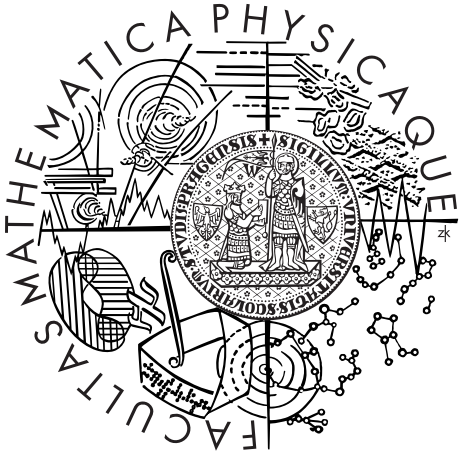
Charles University in Prague
Faculty of Mathematics and Physics

**MASTER THESIS**




Vojtěch Tůma

# Nowhere-dense classes of graphs

Department of Applied Mathematics

Supervisor of the master thesis: doc. Mgr. Zdeněk Dvořák, Ph.D.
Study programme: Computer Science
Specialization: Discrete Models and Algorithms

Prague 2013

I would like to thank to

- my advisor doc. Mgr. Zdeněk Dvořák, Ph.D., for being a great advisor,
- my collaborator Mgr. Martin Kupec, for being a great collaborator and drawer,
- my parents, for being great parents,
- all colleagues from our department for perfect working environment,
- all the people listed in the Bibliography section for giving me the opportunity to spend hours and hours reading through their brilliant work.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... on ..... Author signature

Název práce: Řídké třídy grafů

Autor: Vojtěch Tůma

Katedra: Katedra aplikované matematiky

Vedoucí bakalářské práce: doc. Mgr. Zdeněk Dvořák, Ph.D., IUUK

Abstrakt: V této práci se zabýváme řídkými třídami grafů a jejich vlastnostmi využitelnými pro návrh algoritmů a datových struktur. Speciálně se zaměřujeme na nedávno zavedené koncepty omezené expanse a stromové hloubky, které zavedli J. Nešetřil a P. Ossona de Mendez. V této práci nejprve podáme stručný úvod do problematiky a shrneme důležité výsledky a nástroje z parametrisované složitosti a algoritmické teorie modelů.

Hlavní část této práce, aplikace teoretických poznatků, přináší dva nové výsledky z oblasti dynamických datových struktur. První slouží k udržování dekomposice grafu s omezenou stromovou hloubkou, druhá počítá výskyty zadaného podgrafu v udržovaném grafu. Časová i prostorová složitost operací obou struktur je při použití na řídké třídy grafů nízká.

Klíčová slova: řídká teorie grafů, algoritmické metavěty, dynamické datové struktury

Title: Nowhere-dense classes of graphs

Author: Vojtěch Tůma

Department: Department of Applied Mathematics

Supervisor of the master thesis: doc. Mgr. Zdeněk Dvořák, Ph.D., IUUK

Abstract: In this thesis we study sparse classes of graphs and their properties usable for design of algorithms and data structures. Our specific focus is on the concepts of bounded expansion and tree-depth, developed in recent years mainly by J. Nešetřil and P. Ossona de Mendez. We first give a brief introduction to the theory as whole and survey tools and results from related areas of parametrised complexity and algorithmic model theory.

The main part of the thesis, application of the theory, presents two new dynamic data structures. The first is for keeping a tree-depth decomposition of a graph, the second counts appearances of fixed subgraphs in a given graph. The time and space complexity of operations of both structures is guaranteed to be low when used for sparse graphs.

Keywords: sparse graph theory, algorithmic metatheorems, dynamic data structures

# Contents

# Chapter 1

# Introduction

One of the most prominent subfields of graph theory is definitely the theory of sparse graphs, where the object of study is not the family of all graphs, but somehow restricted sub-family. This restriction can be of various forms, for instance, we can simply prescribe a bound on the maximum number of edges in the graph, or we can define a specific parameter for graphs and investigate only those graphs for which this parameter is small. The fundamental example of sparse graphs are the planar graphs, which have very natural definition—they can be drawn in a plane without edge crossings—and also posses many good structural properties, e.g., they have very few edges and contain small separators. Planar graphs have been studied at least since the 19th century, when the famous Four colour conjecture was formulated. Since then this area has evolved profoundly, both in terms of new conjectures and solved problems.

What makes the idea of restricting graphs so attractive? One argument is that the restrictions carry with them very useful properties which make the graphs much easier to work with, compared to unstructured chaos of the family of all graphs. However, the "true reason" is much deeper. Ideally, we would like to be able to solve any problem on graphs in fast, exact and reliable way. But the results from complexity theory tell us, that unless P is equal to NP, this is not possible – thus we have to make a compromise. One way is to have algorithms that are either not exact and approximate only, or algorithms that are not reliable and work using some randomness. The other way is to restrict the family of investigated graphs. And very often, this restriction is exactly of the "sparse" or "structural" style.

However, not only algorithmic applications motivate research in this area. Structural results about graph classes are interesting on their own and often provide tools for proving theorems about graphs in general. Also, many results about graphs can be generalised to relational structures and model theory. In turn, tools and language of model theory have proven very applicable for purely graph theoretical results.

Yet another area this thesis is related to is the area of dynamic data structures. In the usual static setting, we are given a property, e.g. "graph is triangle free", and we are supposed to be able to decide for a given graph whether it has the property or not. In the dynamic setting, the graph we are dealing with dynamically changes over time. That is, we must implement operations of two kinds, *query*—reporting whether the graph has the property or not, and *modify*—for

instance, adding or removing an edge to the graph.

There are again various motivations for this kind of approach. Firstly, many natural phenomena, for instance a graph of some social network, are from the definition changing over time. Instead of having a static algorithm and running it repeatedly after each change, it makes more sense to keep a "dynamic track of information". The same applies for the situation, where we are running some subprocedure on a structure we are gradually building. Examples of such applications are given later.

In the rest of the thesis, the reader first encounters a gentle and informal introduction to the vast areas of parametrised complexity, sparse graph theory, algorithmic model theory and dynamic data structures. Emphasis is on results and tools with algorithmic flavour, especially on those that will be applied or are related to the next part, where two data structures the author has developed are presented with all rigorous detail.

The first data structure serves to keep a tree-depth decomposition of a graph. Decompositions are a corner-stone of structural graph theory. Possibly the most famous structural result, the Graph Minor Theorem of Robertson and Seymour, investigates the decompositional parameter of tree-width. And as if that were not enough, tree-width gained even more popularity with the Courcelle's theorem, which puts it in the context of algorithmic meta-theorems. Both results rely heavily on the associated decomposition of the graph, and both have inspired other results that investigate other parameters and decompositions in these contexts. More information about tree-depth and its importance will be given in 2.3.1.

The second data structures serves to keep track of occurrences of some specified patterns in a larger structure. A simple example thereof is counting of triangles in a planar graph, which can be modified by adding and removing edges, under the condition that it remains planar. This so-called subgraph problem has a rich history in both static and dynamic setting, and it has found applications in areas like social networking or bioinformatics.

# Chapter 2

# Preliminaries

## 2.1 Parameterised Complexity

The purpose of this section is to introduce notions from the field of computational and parametrised complexity, and shortly survey its historical development as well as its important tools.

We use standard terminology and notation from graph theory, see e.g. the classical textbook by Diestel [28]. Throughout the thesis, by $G$ we usually denote a big "sparse" graph whereas $H$ stands for small graph which can be arbitrary. Very often we will work with (rooted) trees and forests, which we denote by $T$ and $F$ respectively.

For the analysis of algorithms and their running time we use the standard random-access machine (RAM) model, see the book [20] for details. We make use of the asymptotic $O$-notation, and sometimes hide expression independent of $|V(G)|$ in it – such expressions might depend on some structural parameter of the considered graph class, because we consider that parameter to be very small compared to the size of the graph. Also, when we say that an algorithm runs in polynomial (linear) time, we mean that the dependence on $|V(G)|$ is polynomial (linear); dependence on other parameters may be worse.

We assume familiarity with basic notions of computational complexity, namely with NP-hardness and reductions. For reference, see the book by Arora and Barak [9]. The notion of NP-hardness, which appeared in Cook's paper in 1971 [19], serves as a first step for distinguishing whether for a problem there is an efficient algorithm. Informally, a problem is NP-hard, if an efficient algorithm for it gives an algorithm for every problem solvable in polynomial time with a help of guessing oracle. A more practical criterion of determining that a certain problem $P$ is NP-hard is showing that some existing NP-hard problem $P'$ can be solved using $P$. A list of known NP-hard problems is huge – already in 1972 Karp [62] has provided a list of 21 NP-hard combinatorial problems, for example vertex cover, maximum clique or 0-1 integer programing. For more information, see the book by Garey and Johnson [55]

The chief goal of parametrised complexity is to provide a finer classification of problems that are NP-hard, and identify in what cases we can achieve an efficient solution of a problem that is generally NP-hard. As the name suggest, the approach to do this is to measure the time complexity of an algorithm not with respect just to $n$, but to some other value, called *parameter*, which usually

represents some property of the graph or is just the maximum allowed size of a solution. Let us give some specific examples.

### 2.1.1 Examples of parameterised problems

A *vertex cover* of a graph $G$ is a set of vertices $A$ such that every edge of $G$ is incident to at least one vertex of $A$. The problem of finding minimum vertex cover for $G$ is NP-hard. However, if we consider parameterisation by the size of the solution, that is, we want to decide whether $G$ has a vertex cover of size at most $k$, there is an algorithm which runs in the time $O(2^k \cdot n)$. The idea behind it is an example of the so-called depth-bounded search tree method. Consider arbitrary edge $uv$ of $G$. If there is a vertex cover $A$ of $G$, then at least one of $u$, $v$ must be in $A$. Therefore we branch into two subproblems, one assumes $u \in A$ and works in $G - \{u\}$, the other $v \in A$ and works in $G - \{v\}$. After investigating $k$ edges in every branch, if we are left with an edgeless graph, we have found a vertex cover of size at most $k$, otherwise there is none. A problem for which there is an algorithm with such kind of running time, that is, of the form $f(k)p(n)$ where $f$ is arbitrary function and $p$ is a polynomial, is called *fixed-parameter tractable* (FPT).

Another algorithmic technique, called kernelization or data reduction, tries to make the problem easier by simply reducing its size, instead of some clever organisation of the data. Let us illustrate it on the vertex cover problem also. First, notice that if there is a vertex $v$ of degree one, then we can clearly put into the cover the other vertex of the edge adjacent too $v$. Also, if there is a vertex $v$ of degree strictly greater than $k$ and we are looking for a vertex cover of size at most $k$, then we must always include $v$ in it, otherwise we would not be able to cover all the edges adjacent to $v$. The reduction consists of applying these two rules as long as possible. Now comes a crucial observation, that if the reduced graph has strictly more vertices than $k^2$, then it cannot have a vertex cover of size at most $k$ – every vertex has degree at least two so the graph has more than $k^2$ edges, but every vertex cover at most $k$ edges. Therefore, we only have to solve the vertex cover for a graph of size at most $k^2$, which can be done by brute force. The reduction rules can be applied in time $O(n+m)$ where $m$ is the number of edges, and the brute force search in time $O(2^{k^2}k)$. The kernelization idea is of particular importance for us, since our application in chapter 3 uses it. Also, the relation between FPT and kernelization is straightforward: a problem has an FPT algorithm iff there is an algorithm which reduces any instance of the problem to an instance of size bounded by a function of the parameter only, such that the reduction preserves solutions (allows recovery of the solution).

