanujan, and Saket Saurabh. FO Model Checking on Posets of Bounded Width. *CoRR abs/1504.04115*, 2015.

[78] Jakub Gajarský, Petr Hlinený, Jan Obdrzálek, Sebastian Ordyniak, Felix Reidl, Peter Rossmanith, Fernando Sanchez Villaamil, and Somnath Sikdar. Kernelization Using Structural Parameters on Sparse Graph Classes. In *Algorithms – ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 529–540. Springer, Berlin, Heidelberg, September 2013.

[79] Robert Ganian, Petr Hlinený, Joachim Kneis, Alexander Langer, Jan Obdrzálek, and Peter Rossmanith. On Digraph Width Measures in Parameterized Algorithmics. In Jianer Chen and Fedor V Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark 2009*, pages 185–197, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[80] Robert Ganian, Petr Hlinený, Jaroslav Nešetřil, Jan Obdrzálek, Patrice Ossona de Mendez, and Reshma Ramadurai. When Trees Grow Low: Shrubs and Fast MSO1. In *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, pages 419–430, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[81] Robert Ganian, M S Ramanujan, and Stefan Szeider. Discovering Archipelagos of Tractability for Constraint Satisfaction and Counting. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1670–1681, Philadelphia, PA, December 2015. Society for Industrial and Applied Mathematics.

[82] Robert Ganian, Friedrich Slivovsky, and Stefan Szeider. Meta-kernelization with structural parameters. *Journal of Computer and System Sciences*, 82(2), March 2016.

[83] Serge Gaspers and Stefan Szeider. Backdoors to Satisfaction. In *Automata, Languages and Programming*, pages 287–317. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[84] Georg Gati. Further annotated bibliography on the isomorphism disease. *Journal of Graph Theory*, 3(2):95–109, 1979.

[85] Archontia C Giannopoulou, Paul Hunter, and Dimitrios M Thilikos. LIFO-search: A min-max theorem and a searching game for cycle-rank and tree-depth. *Discrete Applied Mathematics*, 160(15):2089–2097, 2012.

[86] Archontia C Giannopoulou, George B Mertzios, and Rolf Niedermeier. Polynomial Fixed-Parameter Algorithms: A Case Study for Longest Path on Interval Graphs. *CoRR abs/1506.01652*, 2015.

[87] Petr A Golovach. Editing to a Connected Graph of Given Degrees. In Erzsebet Csuhaj-Varju, Martin Dietzfelbinger, and Zoltan Esik, editors, *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, pages 324–335, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[88] Petr A Golovach. Editing to a Graph of Given Degrees. In *Parameterized and Exact Computation*, pages 196–207. Springer International Publishing, Cham, September 2014.

[89] Petr A Golovach and George B Mertzios. Graph Editing to a Given Degree Sequence. *arXiv.org*, January 2016.

[90] Seth Greenblatt, Thayne Coffman, and Sherry Marcus. Behavioral Network Analysis for Terrorist Detection. *Emergent Information Technologies and Enabling Policies for Counter-Terrorism*, 2006.

[91] Martin Grohe. Logic, graphs and algorithms. In Thomas Wilke, Jörg Flum, and Erich Grädel, editors, *Logic and Automata – History and Perspectives*, pages 357–422. Amsterdam University Press, Amsterdam, 2007.

[92] Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 89–98, New York, New York, USA, May 2014. ACM.

[93] Martin Grohe and Dániel Marx. Structure Theorem and Isomorphism Test for Graphs with Excluded Topological Subgraphs. *SIAM Journal on Computing*, 44(1):114–159, 2015.

[94] Martin Grohe and Thomas Schwentick. Locality of order-invariant first-order formulas. *ACM Transactions on Computational Logic*, 1(1):112–130, July 2000.

[95] Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A Structural View on Parameterizing Problems: Distance from Triviality. *Parameterized and Exact Computation*, 3162:162–173, 2004.

[96] Frank Gurski and Egon Wanke. The Tree-Width of Clique-Width Bounded Graphs without Kn,n. In *Graph-Theoretic Concepts in Computer Science*, pages 196–205. Springer Berlin Heidelberg, Berlin, Heidelberg, June 2000.

[97] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8(1-2):171–186, March 1976.

[98] Wilfrid Hodges. *A Shorter Model Theory.* Cambridge University Press, 1997.