To show a problem which is not FPT unless P=NP, consider the following. A *$k$-colouring* of a graph $G$ is a mapping $c\colon \{1..k\} \to V$ such that for every edge $uv$ we have $c(u) \neq c(v)$. The problem is to determine whether a graph has a $k$-colouring. For $k \leq 2$ this problem is trivial, but already for $k = 3$ this problem is NP-hard, thus we cannot parametrise it by the number of colours.

This problem remains NP-hard even for planar graphs, thus trying restriction to such classes of graphs also does not help. However, planarity indeed helps in many cases. For instance, if we are deciding 4-colourability, there is a simple algorithm for planar graphs – one that always answers "yes", which is guaranteed

to be correct thanks to the 4-colour theorem [5]. For something less trivial, consider the problem of *dominating set*. A set $A$ of vertices is dominating if every vertex of $G$ is either in $A$ or adjacent to a vertex in $A$. When we restrict ourselves to planar graphs only, there is an algorithm for deciding the existence of a dominating set of size at most $k$ that runs in time $O(8^k \cdot n)$ [3]. This result is obtained by the method of bounded-depth search trees, but in a very intricate way – compared to the vertex cover example.

### 2.1.2 Parameterised Complexity Hierarchy

The problem of dominating set itself seems to be much harder than the vertex cover. When tackling some computational problem, the first step is usually to either find a polynomial algorithm (hence showing the problem is easy) or prove that it is NP-hard. Sadly, this is not always feasible, but luckily, parametrised complexity gives us a finer classification hierarchy. It consists of classes $W[1]$, $W[2], \ldots$, which can be defined by satisfiability problems of various restricted logics or as classes of problems solvable by some restricted boolean circuits. As we are only illustrating the theory, let us just give explicit description of the classes $W[1]$ and $W[2]$, which are the most commonly encountered ones anyway. The *weighted CNF-satisfiability* is a problem of deciding for a given formula $\phi$ in conjunctive normal form and an integer $k$ whether there is a satisfying assignment for $\phi$ such that at most $k$ variables are set to true. Analogously, *weighted 3-CNF-satisfiability* is restricted to formulas with clauses of size at most three. A parameterised problem $P$ is in $W[1]$ iff there exists a *parametrised reduction* from $P$ to weighted 3-CNF-satisfiability. Parametrised reduction works similarly as to reductions in the classical complexity theory, with the addition that the size of the parameter of the image depends only on the size of the parameter of the reduced instance.

For instance, the problem of finding a clique and the problem of finding an independent set are reducible to each other even in the parametrised sense – by taking the complement of the graph, independent set becomes a clique and vice versa, and their sizes do not change. On the other hand, although the vertex cover problem is reducible in the classical sense to the independent set—again by taking the complement of the graph—it is not reducible to it in the parametrised sense, since if the size of the cover is $k$, the size of the corresponding independent set is $n - k$, which does not depend on the value of $k$ only.

The $W[1]$-hard problems are defined as those to which weighted 3-CNF-satisfiability can be reduced in the parametrised sense, $W[1]$-complete problems are then those that are both $W[1]$ and $W[1]$-hard. The classes $W[2]$, $W[2]$-hard and $W[2]$-complete are defined similarly, but with weighted CNF-satisfiability in place. Notable examples of $W[1]$-complete problems are the problems of finding a clique (parameterised by the size of the clique) or deciding whether a non-deterministic single-tape Turing machine stops within $k$ steps. Notable examples of $W[2]$-complete problems are the dominating set problem or deciding whether a non-deterministic multi-tape Turing machine stops within $k$ steps. Problems that admit an FPT solutions are sometimes called $W[0]$ – as we have seen, vertex cover is one such problem. In general, proving that some problem, say $k$-dominating set, is in $W[2]$, is not as strong as proving that for some fixed $k$ it is NP-hard,

but it shows that it is unlikely to expect anything better than algorithm running in time $O(n^k)$ – this is the "finer classification" we mentioned before.

All the $W[i]$ classes are contained in the classes $W[P]$, for which there is a nice definition from [1] – a problem is in $W[P]$ if it can be solved by a non-deterministic machine which does at most $h(k)\log n$ non-deterministic step and at most $f(k)p(n)$ steps altogether, for some functions $f$ and $h$ and a polynomial $p$. The original definition [32] is also in terms of satisfiability problems. Reader familiar with more complexity theory can observe a similarity between Ladner's theorem, which says that there are infinitely many classes between P and NP (unless P=NP). This has actually been one of the motivations for the development of the whole parameterised complexity theory (see [31]). Also, the polynomial hierarchy (from complexity theory), defined in terms of quantifier alterations, has some relation to our world, see the logic-related chapters of [49].

The hardness of dominating set did not discourage mathematicians form working on it, quite the contrary, it stimulated a deeper investigation of the problem. Instead of parametrising just by the size of the solution, we can look more into the structure of the graph. Specifically, there is an FPT-algorithm for dominating set parameterised additionally by the genus of the graph, running in time $O((24g)^k n^2)$ [43], obtained again by the technique of bounded-depth search trees. This is not an exceptional case – for hard problems, we generally try to find as detailed and fine parameterisation as possible. Such approach brings us better understanding to where exactly lies the hardness of the problem, in other words, which parameter has the most influence on the complexity of solution and causes the "combinatorial explosion". Results of this kind are interesting from the theoretical point of view and can also explain why some heuristics perform better in practice.

Speaking of the genus, it brings us to another interesting topics. First of all, deciding the genus of a graph was shown to be NP-complete in 1989 [108] and later in 1999 [77] to be solvable when parameterised by the genus in linear time in the size of the graph. Genus and graphs on surfaces played an important role in the landmark Graph Minor Theorem by Robertson and Seymour [93], which says that every proper minor-closed graph class can be described by finitely many minor obstructions. One of the consequences actually partly inspired the development of the parametrised complexity theory itself. The problem of testing whether a graph $H$ is a minor of a graph $G$ is in general NP-complete (follows already from Karp [62] by considering for instance Hamiltonian path), but for a fixed $H$ can be solved in time $O(|V(G)|^3)$ [98]. This implied already in 1990 that deciding of the genus of the graph is FPT.

However, parameterised complexity differs from the graph minor theory in a profound detail. The results of graph minor theory are often either existential or very often involve large constants making them not too practical, whereas parameterised complexity has always held practical feasibility important [33]. Still, the importance of the graph minor theory is huge, as we will see in the next section. Before we get there, let us just mention general sources for parametrised complexity theory. The first monograph is from 1999 by Downey and Fellows [34], fathers of the theory, followed by a book by Niedermeier [87] which emphasises algorithms and applications and by a book by Flum and Grohe [49] which is oriented more towards structural results.

## 2.2 Structural Graph Theory

The purpose of this section is to introduce some notions commonly used in structural approach to graph theory. The word "structural" itself does not have any precise definition, but results and approaches in this part of graph theory have many ideas in common, which we try to exemplify here. One of the possible interpretations of the term is that we have a graph class $\mathcal{C}$ and try to find a simple description of every graph in $\mathcal{C}$ – say, we want to have a small set of "basic components" and rules how to glue them together according to some structure. A trivial example thereof is the classical theorem for 2-connected graphs – every such graph arises from a cycle by gluing paths to it.

Another recurring idea in this area is capturing of "structuredness" of graphs in some well-defined parameter. The simpler the structure, the lower the value of the parameter is. This is somehow shared with the parameterised complexity, but we should stress a slight difference – whereas in parametrised complexity, the parameter is very often (but definitely not always) the size of the solution, here the parameter is solely a property of the graph itself. This difference also reflects the emphasis of these two areas – in parametrised complexity, we are more interested in the nature of the computational problem; in structural graph theory, the graphs themselves are the point of interest.

### 2.2.1 Treewidth

No other parameter deserves a prominent place more than the ubiquitous treewidth. A *tree-decomposition* of a graph $G = (V, E)$ is a tree $T = (B, F)$ where to each $b \in B$ we associate a subset of vertices of $G$, such that

- $\bigcup_{b \in B} b = V$, that is, every vertex is contained in some $b$,
- $\forall \{u, v\} \in E \; \exists b \in B \; \{u, v\} \subseteq b$, that is, every edge is contained in some $b$,
- $\forall v \in V$ the set $\{b \colon v \in b\}$ induces a connected subgraph of $T$.

The vertices of $T$ are usually called bags. The *width* of a decomposition is defined as $\max_{b \in B} |b| - 1$. The *treewidth* of a graph $G$ is the minimum width of a decomposition of $G$. For instance, the trees have treewidth one (which is the explanation for the $-1$ in the definition). The treewidth of a graph intuitively measures how close a graph is to a tree, in other words, how "thick" its tree-like structure is.

This definition of treewidth comes from the Graph Minor Theorem area [94]. However, other equivalent definitions have appeared even before, for instance one as subgraphs of $k$-trees. A $k$-tree is either a clique on $k$ vertices, or arises from a smaller $k$-tree by a adding a new vertex and joining it to a clique of $k-1$ vertices. Clearly, a graph has treewidth at most $k$ iff it is a subgraph of a $(k-1)$-tree. This definition comes from [10], and has had algorithmic/decompositional applications in quite early time, see [7]. We mention that the definition of $k$-tree was inspired by that of simplex, a notion from topology. In the whole Graph Minor Theory, topology of graphs and their embedding properties play an important role.

### Computing tree-width

One can either compute treewidth exactly, which is NP-complete [8] but FPT with respect to the actual value of treewidth [12]. However, it has been shown that this FPT algorithm (which is even linear) does not perform very well practice [101]. On the other hand, there are many heuristics that do not give exact value of treewidth but usually quite good approximation and perform well in practice; these heuristics exploit the variety of equivalent definition of treewidth [15]. For instance, one can construct tree-decomposition recursively by the *min-degree rule*, which selects vertex $v$ of smallest degree, puts $v$ in a bag with all its neighbours, removes $v$ from the graph and adds an edge between every two non-adjacent neighbours of $v$. Also, there are algorithms that aim on specific small values of $k$, or algorithms that work with specific graph classes.

Furthermore, it is interesting to obtain for a given graph $G$ lower bounds on treewidth. A simple one comes from degeneracy – treewidth is at least as big as the degeneracy of $G$. However, this bound is sometimes horribly bad, for example for planar graphs – their degeneracy is at most 5, but treewidth can be arbitrarily big (e.g., grid). For planar and similar graphs the notion of *brambles* is useful for obtaining a better bound. A bramble is a collection of subsets of vertices, such that each two subsets either share a vertex or there is an edge between two vertices, one from each set. The *order* of a bramble is the size of a smallest set $S$ such that $S$ intersects every set in the bramble. Brambles are, in a sense, a dual concept to treewidth, as the maximum order of a bramble is equal to treewidth plus one [102]. There are some heuristics that construct brambles of large order, thus lowerbound treewidth [16]. For more on computing treewidth, see [14, 13].

### Applying treewidth

First of all, it is important to say that graphs with bounded treewidth do appear in practice – probabilistic networks, chemical compounds, control-flow graphs of programs all tend to have small treewidth (see [13]). Graph classes with small tree-width are for instance outerplanar graphs (planar graphs that can be drawn in plane such that every vertex lies in the outerface) or series-parallel graphs (a class of graphs that arise by gluing $K_2$ graphs according to two rules of parallel composition and series composition). Planar graphs do not have bounded tree-width, but something can be salvaged – the tree-width of a planar graph is bounded by a function of its diameter, which makes them a class with local-bounded treewidth, a notion we will discuss more in 2.4.

Treewidth has many applications, both outside graph theory – e.g., VLSI layouts, matrix factorisations, evolution theory (see [11]); and in graph theory itself. Many NP-complete or even PSPACE- and P#-complete problems become polynomial on graphs with bounded treewidth. Examples of these include Hamiltonian path, Dominating set, Vertex cover, Chromatic number. But it is not the list of individual problems that is the most striking, much stronger result is a theorem by Courcelle [21], that states that for every problem from a very general class there exists an FPT algorithm with respect to the length of the formula and treewidth to decide the formula. The class of such problems includes for instance those mentioned above. This theorem is an example of *meta-theorem*, that is, a theorem that gives "infinitely many" algorithms. We will discuss this matter

more in 2.4.

**Beyond Treewidth**

Treewidth is not the only parameter of its kind. *Pathwidth* is defined as treewidth with the exception that the decomposition graph is a path, not an arbitrary tree. This parameter also rose during the proof of the Graph Minor Theorem in [91], as a tool to deal with forest minors. *Cliquewidth*, appearing first in [24], is defined by a sequence of gluing recursive operations, the value of the parameter depends on the complexity of operations (number of auxiliary labels they require). Its name stems from the fact that complete graphs have bounded cliquewidth, so, unlike treewidth, there are classes with high edge density that have bounded cliquewidth. For the other direction, treewidth at most $k$ implies cliquewidth at most $2^k$. Analogues of Courcelle's theorem [21] hold for cliquewidth [22, 23], although slightly weaker. As for some sad news, FPT algorithm for computing cliquewidth is not known. Also, Hamiltonian cycle or Chromatic number are $W[1]$-hard with respect to cliquewidth [50].

Another interesting parameter is *neighbourhood diversity*. It has been defined in [71] while describing classes for which a faster version of Courcelle's theorem holds, which is partly related to our result in Chapter 3. Graph has neighbourhood diversity at most $k$ if its vertices can be partitioned into $k$ groups such that the neighbourhood of a vertex depend only on the group the vertex is in. This parameter is incomparable to treewidth (consider a path and a clique), but cliquewidth is at most neighbourhood diversity +1. For more information about this parameter, see [65].

Important property of treewidth is the separation of vertices. This has inspired the definition of *branchwidth* [97], which instead of vertices separates edges. A decomposition for this parameter is a cubic tree $T$ where the leaves are labelled by edges of the decomposed graph $G$. Each non-leaf vertex $t$ of $T$ partitions the edges of $G$ in two sets, the number of vertices of $G$ that is common to both sets is the width $t$. The width of the decomposition is then the maximum over all $t$, the branchwidth of the graph is minimum width over all decompositions. In terms of value it is essentially the same as treewidth – $b \leq t + 1 \leq \lceil \frac{3}{2} b \rceil$. However, its edge-aimed definition makes it easy to generalise into hyper graphs, which was important for the Graph Minor Theorem. Also, branchwidth can be defined for other structures for which some kind of connectivity makes sense—for instance, matroids.

For more information about these and other width parameters, see the survey [59].

### 2.2.2 Structural Theorems

This part deals with results of more theoretical kind. Namely, we will discuss the already mentioned Graph Minor Theorem in some detail and related results.

The oldest investigated graph class are definitely the planar graphs. The most famous problem related to them, the question whether for every planar graph we can colour its vertices by 4 colours such that no two adjacent have the same colour (that is, *properly coloured*), has been posed already in 1852. Its solution came in 1977 [5] and was based on the technique of discharging – basically, one

argues that every big graph can be reduced to a smaller one such that a solution can then be extended, and then a number of minimal configurations with respect to the reduction rules is analysed by hand. The proof of this theorem has been revolutionary since it has used computer for analysing the minimal configurations. Even today, no short proof of this theorem is known (although there is a simpler proof [90]).

This problem gave rise to the branch of graph theory called graph colouring. Chromatic number of graph $G$ is the minimal number $c$ such that vertices of $G$ can be properly coloured with $c$ colours. The question is usually of the form: Given a class of graphs, what is an upper bound on chromatic number of graphs in this class? It has been solved for graphs embeddable on a given surface – a tight upper bound was given already in 1890 [58], with the exception of the plane; and the answer is given by the Heawood formula:

$$\left\lfloor \frac{7+\sqrt{49-24\chi}}{2} \right\rfloor,$$

however, the bound has been shown to be tight much later – the last case has been solved in 1968 (see [89]).

There is another important theorem tied to planar graphs, Kuratowski's theorem [70]. It says that graph is planar if and only if it does not contain $K_5$ or $K_{3,3}$ as a minor. This is a recurring pattern in structural graph theory – characterising a graph class by forbidden substructures. Can this theorem be also extended to other surfaces? The situation is quite funny – although we know the list of forbidden minors explicitly only for the projective plane (and there are 35 forbidden minors, see [78]), the answer is yes. It has been proven for non-orientable case first [6] and later in general [96].

### Graph Minor Theorem

However, this kind of characterisation can be extended even further, namely to proper minor-closed classes. A class $\mathcal{C}$ is *minor-closed* whenever for every $G \in \mathcal{C}$ and for every $H$, if $H$ is a minor of $G$ then $H \in \mathcal{C}$. Furthermore, class is *proper* if it does not contain every graph. The already mentioned Graph Minor Theorem [93], whose proof spans over 20 papers, then says that every proper minor-closed class can be characterised by a finite list of forbidden minors. Crucial is the word finite – the existence of an infinite list is trivial, because one can just take all graphs not in the given class. Alternatively, the theorem can be spelled in terms of orderings: there are no infinite antichains for the minor relation on graphs, that is, the class of graphs is well-quasi-ordered by the minor relation.

The importance of this theorem is illustrated by the fact that a great number of graph properties are inherited by minors – for instance, being cycle-free, embedabbility to a fixed surface or other topological properties. Also, all the mentioned width parameters are preserved by taking minors. Another example is the property of being series-parallel. It has been proved already in 1952 [30] that series-parallel graphs are characterised by not having a $K_4$-minor. Similarly, forests have no $K_3$-minor, and 4-colourable graphs have no $K_5$-minor (this is proven using the Four Colour Theorem).

The Graph Minor Theorem, originally called *Wagner's conjecture*, has been possibly inspired by a simpler case saying that the class of all rooted forests is

well-quasi-ordered by the minor relation, proven in 1960 [69, 106]. The proof is not that hard – one assumes the converse statement and constructs a "minimal" infinite antichain. Then, using the fact that every rooted tree can be decomposed into two trees by cutting one of the edges between the root and its first son, a smaller antichain is obtained, which is a contradiction.

The proof of the Graph Minor Theorem itself is much harder, although it also starts by taking a minimal infinite antichain. However, no decomposition as for trees is at hand, which means we have to work harder. Let $G_1$ be the first graph of the sequence, then we know that the remaining graphs have no $G_1$-minor and thus have special structure.

First consider the case that $G_1$ is planar. Here is a point when treewidth comes into play – in [95] it is proved that for every planar $H$ the class of graphs that do not have $H$ as a minor has bounded treewidth, and that for every class $\mathcal{C}$ with bounded treewidth there is a planar graph $H$ such that $\mathcal{C}$ is $H$-free (that is, no graph in $\mathcal{C}$ has a minor isomorphic to $H$). The proof of the theorem uses dual characterisation of treewidth – graphs with large treewidth have big grid minors, and vice versa. Thus, our $G_1$-free sequence has bounded treewidth. Then the original proof for forests can be modified by working with the decompositions instead.

When $G_1$ is not planar, the situation is more complicated, and one has to invoke the so-called Structure Theorem from [99]. This theorem basically says that $G_1$-free graphs have "two-dimensional" component for which certain modification of the tree-approach works, and "approximately embeddable" component, for which topological methods are used. For more surveying of this topic, see the survey [74].

## 2.3 Bounded Expansion and Nowhere-denseness

In this section we will introduce the concepts of bounded expansion and nowhere-dense classes. As these concepts are fundamental to our results in Chapters 3 and 4, we will investigate them more formally and thoroughly. The whole theory has been developed mainly by Nešetřil and Ossona de Mendez, who have recently authored a corresponding monograph [85] to which we refer the reader for this whole section.

### 2.3.1 Treedepth

Treedepth is yet another parameter to measure structuredness of a graph. The name comes from the paper [81], where it has been defined as follows. Let $T$ be a rooted tree, we define the partial order on $T$ by putting $a \leq b$ when the vertex $a$ lies on the path from $b$ to the root of $T$. The *depth* of a tree is the length of a longest path from a leaf to the root. The closure $\mathrm{clos}(T)$ of $T$ is a graph obtained by adding all edges $(a, b)$ whenever $a \leq b$. A *treedepth decomposition* of $G$ is a rooted tree $T$ on the vertex set $V(G)$ such that $E(G) \subseteq E(\mathrm{clos}(T))$. The *treedepth* $\mathrm{td}(G)$ of $G$ is the minimum of depth over all treedepth decompositions of $G$.

There is an equivalent, recursive, definition, called *elimination tree*:

$$\mathrm{td}(G) = \begin{cases} 1 \text{ when } G \text{ is a single vertex,} \\ \min_{v \in V(G)} \mathrm{td}(G - \{v\}) \text{ when } G \text{ is connected,} \\ \max \mathrm{td}(C) \text{ where } C \text{ runs over connected components of } G. \end{cases}$$

We also immediately see that when we run a depth-first search algorithm on a graph, we obtain a treedepth decomposition. As the depth of a depth-first search tree cannot be greater than the length of a longest path in $G$, and treedepth of a path of length $l$ is $\log_2(l)$, this procedure yields a decomposition of depth at most $O(2^{\mathrm{td}(G)})$. In 2.4, we will see that computing treedepth exactly is FPT when parametrised by treedepth. This is best possible, as computing treedepth is NP-complete.

One can immediately observe that $\mathrm{td}(G) \geq \mathrm{tw}(G)+1$, because taking all root-leaf paths as bags yields a treewidth decomposition, even a pathwidth decomposition. For the converse, there is an inequality $\mathrm{td}(G) \leq (\mathrm{tw}(G)+1)\log_2(3|V(G)|)$ – first, for every tree $T$ on $n$ vertices we have $\mathrm{td}(T) \leq \log_2(n)$ by using the recursive definition and always taking the centre of the graph as a root. Second, replacing every vertex in $T$ by a clique of size $\mathrm{tw}(G)+1$ increases the depth at most $(\mathrm{tw}(G)+1)$-times. In connection with [15], this yields a $\log^2(|V(G)|)$-approximation algorithm for treedepth. One can say that treewidth measures similarity to a tree, and treedepth measures similarity to a star. From below, we see immediately that treedepth of a graph is bounded by the size of its minimum vertex cover.

Let us now illustrate how treedepth made a name for itself in the context in structural graph theory. The graph colouring problem asks to colour the graph such that globally only few colours are used. A possible way to generalise this problem is to impose an additional condition that we want to use locally many colours – such colouring obviously gives more information. Instances of this include acyclic chromatic number – there is no subgraph isomorphic to a cycle which gets only two colours, or star chromatic number – no path of length 3 gets two colours. In general, we can ask for $\chi(f, G)$, the minimum number of colours needed to properly colour $G$ such that for every graph $H$ and every subgraph $H'$ of $G$ isomorphic to $H$ we have that $H'$ gets at least $f(H)$ colours. For acyclic chromatic number, $f$ assigns 3 to every cycle and is 0 otherwise, analogously with path of length 3 for star chromatic number.