[99] Falk Hüffner, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Fixed-Parameter Algorithms for Cluster Vertex Deletion. *Theory of Computing Systems*, 47(1):196–217, 2010.

[100] Neil Immerman. *Descriptive Complexity.* Springer Science & Business Media, New York, NY, January 1999.

[101] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[102] Haim Kaplan, Ron Shamir, and Robert Endre Tarjan. Tractability of Parameterized Completion Problems on Chordal, Strongly Chordal, and Proper Interval Graphs. *SIAM Journal on Computing*, 28(5):1906–1922, 1999.

[103] Richard M Karp. Reducibility Among Combinatorial Problems. In Raymond E Miller, James W Thatcher, and Jean D Bohlinger, editors, *Complexity of Computer Computations Proceedings of a symposium on the Complexity of Computer Computations, Yorktown Heights, New York*, pages 85–103, Boston, MA, 1972. Complexity of Computer Computations.

[104] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, March 2012.

[105] Dean Kelley. Technical Report Column. *SIGACT News*, 45(4):48–57, December 2014.

[106] Daniel J Kleitman and Douglas B West. Spanning Trees with Many Leaves. *SIAM Journal on Discrete Mathematics*, 4(1):99–106, 1991.

[107] Jan Kratochvíl. A Special Planar Satisfiability Problem and a Consequence of Its NP-completeness. *Discrete Applied Mathematics*, 52(3):233–252, 1994.

[108] Stefan Kratsch and Pascal Schweitzer. Isomorphism for graphs of bounded feedback vertex set number. In *SWAT'10: Proceedings of the 12th Scandinavian conference on Algorithm Theory*, pages 81–92, Berlin, Heidelberg, June 2010. Springer-Verlag.

[109] Stephan Kreutzer. Algorithmic Meta-theorems. *Parameterized and Exact Computation*, 5018(Chapter 3):10–12, 2008.

[110] Stephan Kreutzer and Anuj Dawar. Parameterized Complexity of First-Order Logic. *Electronic Colloquium on Computational Complexity*, 16:131, 2009.

[111] Richard E Ladner. On the Structure of Polynomial Time Reducibility. *Journal of the ACM*, 22(1):155–171, 1975.

[112] John M Lewis and Mihalis Yannakakis. The Node-Deletion Problem for Hereditary Properties is NP-Complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[113] Leonid Libkin. *Elements of Finite Model Theory*. Springer Science & Business Media, Berlin, Heidelberg, 2004.

[114] Don R Lick and Arthur T White. k-Degenerate graphs. *Canadian Journal of Mathematics*, 22:1082–1096, 1970.

[115] Steven Lindell. A logspace algorithm for tree canonization (extended abstract). In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 400–404, New York, New York, USA, July 1992. ACM.

[116] Richard J Lipton, Lawrence Snyder, and Y Zalcstein. The Complexity of Word and Isomorphism Problems for Finite Groups. Technical Report 91, Yale University, March 1977.

[117] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, pages 41–72, 2011.

[118] Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Fixed-Parameter Tractable Canonization and Isomorphism Test for Graphs of Bounded Treewidth. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 186–195, 2014.

[119] José Luis López-Presa, Antonio Fernández Anta, and Luis Núñez Chiroque. Conauto-2.0: Fast Isomorphism Testing and Automorphism Group Computation. *CoRR abs/1406.4718*, cs.DS, 2011.

[120] Eugene M Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.

[121] Dániel Marx. Parameterized coloring problems on chordal graphs. *Theoretical Computer Science*, 351(3):407–424, 2006.

[122] Dániel Marx. Parameterized Complexity and Approximation Algorithms. *Computer Journal*, 51(1):60–78, 2008.

[123] Luke Mathieson. *The Parameterized Complexity of Degree Constrained Editing Problems*. PhD thesis, Durham University, Durham, 2009.

[124] Luke Mathieson. Graph Editing Problems with Extended Regularity Constraints. *CoRR abs/1406.4718*, cs.CC, 2015.

[125] Luke Mathieson and Stefan Szeider. Editing graphs to satisfy degree constraints: A parameterized approach. *Journal of Computer and System Sciences*, 78(1):179–191, 2012.

[126] Brendan D McKay. Practical Graph Isomorphism. Technical report, Vanderbilt University, 1981.

[127] Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60, January 2014.

[128] Gary Miller. Isomorphism testing for graphs of bounded genus. In *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 225–235, New York, New York, USA, April 1980. ACM.