If we put $f(H) = g_k(H) = \min(k, \mathrm{tw}(H)) + 1$, we get to an important result dubbed *low treewidth colouring* [27]. It says that for every proper minor-closed graph class the numbers $\chi(g_k, G)$ are bounded (the bound depends on $k$ and the class itself). In other words, for every graph $K$ and every $j \in \mathbb{N}$ there exists $k \in \mathbb{N}$ such that every $K$-free $G$ can be partitioned into $k$ graphs such that any $j' \leq j$ parts induce graph with treewidth at most $j' - 1$. This result, originally conjectured by Thomas in 1995 and motivated by a weaker result that held for surfaces only [29], tells us that although minor-closed classes can have unbounded treewidth, there is still a lot of "bounded structure". The proof uses tools from the proof of the Graph Minor Theorem.

The paper [81] investigates how many colours can we require locally, in other words, how big can the function $f$ from the definition of $\chi(f, G)$ be. Let $H$ be a graph and $\mathcal{C}$ a proper minor-closed graph class, the *upper chromatic number* $\bar{\chi}(H)$

is the greatest integer such that there exists a constant $c = c(\mathcal{C}, H)$ such that for any graph $G \in \mathcal{C}$ there exist a proper colouring of $G$ with $c$ colours such that any subgraph of $G$ isomorphic to $H$ gets at least $\bar{\chi}(H)$ colours. That is, $\bar{\chi}(H)$ is the largest possible value of $f(H)$ so that $\chi(f, G)$ are bounded for every proper minor-closed class. The surprising crucial result of [81] is that $\bar{\chi}(H) = \mathrm{td}(H)$. As treedepth upperbounds treewidth up to $-1$, this results generalises the previous one, and we use the name *low treedepth colouring* for it.

To illustrate how colouring and bounded-depth trees get along, we just show another equivalent definition of treedepth, that of *centred colouring* – it is a colouring of a graph such that for every its connected subgraph there exist a Colo which appears exactly once. The minimum number of colours required for a centred colouring is precisely the treedepth of the graph – when constructing the colouring from the decomposition, remove all roots of the forests and assign a new colour to them, when constructing the decomposition from the colouring, pick a connected component and select the vertex with the unique colour to be its root.

An interesting application of the result of [81] is a bound on chromatic numbers of exact powers. The $p$-th exact power $G^p$ of a graph $G$ is the graph on the same vertex set and edge $(x, y)$ iff there exists a path of length $p$ in $G$ between $x$ and $y$. For proper minor-closed class $\mathcal{C}$, chromatic numbers of even exact powers of graphs from $\mathcal{C}$ are clearly unbounded – consider stars. However, for every odd $p$ there exists an integer $k$ such that for every $G \in \mathcal{C}$ with odd-girth at least $p + 1$ the chromatic number of $G^p$ is at most $k$. Another corollaries are from the area of restricted dualities, which we do not deal with in this thesis.

To show treedepth in the light of forbidden substructures, we mention the paper [36] which presents minimal minor/subgraph/induced subgraph obstructions for the class of all graphs of tree-depth at most $t$. From the Graph Minor Theorem it follows that the number of such obstructions is finite, but in this case more has been revealed – there is an upper bound $2^{2^{t-1}}$ on the size of forbidden graphs, and there are exactly $\frac{1}{2}2^{2^{t-1}-t}(1 + 2^{2^{t-1}-t})$ of obstructions (for all three relations) that are acyclic. The paper also presents simple structural lemma for constructing bigger obstructions from smaller ones, and gives precise list of obstructions for $t \leq 3$.

### 2.3.2 Bounded expansion

In the previous subsection, we looked at the function $\chi(f, G)$ from the point of view of $f$, that is, how big can the function be for individual $H$. In this subsection, we look at it from the point of view of $G$, in other words, can we extend the result on low treewidth colouring beyond minor-closed classes? The answer is yes, more precisely, the answer is classes of graphs with bounded expansion. The notion of bounded expansion appeared in [82] and as in the case of treedepth, there are many equivalent definitions.

Let us introduce the ideas behind the definitions before we state them formally. If the maximum average degree (mad) of graphs in a class is bounded, then the chromatic number of graphs in that class is also bounded. However, even more is true – let $\mathcal{C}$ be a class of graphs, and $\mathcal{C}'$ the class $\{G' \colon G'$ arises from a $G \in \mathcal{C}$ by contracting star forests$\}$. If the mad of graphs in $\mathcal{C}'$ is bounded, then the star

chromatic number of $\mathcal{C}$ is also bounded [79]. For the other direction, if the star chromatic number of $\mathcal{C}$ is bounded by $N$, then the mad of $\mathcal{C}$ is also bounded – for every two colours $i, j \leq N$, orient the edges in the star forest induced by the colours $i, j$ towards the roots, then every vertex in the original graph has indegree at most $\binom{N}{2}$.

Can this be generalised? Let $\chi_p(G)$ be the minimum number of colours such that there is a colouring of $G$ for which for every $i \leq p$ each $i$ colour classes induce a graph with treedepth at most $i$. Note that $\chi_1$ is the usual chromatic number and $\chi_2$ is the star chromatic number. We say that $H$ is a *$d$-depth minor* of $G$, if $H$ arises from $G$ by contracting connected subgraphs of radius at most $d$ and omitting some edges/vertices. The question is then whether there exist functions $f_1$ and $f_2$, such that for every $p$ and every class $\mathcal{C}$:

- if the minors of depth at most $f_1(p)$ of $\mathcal{C}$ have bounded mad, then the graphs in $\mathcal{C}$ have bounded $\chi_p$,
- if the graphs in $\mathcal{C}$ have bounded $\chi_{f_2(p)}$, then the class of all minors at depth $p$ has bounded mad.

Such functions do exist, and, in way, define the bounded expansion classes.

Another way to discover such classes is to ask "how to define sparseness"? Clearly, if mad of a class is not bounded, then the class cannot be sparse. However, even bounded mad does not guarantee sparseness – consider a class arising from the class of all cliques by subdividing every edge once. Such class has mad $= 2$, but still retains many properties of cliques and thus should not be called sparse. So, we want to forbid this (in a mild manner), and we arrive at the definition of *greatest reduced average density* – $\nabla$. Let $G$ be a graph, $\nabla_r(G)$ is $\max \frac{E(H)}{V(H)}$ where the maximum is over all $r$-depth minors $H$ of $G$. A class $\mathcal{C}$ has *bounded expansion* if there exists a function $f$ (called *expansion function*) such that $\nabla_r(G) \leq f(r)$ for every $G \in \mathcal{C}$. In other words, the mad after contracting $r$-depth minors is bounded, but the bound does not have to be uniform. Let us also explicitly state that for a bounded expansion class the average degree of a graph $G$ is at most $2\nabla_0(G)$; hence, graphs in any class of graphs with expansion bounded by $f$ have average degree bounded by a constant $2f(0)$. Similarly, we conclude that every $G \in \mathcal{C}$ has an orientation (even acyclic one) with in-degree at most $2f(0)$.

Bounded expansion classes properly generalise proper minor-closed classes – for which the expansion function is constant, which follows from the results [64, 107]. For instance, bounded degree classes have bounded expansion—the expansion function is an exponential one—but do not form a proper minor-closed class, as the minor closure of the class of all cubic graphs is the class of all graphs. Another interesting example is the class of graphs that can be drawn in the plane (or any other fixed surface) such that every edge crosses at most $k$ other edges.

In the paper [82], this notion of bounded expansion is investigated with respect to low treedepth colourings. First of all, bounded expansion classes have low treedepth colouring, which generalises the previous result for proper minor-closed classes, but surprisingly even more is true – they precisely characterise it. That is, a class has low treedepth colouring if and only if it has bounded expansion. In order to prove the result, authors investigated two notions which are of independent interest. First is *stability with respect to lexicographic product*

– the graph $G \bullet K_c$ is obtained from $G$ by replacing every vertex by a copy of $K_c$, and for every two cliques whose original vertices were adjacent we put all edges between them. In other words, $G \bullet K_c$ is a "$c$-thickening" of $G$. The bounded expansion classes are stable – that is, for fixed $c$ the class $\{G \bullet K_c \colon G \in \mathcal{C}\}$ has also bounded expansion, albeit the expansion function for it is bigger than that for $f$. This implies that the bounded-crossings class from previous paragraphs has bounded expansion – the lexicographic product of planar graphs with $K_k$, where $k$ is the bound on the crossing number, contains the bounded-crossings class as topological minors of depth 2. We should also note that minor-closed classes are not stable with respect to this product – every graph is a minor of the product of some planar graph and $K_2$.

Second and for our later results more important notion is that of fraternal augmentations. Let $\overrightarrow{G}$ be a directed graph, then a *1-transitive fraternal augmentation* of $\overrightarrow{G}$ is a directed graph $\overrightarrow{H}$ such that whenever $(x, y)$ and $(y, z)$ are arcs of $\overrightarrow{G}$ then $(x, z)$ is an arc of $\overrightarrow{H}$ (transitive edges) and whenever $(x, y)$ and $(z, y)$ are arcs of $\overrightarrow{G}$ then either $(x, z)$ or $(z, x)$ is an arc of $\overrightarrow{H}$ (fraternal edges). Furthermore, we require $\overrightarrow{H}$ to retain all edges of $\overrightarrow{G}$. The augmentation is *tight*, if it is edge-minimal among all augmentations, that is, no extra edges were added. *Transitive fraternal augmentation* of $\overrightarrow{G}$ is a sequence $\overrightarrow{G} = \overrightarrow{G}_1 \subseteq \overrightarrow{G}_2 \subseteq \ldots \subseteq \overrightarrow{G}_i \subseteq \ldots$. The bounded expansion classes are also stable with respect to augmentations, that is, the class $\{G' \colon G'$ is the underlying undirected graph of $\overrightarrow{G}'$ which is a tight 1-transitive fraternal augmentation of $\overrightarrow{G}$, where $\overrightarrow{G}$ is an orientation with maximum indegree $k$ of a graph $G$ in $\mathcal{C}\}$ has bounded expansion. The iterative version says that there exists a function $f$ such that every graph $G \in \mathcal{C}$ has a transitive fraternal augmentation $\overrightarrow{G}_1, \overrightarrow{G}_2, \ldots$ where maximum indegree of $\overrightarrow{G}_i$ is at most $f(i)$. The fraternal augmentations played an important role in the proof of existence of low treedepth colouring for bounded expansion classes. Intuitively, the edges that arise from augmentations help to find the tree whose closure contains the original graph. We will use the augmentations similarly in our result in Chapter 4.

The proof of the original result with low treewidth colouring [27] relied on the Structure Theorem and thus did not yield a practical algorithm. However, in [80] a simple algorithm for finding a low treedepth colouring is shown, and when restricted on a fixed class with bounded expansion, its running time is linear in $|V(G)|$. For fixed $k$, $\overrightarrow{G}_k$ from the transitive fraternal augmentation of $G$ can also be computed in linear time, when $G$ is from a class with bounded expansion. The existence and algorithmical constructibility of low treedepth colouring has an important corollary in subgraph testing and deciding general first-order properties for bounded expansion classes. We postpone the discussion to 2.4.1.

Let us summarise the many-facetedness of bounded expansion. For a class of graphs $\mathcal{C}$, the following conditions are equivalent:

- $\mathcal{C}$ has bounded expansion,
- $\mathcal{C}$ has low treedepth colourings,
- $\mathcal{C}$ has low treewidth colourings,
- for every $c$, the class $\mathcal{C} \bullet K_c$ has bounded expansion,

- for every $k$ the class of 1-transitive fraternal augmentations of directed graphs $\overrightarrow{G}$ with maximum indegree at most $k$ and $G \in \mathcal{C}$ has bounded expansion,
- there exists a function $f$ such that every graph $G \in \mathcal{C}$ has a transitive fraternal augmentation $\overrightarrow{G}_1, \overrightarrow{G}_2, \ldots$ where maximum indegree of $\overrightarrow{G}_i$ is at most $f(i)$.

We also mention that although we have defined the grad and expansion function in terms of minors, the same results can be obtained when using topological minors. Specifically, class has bounded expansion with respect to minors iff it has bounded expansion with respect to topological minors; the proof of that amounts to showing that the greatest reduced average density and its topological analogue are polynomially related. This is quite interesting, as in generals topological minors differ profoundly – for instance, the Hajos conjecture turned out to be false for almost all graphs, whereas Hadwiger conjecture is true for almost every graph. Also, analogue of the Graph Minor Theorem for topological minor order is not true.

### 2.3.3 Nowhere dense

In this section we describe the concept of nowhere dense classes, which attain even higher level of generality than bounded expansion classes. Denote by $\mathcal{C}\nabla i$ the class $\{G\colon G \text{ is an } i\text{-depth minor of some graph in } \mathcal{C}\}$. Class $\mathcal{C}$ is called *nowhere dense* if $\sup_{G\in\mathcal{C}\nabla i} \omega(G)$ is finite for every $i$, in other words, no $\mathcal{C}\nabla i$ is the class of all finite graphs. Compare this with the definition of bounded expansion classes, where instead of $\omega(G)$ we were interested in the density. Bounded expansion classes can also be defined as classes for which every $\mathcal{C}\nabla i$ has bounded chromatic number. This shows us instantly that nowhere dense classes properly generalise bounded expansion classes – an example of a class which is nowhere dense but has not bounded expansion is a class of graphs with unbounded chromatic number and no triangles [4].

The origin of nowhere dense classes lies in (finite) model theory. Model theory is a branch of logic which deals with general relation structures and formulas, and finite model theory restricts itself to finite structures only. It is a virtue of this heritage that most of the result we obtain for graphs can be transfered into general relation structures – and this is interesting for other branches as Database theory or Constraint-satisfaction programming since in these areas one works with general relational structures also. Very often, finite model theory investigates which results from general model theory hold when restricted to finite and which do not. One of such question was that of homomorphism preservation, that is, whether a validity of a formula is preserved under homomorphism of the structure. Nowhere dense classes arose as characterisation of classes for which results of such kind hold. For more on this connection, see [83].

There is another characterisation of nowhere dense classes, in terms of asymptotic edge density. If one aims at a definition of sparseness and denseness, it should be complete, that is, every class should be either sparse or dense (and ideally, not both). It turns out that our definitions allow such theorem [84], that is, for every

class one of four possibilities happens:

$$\lim_{r\to\infty} \limsup_{G\in\mathcal{C}\nabla r} \frac{\log|E(G)|}{\log|V(G)|} \in \begin{cases} -\infty & \text{(graphs with no edges)},\\ 0 & \text{(graphs with at most } k \text{ edges)},\\ 1 & \text{(nowhere dense classes)},\\ 2 & \text{(somewhere dense classes)}. \end{cases}$$

The *somewhere dense classes* are exactly those such that there exists an $r_0$ for which $\mathcal{C}\nabla r_0$ is the class of all graphs. We again note that this theorem can be stated in terms of topological minors with the same results.

In a sense, the nowhere dense classes are a quantitative generalisation of bounded expansion classes. Let $n = |V(G)|$, the bounded expansion classes consists of graphs with $O(n)$ edges where and average degree bounded by constant, whereas nowhere dense classes have $n^{1+o(1)}$ edges, that is, for any nowhere-dense class $\mathcal{C}$ and for every $\varepsilon > 0$ there exists a function $g(n) = O(n^\varepsilon)$ such that every graph $G \in \mathcal{C}$ has average degree at most $g(n)$. Algorithms for bounded expansion classes that run in linear time lead to $n^{1+o(1)}$ (that is, almost linear) algorithms for nowhere dense classes. We note that graphs with $n^{1+\varepsilon}$ have some typical properties of random graphs, see for instance the book [4]. This fits into the "randomness vs. structure" paradigm, see [105].

## 2.4 Algorithmic Model Theory

In this part we explore the area of *algorithmic metatheorems* for sparse graph classes. Metatheorems usually say that some families of problems can be solved efficiently on structured graphs classes. The prefix meta- emphases that we give a family of algorithms at once, instead of the usual ratio one algorithm per theorem. The advantage of such approach is also that the obtained algorithms have usually very compact and uniform description.

It turns out that for such kind of results, the language of logic and model theory is the most fitting one, and as knowledge of these is not so widespread among our expected readers, graph theorist, we provide short introduction to notation and results. The chief references for finite model theory are the books [73], aimed more on computer scientists, and [41], which aims more on mathematicians. For introduction to logic, see for instance the book [40], for general model theory, see [60]. For algorithmic metatheorems there are recent surveys [56, 66, 57].

Let us first define general *relational structure*. A *vocabulary* is a collection $\sigma$ of constant symbols $c_1, c_2, \ldots$ and relational symbols $R_1, R_2, \ldots$; each symbol has associated arity. A $\sigma$-structure $\mathcal{A} = (A, \{c_i^A\}, \{R_i^A\})$ consists of universe $A$ together with interpretation of each constant $c_i$ as an element $c_i^A \in A$ and each $k$-ary symbol $R_i$ as a $k$-ary relation $R_i^A \subseteq A^k$. For example, graphs have vocabulary consisting only of the relation for adjacency, a structure is then a set of vertices along with tuples-edges. There are some ways how to turn relational structures into graphs. The first is *Gaifman graph*, where the vertices are elements of the universe, and $(a, b)$ is an edge whenever there exists a $k$-ary relation $R$ and a tuple $(x_1, \ldots, x_k) \in R$ such that $a$ and $b$ are in $(x_1, \ldots, x_k)$. More convenient is *incidence graph*, which is a bipartite graph $(A, B)$ where $A$ are the elements of the universe and $B$ are the tuples in relations, and edge is between a vertex and

a tuple if that vertex is in the tuple. Further ways include e.g. *star selectors* [83]. For a class $\mathcal{C}$ of structures, we say that $\mathcal{C}$ has bounded expansion if the corresponding class of Gaifman graphs has bounded expansion (it turns out that it happens if and only if the corresponding class of incidence graphs has bounded expansion – we prove this in 4.4). Similarly we define other notions like nowhere denseness for classes of structures.

Now we define terms and formulae in certain languages. Let $\sigma$ be a vocabulary and $x_1, \ldots$ a countably infinite set of variables.

- Each constant $c_i$ is a term.
- Each variable $x_i$ is a term.
- If $t_1, t_2$ are terms, then $t_1 = t_2$ is an (atomic) formula.
- If $t_1, \ldots, t_k$ are terms and $R_i$ is a $k$-ary relational symbol, then $R_i(t_1, \ldots, t_k)$ is an (atomic) formula).
- If $\phi_1, \phi_2$ are formulae, then $\phi_1 \wedge \phi_2,\ \phi_1 \vee \phi_2,\ \neg\phi_1, \forall\phi_1,\ \exists\phi_1$ are formulae.

Every such formula is called first order ($\mathbb{FO}$) formula. If we forbid $\forall$ quantification, we obtain a fragment – the existential first order logic ($\exists\mathbb{FO}$). Furthermore, if we add variables $X_1, \ldots$ for sets of elements of the universe, we obtain monadic second order logic ($\mathbb{MSO}$). This logic is further classified into $\mathbb{MSO}_1$ and $\mathbb{MSO}_2$, the first one is over the vocabulary of graphs as defined in the previous paragraphs, the second one contains in the universe both vertices and edges, and has the relational symbol for vertex-edge incidence. Monadic second order logic is itself a fragment of second order logic, where the set quantifiers are not restricted to sets of elements only, but range over relations of arbitrary (finite) arity. Furthermore, existential second order logic holds a connection to computational complexity, embodied in the theorem of Fagin [47] which says that existential second order logic corresponds precisely to the class NP. This theorem led to the development of the area of descriptive complexity [61].

Complexity of a formula is often measured by its *quantifier rank* (qrank), atomic formulas have qrank zero, and formula of the form $Q\phi$ where $Q$ is a quantification over some variable is one higher than the qrank of $\phi$. For conjunction or disjunction, the qrank is defined as the maximum of qrank of conjuncts or disjuncts, respectively. Similarly, we define *quantifier alteration* (qalt), which counts only alteration between $\exists$ and $\forall$ quantifiers. For formulae we define validity – atomic formula is true if the corresponding terms are equal (for the $t_1 = t_2$ case) or the tuple is in the relation; non-atomic formulae are evaluated recursively.

It is natural to ask what can be expressed in various logics. Proving that something can be expressed amounts simply to giving a corresponding formula, proving inexpressibility is usually harder. A classical tool for such result is *Ehrenfeucht-Fraisse game*. It is a game on two structures $A, B$ with players spoiler and duplicator. The spoiler picks elements from the structures and tries to show that the structures are not isomorphic, the duplicator tries to respond by picking elements such that a partial isomorphism is maintained between the picked parts of the structures. If the duplicator has a winning strategy, it tells us that the structures are in a certain sense not distinguishable, namely, it might happen that one has

a given property and the other does not, which shows an inexpressibility result. For instance, this can be used to show that Hamiltonicity is not expressible in $\mathbb{MSO}_1$ (but is clearly expressible in $\mathbb{MSO}_2$). For more details and examples see [73].

Now we move to a definition we will use in Chapter 3 for our result. Two graphs are said to be $n$-equivalent, if they satisfy the same first order formulas with at most $n$ quantifier alterations. This concept of $n$-equivalence is of practical use for evaluating formulae on graphs. It serves to reduce the investigated graph to a small one, so that time-expensive approaches as brute force become possible – recall kernelization from 2.1.1. An example of such application is the following theorem (from section 6.7 of [85]): for every $D, n$ exists $N$ such that every graph $G$ with $\mathrm{td}(G) \leq D$ is $n$-equivalent to one of its induced subgraphs of order at most $N$. This can be extended even to labelled graphs. In our work we use a similar theorem, taken from [53]. Informally, the result says that when one is interested in checking whether a specific formula is true on a class of trees of bounded depth, then one can also assume bounded degree.

Let us also mention that there exists a notion dual to $n$-equivalence for graphs, called rank-$n$ type of a formula. For a structure $A$, a type is a set of all qrank $n$ sentences (i.e., formulae without free variables) that hold in $A$. It turns out that, up to equivalence of formulae, there are only finitely many formulae in a rank-$n$ type, and that there are only finitely many rank-$n$ types. Many results which aim on general formulae deciding, such as metatheorems, work by (brute-force) computing the type of a structure.