[129] Hannes Moser and Dimitrios M Thilikos. Parameterized complexity of finding regular induced subgraphs. *Journal on Discrete Algorithms*, 7(2):181–190, 2009.

[130] Jaroslav Nešetřil and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *European Journal of Combinatorics*, 27(6):1022–1041, August 2006.

[131] Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion I. Decompositions. *European Journal of Combinatorics*, 29(3):760–776, 2008.

[132] Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion II. Algorithmic aspects. *European Journal of Combinatorics*, 29(3):777–791, April 2008.

[133] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity. Graphs, Structures, and Algorithms*. Springer, 2012.

[134] Jaroslav Nešetřil and Saharon Shelah. On the order of countable graphs. *European Journal of Combinatorics*, 24(6):649–663, 2003.

[135] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, February 2006.

[136] Miles Ohlrich, Carl Ebeling, Eka Ginting, and Lisa Sather. SubGemini: Identifying SubCircuits using a Fast Subgraph Isomorphism Algorithm. In *30th Conference on Design Automation*, pages 31–37, New York, New York, USA, 1993. ACM Press.

[137] Adolfo Piperno. Search Space Contraction in Canonical Labeling of Graphs. *CoRR abs/0804.4881*, 2008.

[138] Ilia Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. *Journal of Soviet Mathematics*, 55(2):1621–1643, 1991.

[139] Ronald C Read and Derek G Corneil. The graph isomorphism disease. *Journal of Graph Theory*, 1(4):339–363, 1977.

[140] Fred S Roberts. On the boxicity and cubicity of a graph. In William Thomas Tutte, editor, *Recent Progress in Combinatorics. Proceedings of a conference, Waterloo, Ontario, May* . Recent Progresses in Combinatorics, 1969.

[141] Neil Robertson and Paul D Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[142] Neil Robertson and Paul D Seymour. Graph Minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[143] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 515–524, New York, New York, USA, 2013. ACM Press.

[144] Peter Roopnarine. Graphs, Networks, Extinction and Paleocommunity Food Webs. *Nature Preceedings*, May 2010.

[145] Alejandro A Schäffer. Optimal Node Ranking of Trees in Linear Time. *Information Processing Letters*, 33(2):91–96, 1989.

[146] Uwe Schöning. Graph Isomorphism is in the Low Hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, 1988.

[147] Detlef Seese. Linear Time Computable Problems and First-Order Descriptions. *Mathematical Structures in Computer Science*, 6:505–526, 1996.

[148] Stephen B Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, September 1983.

[149] Iain A Stewart. On the fixed-parameter tractability of parameterized model-checking problems. *Information Processing Letters*, 106(1):33–36, 2008.

[150] Lei Tang and Huan Liu. Graph Mining Applications to Social Network Analysis. *Managing and Mining Graph Data*, 40(Chapter 16):487–513, 2010.

[151] Seinosuke Toda. Computing Automorphism Groups of Chordal Graphs Whose Simplicial Components Are of Small Size. *IEICE - Transactions on Information and Systems*, E89-D(8):2388–2401, August 2006.

[152] Alan M Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, January 1937.

[153] Ryuhei Uehara, Seinosuke Toda, and Takayuki Nagoya. Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. *Discrete Applied Mathematics*, 145(3):479–482, January 2005.

[154] Vijay V Vazirani. *Approximation Algorithms*. Springer Science & Business Media, December 2002.

[155] Klaus Wagner. Über eine Eigenschaft der ebenen Komplexe. *Mathematische Annalen*, 114(1):570–590, December 1937.

[156] Mark A Walch and Donald T Gantz. *Pictographic matching: a graph-based approach towards a language independent document exploitation platform*. a graph-based approach towards a language independent document exploitation platform. ACM, New York, New York, USA, November 2004.

[157] Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1867–1877, Philadelphia, PA, 2014. Society for Industrial and Applied Mathematics.

[158] Koichi Yamazaki, Hans L Bodlaender, Babette de Fluiter, and Dimitrios M Thilikos. Isomorphism for Graphs of Bounded Distance Width. *Algorithmica*, 24(2):105–127, 1999.

[159] Mihalis Yannakakis. Edge-Deletion Problems. *SIAM Journal on Computing*, 10(2):297–309, 1981.

[160] Mihalis Yannakakis. The Complexity of the Partial Order Dimension Problem. *SIAM Journal on Algebraic Discrete Methods*, 3(3):351–358, July 2006.

Index