### 2.4.1 Model Checking Problem

The standard problem in the finite model theory is the *model checking problem*. Let a logic $L$ and a class $\mathcal{C}$ of structures be fixed, the problem is to decide $G \models \phi$ for arbitrary $G \in \mathcal{C}$ and $\phi \in L$. When measuring the complexity, we try to isolate the part that depends on $\phi$ and the part that depends on $G$. We denote this problem as $\mathbf{MC}(L, \mathcal{C})$. Usually, the complexity is of the form $f(\phi) \cdot p(G)$ where $f$ is arbitrary function and $p$ is a polynomial. This fits well into practical applications as in databases, since usually the structure is big and changing often, whereas the formula is small and fixed – therefore it makes sense to do some formula-specific optimisations.

Let us now survey known results. Let $\mathcal{G}$ be the class of all graphs, both $\mathbf{MC}(\mathbb{FO}, \mathcal{G})$ and $\mathbf{MC}(\mathbb{MSO}, \mathcal{G})$ are PSPACE-complete [110]. The already mentioned result of Courcelle [21] says that $\mathbf{MC}(\mathbb{MSO}_2, \{G\colon \mathrm{tw}(G) \leq k)$ is linear in $|V(G)G|$ and non-elementary in $k$ and qalt of $\phi$. Its analogue for cliquewidth [22] works for $\mathbb{MSO}_1$ only. In addition, there is a counting version [22] which not only tells whether there is an satisfying assignment of variables, but also answers how many such assignments are there.

In particular, the result of Courcelle implies that deciding $\mathrm{td}(G) \leq k$ is FPT in $k$ – first, do a depth-first search on $G$. If the resulting depth is greater than $2^k$, treedepth is strictly greater than $k$. Otherwise, $G$ has treewidth at most $2^k$, and as $\mathrm{td}(G) \leq k$ can be expressed in $\mathbb{MSO}$, we can solve it in time linear in $|V(G)|$.

As every existential $\mathbb{FO}$ property is directly equivalent to finding an induced subgraph—the vertices of the subgraph are evaluations of the variables, the result

[44] implies that $\mathbf{MC}(\exists\mathbb{FO},$ planar graphs) is linear in $|V(G)|$. The result exploits the fact that although planar graphs have unbounded treewidth, locally they are simple – namely, the treewidth is a function of diameter [92]. We say that class $\mathcal{C}$ of graphs has *locally bounded treewidth* if there is a function $f$ such that for every graph $G \in \mathcal{C}$ and every vertex $v \in G$ the graph induced by vertices in distance at most $r$ from $v$ has treewidth at most $f(r)$, for every $r$. Consequently, in [51] it has been shown that $\mathbf{MC}(\mathbb{FO},$ locally bounded treewidth) can be decided in almost linear time in $|V(G)|$. For the graph classes excluding a fixed minor, FPT algorithm for $\mathbb{FO}$ model checking was found in the same year [48]. We note that locally bounded treewidth and minor exclusion are mutually incomparable. In order to obtain a unifying result, the concept of locality has been taken to minors also – a graph class $\mathcal{C}$ *locally excludes a minor* if for every $r$ there is a graph $H_r$ such that $H_r$ is not a minor of graphs induced by $r$-neighbourhoods of vertices of graphs in $\mathcal{C}$. This concept generalises graphs with bounded expansion – if $f$ is the function bounding the expansion, then $K_{f(r)+2}$ are locally forbidden. The fixed-parameter tractability for classes locally excluding a minor has been shown in [25]. However, the result relies on the structural results from the proof of the Graph Minor Theorem.

From the existence of low treedepth colourings [80] follows a linear-time algorithm for testing $\exists\mathbb{FO}$ properties for classes of graphs with bounded expansion. For nowhere-dense classes, this algorithm runs in almost linear time. This result has been extended [37] to all properties expressible in FOL, showing that such properties can be decided in linear time on any class with bounded expansion (the nowhere-dense case is still open, although the result extends to classes of graphs with *locally bounded expansion*, which generalises all previously known results). Conversely, if a class of graphs $\mathcal{C}$ is closed on subgraphs and it is not nowhere-dense, then the subgraph problem restricted to $\mathcal{C}$ is $W[1]$-hard (when parameterised by the subgraph). This follows from the result that subgraph testing for the class of all graphs is $W[1]$-hard, and that the somewhere case is reducible to the all graphs case – let $r$ be such that $\mathcal{C}\nabla r$ is the class of all graphs, then instead of looking for subgraph $H$ we can look for its minors at depth $r$. This shows that the result [80] is essentially the best possible.

### Lower Bounds

The drawback of most of the mentioned results for model checking is that although their dependence on $|V(G)|$ is polynomial or even linear, the dependence on $\phi$ is much worse, usually something like tower of exponents as high as the qalt of $\phi$. It turns out that this kind tower is not avoidable even for $\mathbb{FO}$ on simple classes as trees or coloured paths [72, 52] (unless EXP=NEXP).

This motivated a search for meta-theorems similar to [21] on more restricted classes of graphs, such as the result of Lampis [71] that provides algorithms with better dependence on the size of the formula for classes such as those with bounded vertex cover or bounded max-leaf number. This result was subsequently generalised by Gajarský and Hliněný to graphs with bounded tree-depth [53] – as mentioned in 2.3.1, bounded treedepth forbids the presence of long paths, which is the reason for hardness of model checking. Basically, the height of the tower in this result depends on the treedepth but not on the qrank. We will comment more on the the result [53] in Chapter 3, as we use it in order to prove our result.

Lower bounds that concern the complexity part depending on $G$ must concern classes with unbounded treewidth. For instance, the result [75] says that unless P=NP, $\mathbf{MC}(\mathbb{MSO}_1, \mathcal{C})$ cannot be FPT unless for any graph class $\mathcal{C}$ closed under topological minors. The result relies on the presence of large grids, and the fact that grids enable, very roughly said, Turing machine simulation – see [67] for formalisation of this. A further step in this direction is a result [68] which shows (under some complexity theory assumptions) that $\mathbb{MSO}_2$-model checking is not FPT for any class of graphs whose treewidth is at least $\simeq \log|V(G)|$ (and some technical assumptions). This is an improvement, since grids mean imply only square treewidth. Similar result [54] aims on $\mathbb{MSO}_1$ with slightly different complexity assumptions. The logarithmic bound cannot be significantly improved – in [75] a class of graphs with logarithmic treewidth is shown for which $\mathbb{MSO}_2$-model checking is FPT (the result do not contradict each other because of the constants hidden in asymptotic).

## 2.5 Dynamic Data Structures

In this section we define what does "Dynamic data structure" mean, and survey some existing ones. However, this section is, compared to the previous ones, rather short, as our results do not use much of existing data structure theory; most examples here serve only for illustration of possible applications.

*Dynamic data structure* for deciding graph property $\mathcal{P}$ is a data structure which represents varying graph $G$ and supports the following operations:

- add edge $(u, v)$ – modify the varying graph so that it represents $G + (u, v)$,
- remove edge $(u, v)$ – modify the varying graph so that it represents $G - (u, v)$,
- initialise by $G$ – initialise the structure so that the varying graph it represents $G$,
- query – report whether $G$ has the property $\mathcal{P}$.

This is the most basic setting, which is usually somehow modified. For instance, we might require restriction to some specific graph class – that is, we implicitly assume that all the operations that modify the represented graph respect this restriction. Or the query operation might be extended such that it reports a witness for the property, for instance if the property is represented by an existential first order logic formula, the structure has to report which vertices satisfy the formula.

The complexity of the structure is measured in running time for individual operations. Very often, amortised complexity is used – very roughly, it says that we measure the total complexity of a sequence of operations, instead of focusing on worst case and multiply it by the number of operations (which might be quite unrealistic). For details, see [20].

As for applications, many of the problems occurring in nature are dynamical in principle. For instance, social networks or dynamical systems as those encountered in physics or biology represent phenomena that change over time. Similarly, in computer science we need data structures to describe computer networks or to store databases. In this situation, classical static algorithms have

to be dynamised. For general techniques that can be used for dynamisation, see [88].

Another application occurs when one reduces a graph by removing edges, and each time an edge is removed, some procedure has to be performed. Instead of running the procedure from scratch every time, it makes sense to keep some dynamic information. Classical examples are the usage of a disjoint-find-union data structure in minimal spanning tree algorithms [20] or Link-cut trees for network flow algorithms [103]. A more recent example is for colouring graphs on surfaces, we describe it as a part of our result in Chapter 4.

Possibly oldest dynamic data structure is the AVL tree [2], which represents set of elements ordered by their keys, and allows insertion, deletion and retrieval in logarithmic time. This is an example of a binary search tree, a term which fits to many structures that have appeared since then. Other important binary search trees include splay tree [104] and tango trees [26], which are so far the best data structure in competitiveness sense. Competitiveness is a measure of how well the algorithm performs when compared to an algorithm that can "see the future", that is, has the knowledge of what queries and updates it will have to do, and thus can perform optimally. An area where dynamic data structures flourished is computational geometry, again, see [88] for more details.

# Chapter 3

# Dynamic Tree-depth Decomposition

In this chapter, we describe our application of the theory. It is a dynamic data structure for representing a graph $G$ with treedepth at most $D$. For the definition of treedepth and relevant concepts, see 2.3.1.

The structure allows addition and removal of edges and vertices such that the resulting graph still has tree-depth at most $D$, in time bounds depending only on $D$. A tree-depth decomposition of the graph is maintained explicitly.

This makes the data structure useful for dynamization of static algorithms for graphs with bounded tree-depth. As an example application, we give a dynamic data structure for $\mathbb{MSO}$-model checking, with time bounds for removal depending only on $D$ and constant-time testing of the property, while the time for the initialization and insertion also depends on the qalt of the formula expressing the property.

The result has been co-authored by Zdeněk Dvořák and Martin Kupec, its full version can be found at [38].

Let us state the result precisely.

**Theorem 1.** *Let $\phi$ be a $\mathbb{MSO}_2$ formula and $D \in \mathbb{N}$. There exists a data structure for representing a graph $G$ with $td(G) \leq D$ supporting the following operations:*

- *insert edge $e$, provided that $td(G + \{e\}) \leq D$,*
- *delete edge $e$,*
- *query—determine whether $G$ satisfies the formula $\phi$.*

*The time complexity of deletion depends on $D$ only, in particular, it does not depend on $\phi$ or $|V(G)|$. The time complexity of insertion depends on $\phi$ and $D$, but does not depend on $|V(G)|$. The time complexity of the initialization of the data structure depends on $\phi$, $D$ and $|V(G)|$. The query is done in constant time, as is addition or removal of an isolated vertex.*

The dependence of the initialization and edge insertion is roughly a tower of height $D$ where the highest element of the tower is the qrank of $\phi$ squared – that is, the dependence on $\phi$ is elementary.

The basic idea of the data structure is to explicitly maintain a forest of smallest depth whose closure contains $G$, together with its compact constant-size summary

obtained by identifying equivalent subtrees. This summary is sufficient to decide the property expressed by $\phi$, which we show using the notion of $n$-equivalency from 2.4 and a result from [53]. Informally, the result says that when one is interested in checking whether a specific formula is true on a class of trees of bounded depth, then one can also assume bounded degree.

## 3.1 Auxiliary tools

All trees we work with are rooted. For simplicity, we also assume in this section that all graphs we work with are connected. If we encounter a disconnected graph, we consider each of its connected components individually.

Two trees are isomorphic if there exists a graph-isomorphism between them such that the root is preserved under it. Mostly we will work with trees with vertices labelled from some set of $l$ labels – two $l$-labelled trees are $l$-isomorphic if they are isomorphic as trees and the isomorphism preserves labels. A *limb* of a vertex $v \in T$ is the subgraph induced by some of the children of $v$. A second-order logic formula $\phi$ is in $\mathbb{MSO}$ logic, if all second-order quantifiers are over sets of elements (vertices) and the language contains just the relation $edge(u, v)$.

The following result is a simplification of Lemma 3.1 from [53].

**Lemma 2.** *Let $\phi$ be an $\mathbb{MSO}$ sentence, $l,\ D \in \mathbb{N}$. Then there exists a number $S$ with the following property. Let $T$ be an $l$-labelled tree of depth at most $D$ with vertices labelled with $l$ labels, and $v$ a vertex of $T$. If $v$ has more than $S$ pairwise $l$-isomorphic limbs, then for the tree $T'$ obtained by deleting one of those limbs we have that*

$$T' \text{ satisfies } \phi \iff T \text{ satisfies } \phi.$$

The Lemma implies in particular that with respect to $\phi$-checking there are only finitely many $l$-labelled trees of depth at most $D$ – that is, every $l$-labelled tree of depth at most $D$ is $\phi$-equivalent to some $l$-labelled tree of depth at most $D$ and maximum degree at most $S$. We call such trees *$\phi$-minimal*.

Let $G$ be a graph of tree-depth $D$, the *tree decomposition* $T$ of $G$ is a $2^{D-1}$-labelled tree such that $G \subseteq \mathrm{clos}(T)$, where a vertex $v$ is labelled by a 0-1 vector of length $D-1$ that encodes the edges between $v$ and the vertices on the path from $v$ to the root (1 whenever the edge is present, 0 otherwise). Let $l_D$ be a set of labels we describe later, *compressed tree decomposition* of the graph $G$ is an $l_D$-labelled tree $C$ obtained from a tree decomposition $T$ of $G$ as follows. For every vertex, all its limbs that are pairwise-isomorphic are deleted except for one representative, in which we additionally store the number of these limbs. Vertices of $C$ are called *cabinets*, and the underlying tree decomposition $T$ is called a *decompression* of $C$. A set of all vertices corresponding to the same cabinet (that is, inducing $l_D$-isomorphic limbs) and having the same vertex as a father in the decompression is called a *drawer*. Thus every cabinet is disjointly partitioned into drawers. For an example how a graph, its tree decomposition and compressed tree decomposition look like, see the figures on page 27 (in the compressed tree decomposition, the number next to the drawers denotes how many vertices are there in each drawer).

Now we describe the labelling. We start inductively, with $l_0$ being just a set of vectors of length $D-1$. Assume that $\phi$ is some fixed formula we specify later (in Section 3.2.5) and let $S$ be the number obtained from applying Lemma 2 to

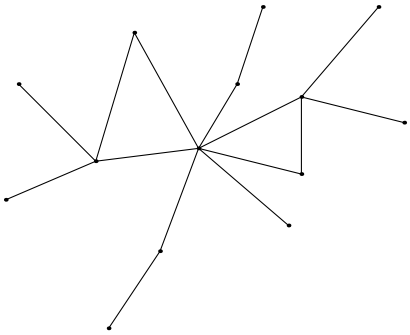


Figure 3.1: Graph

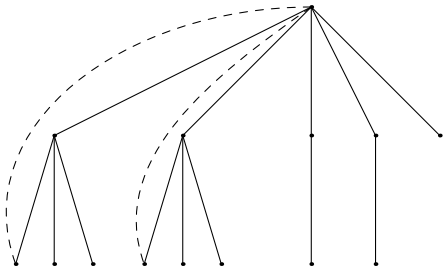


Figure 3.2: Tree-depth decomposition

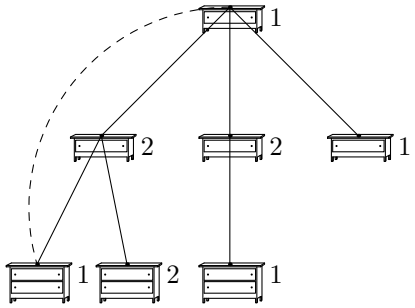



Figure 3.3: Compressed graph

it. Let $B$ be a cabinet that induces a subtree of depth $t' \leq t$ in $C$. The label of $B$ consists of the label of a corresponding vertex $b$ of $T$ and of a vector $vec$ with entry for every $l_{t-1}$-labelled $\phi$-minimal tree $M$ of depth smaller than $t'$ with value

$$vec_M = \min\{S, \text{number of limbs of } b \text{ which are } \phi\text{-equivalent to } M\}.$$

During the update operations, we will be occasionally forced to have more than one cabinet for a given isomorphism type (that is, a cabinet will have two pairwise-isomorphic limbs). Both the decomposition and the individual cabinets that have isomorphic children will be called *dirty*.

## 3.2 Data structure

Our data structure basically consists of storing some extra information for every vertex $v$ of the represented graph $G$, and of a compressed tree-depth decomposition $T$ of $G$ with depth at most $D$. We will store the following for every $v \in G$.

- Label of the cabinet corresponding to $v$, that is, the vector of its neighbors on the path to root and the vector with the numbers of limbs of $v$ isomorphic to individual $\phi$-minimal trees.
- Pointer to the father of $v$ in $T$ (more precisely, pointer to a vertex $u$ of $G$ that is the father of $v$ in the decompression of $T$). However, in some operations we need to change the father of all vertices in a drawer at once – thus instead of storing father individually for every vertex, for every drawer we will maintain a pointer to the common father of the vertices in this drawer, and every vertex in the drawer will have a pointer to this pointer.

- Linked list of sons of $v$ in $T$. This is again implemented by having a linked list of drawers at $v$, and for every drawer in it, a linked list of vertices in this drawer.

Additionally, we keep the vertex $v$ which is the root of $T$ and we call it $r$ – we again assume connectedness of $G$ in this section, otherwise we keep a list of roots corresponding to individual components.

### 3.2.1 Extraction of a path

In this subsection we describe an auxiliary operation of extracting a path. It can be seen as a temporary decompression of a part of $T$ in order to make some vertex accessible. The result of extracting $v$ from $T$ is a dirty compressed tree decomposition $T'$ of $G$, such that on the cabinets in $T'$ corresponding to $r - v$ path there are no cabinets to which corresponds more than one vertex of $G$.

First, we find the vertices of the $r - v$ path, and the corresponding cabinets in $T$. This is done by simply following the father-pointers from $v$, and then by going backwards from $r$, always picking the cabinet that corresponds to the label of the vertex on the $r - v$ path. Then, for every cabinet $B$ on this path with more than one vertex, let $b$ be the vertex of the $r - v$ path lying in $B$, and $c$ its father – which we assume to be the only vertex in its cabinet, $C$. We remove $b$ from the lists of sons of $c$ of the label of $b$, and move $b$ into a new list for $c$, and do the corresponding change in $T'$, that is, creating a new cabinet of the same label as a son of $C$, thus making $C$ a dirty cabinet.

The complexity of this operation is clearly linear in $D$.

### 3.2.2 Edge deletion

Edge deletion is simple – let $vu$ be the edge to be deleted, with $v$ the lower vertex (in the tree-order imposed by $T$). We extract the vertex $v$ from $T$. Now $u$ lies on the $r - v$ path, and as there are no other vertices in the cabinets on the corresponding path in $T$, we remove the edge $vu$ from the graph and change the labels for the cabinets and vertices accordingly. The only affected labels are on the $r - v$ path, and we will precompute during initialization what the label should change into. It can also happen that removal of such edge disconnects the graph – this also depends only on labels and thus will be precomputed in advance. When such situation occurs, we split $T$ into two components – the new root depends only on the labels, and the vertices for which labels change are only on the $r - v$ path.

Now, we need to clean the dirty cabinets. As the only dirty cabinets are on the $r-v$ path, we traverse this path, starting from $v$ and going upwards, and for every vertex $w$ in a dirty cabinet, we compare the label of $w$ with the labels of other present drawers at the father of $w$, and move $w$ to the correct drawer/cabinet.

The complexity of this operation is clearly linear in $D$.

### 3.2.3 Rerooting

Rerooting is also an auxiliary operation, which will allow us to easily handle the edge insertion. This operation takes a compressed tree decomposition $T$ and a

vertex $r_N$ of $G$, for which we have a guarantee that there is a tree decomposition with depth at most $D$ such that $r_N$ is its root, outputs one such compressed tree decomposition $T'$ and updates data for vertices in $G$ accordingly. In this subsection we denote by $r_O$ the root of $T$, that is, the old root.

We proceed as follows:

1. extract $r_N$ from $T$,
2. remove $r_N$ from $T$ entirely,
3. consider the connected components thereof – those that do not contain $r_O$ have depth $< D$ and thus can be directly attached under $r_N$. Recurse into the component with $r_O$.

Only the third point deserves further explanation. The components are determined by the labels only, so we will precompute which labels are in which components and what vertices are the roots of the components. Every connected component of $T - r_N$ that does not contain $r_O$ must have as its highest vertex (under the tree-order) a son of $r_N$, thus these components are already in their proper place. For the component $C$ with $r_O$, either it has depth $< D$ and thus can be attached under $r_N$, with $r_O$ being a son of $r_N$. We have to deal with two details – firstly, there might be some edges to $r_N$ from vertices that were above $r_N$ in $T$ – but none of these vertices was in a cabinet with more than one vertex, thus we only change the labels accordingly.

Secondly, the limbs of $r_N$ in $T$ that are in $C$ have no father after removal of $r_N$. But as they are in $C$, for every such limb there is an edge from it to some vertex on the $r_N - r_O$ path $T$. Choose lowest such vertex, and make it new father for that limb. This refathering is done by using the pointers for the drawers – note that every cabinet that is a root of such limb consist only of single drawer, thanks to the extraction of $r_N$. Thus the total number of operations we have to do is linear in $D$ and the maximum number of children of $r_N$, which is $l_d$. As in the case of edge deletion, we have to clean dirty cabinets (which are in $C$) in the end. This can again be done by simply comparing labels on that former $r_N - r_O$ path.

However, it might happen that $C$ has depth exactly $D$. But we are guaranteed that there exist a tree decomposition with $r_N$ as a root, which implies that there exists a tree decomposition of $C$ with depth $D - 1$. If we know which vertex can serve as a root of such decomposition, we can apply the operation recursively. We describe the procedure to find a root in Section 3.2.5. An additional thing we have to care about is that some vertices of $C$ have an edge to $r_N$ – this information has to be preserved in the recursive call. But the number of such vertices is bounded by a function of $D$ and thus it is not a problem – we only modify their labels accordingly. After this recursive call, we again clean dirty cabinets.

The complexity of this operation for one call is linear in $D + l_d+$ time to find new root, and there are at most $D$ recursive calls.

### 3.2.4 Edge insertion

Let $u, v$ be two vertices not connected with an edge, such that $G + uv$ has treedepth at most $D$, we now describe how to add such edge. If the edge $uv$

respects the tree-order (that is, either $u$ lies on $v - r$ path or vice versa), we just extract the lower of the two vertices, add the edge, and get rid of dirty cabinets.

Otherwise, there exists a vertex $r_1$ which is a root of some tree decomposition of $G + uv$. We describe the procedure for finding it in Section 3.2.5. Reroot into this vertex to obtain decomposition $T_1$. Now, $u$ and $v$ must be in the same connected component $C_1$ of $T_1 - r_1$. Again, unless the edge $uv$ respects the tree-order now, we can find a vertex $r_2$ in $C_1$ which is a root of some tree decomposition of $C_1 + uv$ of depth at most $D - 1$, and reroot into it to obtain decomposition $T_2$ of $C_1$, with $u$ and $v$ lying in the same component $C_2$ of $C_1 - r_2$. Carrying on in the obvious manner, this process stops after at most $D$ iterations.

The complexity of this operation is $O(D\cdot$ complexity of rerooting).

Finally, we just remark that addition and removal of a vertex (without incident edges) is implemented trivially by just adding/removing new component with the corresponding label.

### 3.2.5 Finding a root

Let us recall what we have to face in this section. We want to find a vertex $v$ such that there is a tree $T$ of depth at most $t' \leq t$ such that its closure contains the connected component $C$ of the graph $G+(a,b)-\{v_1, v_2, \ldots, v_k\}$ as a subgraph and $v$ is a root of $T$. The vertices $v_1, v_2, \ldots, v_k$ correspond to the roots found in previous applications of this procedure, $(a, b)$ denotes the edge we are trying to add.

At this point we define the formula $\phi$ according to which we constructed the labelling of our trees. Let $\gamma(C)$ be a formula which is true whenever $C$ is connected — this is easily seen to be expressible in $\mathbb{MSO}$ logic — and $\tau_d(G)$ the following formula:

$$\tau_d(G) = (\exists v \in G)(\forall C \subseteq G)(\gamma(C - \{v\}) \Rightarrow \tau_{d-1}(C - \{v\})),$$

with $\tau_1(v)$ being always true. Then $\tau_d(G)$ says that there exists a tree $T$ with depth at most $d$ such that $G \subseteq \mathrm{clos}(T)$. Furthermore, as we need to express the addition of an edge, we work with the logic with two extra constants $a, b$, and modify the formula for $\gamma$ accordingly to obtain $\gamma'$ and $\tau'$. The resulting formula $\tau'_t$ is the formula $\phi$.

Using Lemma 2, we construct all $\phi$-minimal trees – note that we have to consider every possible evaluation of the constants $a, b$, that is, we construct all trees of depth at most $t$ such that no vertex has more than $S$ pairwise-isomorphic limbs, and then for every two of labels, we choose two arbitrary vertices having that label, and choose them to be $a$ and $b$. For every such minimal tree, we evaluate the formula, that is, we find which vertex is to be the root of the tree decomposition. It might happen that the formula is false, that is, no such vertex exists, which means we evaluated $a$ and $b$ so that the graph has tree-depth greater than $d$. But such evaluation will not occur during the run of the structure — recall that we restricted the edge additions — and thus we can safely discard these minimal trees. Thus for every minimal tree we store the label of the vertices that can be made root, and when applying the rerooting subroutine, we find an arbitrary vertex of this label. This has complexity at most $D$, because when

looking for the given vertex, we follow first pointer from the corresponding linked list of children for a vertex.

This means that the total complexity of the edge insertion is $O(D(D + l_D))$. Finally, let us remark on the complexity of initialization. From [53] we conclude that $l_D$, that is, the number of $\phi$-minimal trees, is roughly a tower of 2's of height linear in $D$, to the power $|\phi|^2$. The complexity of operations we do for every $\phi$-minimal tree is bounded by a polynomial in $D$ and $l_D$.

### 3.2.6 Dynamic model checking

We now describe how to modify the structure so that it also allows queries of the form "does $G$ satisfy the formula $\varphi$", where $\varphi$ is some fixed MSO formula. The modifications affect only the Section 3.2.5. Instead of using just the formula $\phi$ to obtain the $\phi$-minimal trees, we apply the Lemma 2 to the formula $\varphi$ also and in the construction of the minimal trees and the labelling, we use the higher of the two numbers obtained from the Lemma. Then for every such obtained minimal tree we evaluate whether it satisfies $\varphi$ or not, this time without evaluating the constants $a, b$. Finally, we remark that in principle, we are computing the type of $G$, so we do not have to have the formula $\phi$ fixed, but can switch between formulae of the same qalt. This is common to many results in this area.

# Chapter 4

# The Subgraph Problem

In this chapter we present a dynamic data structure for representing a graph $G$, which allows addition and removal of edges from $G$ and can determine the number of appearances of a graph $H$ as an induced subgraph of $G$. The graph $H$ is arbitrary but fixed for the design of the structure. The queries are answered in constant time. When the data structure is used to represent graphs from a class with bounded expansion the amortized time complexity of updates is polylogarithmic. For nowhere dense classes, the complexity is subpolynomial. This result has been co-authored by Zdeněk Dvořák, and appeared as [39].

An exemplar application of such data structure is an algorithm for finding 5-coloring of a graph on torus, based on the result of Thomassen [109]. Here, the algorithm performs various reductions of the considered graph, and after each reduction, it needs to test whether the reduced graph contains some of four specific subgraphs. Rather than running a subgraph testing algorithm each time, it makes more sense to update the information about subgraphs dynamically.

We actually deal with the counting version of the problem, i.e., determining how many times does a fixed graph $H$ appear as an induced subgraph in the represented graph. This generalisation has applications in Bioinformatics and Social Networking research – for instance, see [76] and [100].

As the problem of subgraph testing is $W[1]$-hard, unless $W[1]$ =FPT, the discussed data structure cannot have both subpolynomial update and query times when used to represent general graphs (or even graphs from a hereditary class that is not nowhere-dense). The best known general algorithms for the static variant of the problem are based on matrix multiplication; Nešetřil and Poljak [86] gave an $O(|V(G)|^{\omega|V(H)|/3})$-time algorithm, where $\omega$ is the exponent in the complexity of matrix multiplication. This was subsequently refined in [63, 42].

The already mentioned result [37] also provided a semidynamic data structure for the problem. For a fixed first-order formula $\phi$ and a class of graphs $\mathcal{C}$ with bounded expansion, this data structure represents a graph $G \in \mathcal{C}$ and can be initialized in time $O(|V(G)|)$. The data structure enables testing whether the graph satisfies $\phi$ in constant time. However, the data structure is only semi-dynamic – the graph can be modified by adding and removing edges in constant time, but the edge additions are restricted: we can only add edges that were removed before.

In this paper, we eliminate the restriction on edge additions; that is, our data structure allows addition of arbitrary edges, subject to the restriction that the

resulting graph still belongs to the considered (bounded expansion or nowhere-dense) class of graphs. On the other hand, we only handle the case of subgraph testing ($\exists\mathbb{FO}$), not testing of general FO properties.

Let us now state our result precisely.

**Theorem 3.** *Let $H$ be a fixed graph and let $\mathcal{C}$ be a class of graphs. There exists a data structure $\mathrm{ISub}_H(G)$ representing a graph $G \in \mathcal{C}$ which supports the following operations.*

- *Determine the number of induced subgraphs of $G$ isomorphic to $H$.*
- *Add an edge $e$, i.e., transform $\mathrm{ISub}_H(G)$ to $\mathrm{ISub}_H(G+e)$, under the assumption that $G+e$ is in $\mathcal{C}$.*
- *Delete an edge $e$, i.e., transform $\mathrm{ISub}_H(G)$ to $\mathrm{ISub}_H(G-e)$, under the assumption that $G-e$ is in $\mathcal{C}$.*

*If $\mathcal{C}$ has bounded expansion, then the time complexity of query and edge removal is $O(1)$, while the amortized time complexity of edge addition is $O(\log^h |V(G)|)$, where $h = \binom{|V(H)|}{2} - 1$. The structure can be initialized in $O(|V(G)|)$ and the space complexity for the structure is $O(|V(G)|)$. If $\mathcal{C}$ is nowhere-dense, then the time complexity of query is $O(1)$, the amortized time complexity of edge addition or removal is $O(|V(G)|^\varepsilon)$, the time complexity of the initialization is $O(|V(G)|^{1+\varepsilon})$ and the space complexity is $O(|V(G)|^{1+\varepsilon})$, for every $\varepsilon > 0$.*

Using this data structure, we can also count graph inclusions other than induced subgraphs (e.g., subgraphs and homomorphisms), as these counts only depend on which and how many small induced subgraphs appear in $G$. Furthermore, it is easy to modify the data structure to apply to objects other than undirected graphs, e.g., to directed graphs with colors on vertices and edges.

The problem of dynamic subgraph counting was introduced by Eppstein et al. in [46] and later extended in [45]. The *h-index* of a graph $G$ is the largest integer $h$ such that $G$ has at least $h$ vertices of degree at least $h$. Let $\mathcal{C}_h$ denote the class of graphs with h-index at most $h$. Using a differente approach, the data structure of et al. [45] makes it possible to determine the number of all induced subgraphs with *at most four vertices* in constant time, with time complexity $O(h^2)$ per modification if it is used to reprezent a graph in $\mathcal{C}_h$. Note that the class $\mathcal{C}_h$ is closed on topological minors, and thus it has bounded expansion. Therefore, Theorem 3 generalizes this result, but it has somewhat worse time complexity per operation.

The rest of this chapter is organized as follows. In Section 4.1, we describe a basic idea of the data structure for induced subgraphs. In Section 4.2, we give some definitions and auxiliary results needed in the rest of the paper. Section 4.3 contains detailed description of the data structure.

## 4.1 Basic idea

For concreteness, in this section we consider the class of planar graphs, rather than an arbitrary class with bounded expansion. Suppose that we want to keep track of triangles in a planar graph $G$. A simple way to do this is as follows.

Orient the edges of $G$ so that every vertex has in-degree at most 6, which is possible by 5-degeneracy of planar graphs. For an edge $xy$ of $G$, we write $x \to y$ if the edge is oriented towards $y$. For each vertex $u \in V(G)$, we maintain

- the number $n_1(u)$ of pairs of vertices $v, w \in V(G)$ such that $u \to v$, $v \to w$ and $u \to w$;
- the number $n_2(u)$ of pairs of vertices $v, w \in V(G)$ such that $u \to v$, $v \to w$ and $w \to u$.

We also maintain the sums $N_1 = \sum_{u \in V(G)} n_1(u)$ and $N_2 = \sum_{u \in V(G)} n_2(u)$. Consider a triangle $T \subseteq G$ with vertex set $\{x, y, z\}$. By symmetry, we can assume that $x \to y$ and $y \to z$. If $x \to z$, then $T$ contributes 1 to $n_1(x)$. If $z \to x$, then $T$ contributes 1 to each of $n_2(x)$, $n_2(y)$ and $n_2(z)$. Therefore, the number of triangles in $G$ is $N_1 + N_2/3$.

Let us add an edge $xy$ to $G$ and choose its orientation, say $x \to y$. Assume for now that in the resulting orientation, the in-degree of $y$ is still at most 6. Which of the numbers that we maintain are affected? Clearly, if $n_1(u)$ or $n_2(u)$ changes, then either $u$ is incident with the edge $xy$, or $u$ is an in-neighbor of $x$. Thus, we only need to update information for at most 8 vertices of $G$.

Updating an in-neighbor $u$ of $x$ in constant time is easy, as we just need to check whether the path $u \to x \to y$ contributes to $n_1(u)$ and $n_2(u)$ or not. For $y$, the number $n_1(y)$ is unchanged, while the number $n_2(y)$ increases by the number of vertices $v$ such that $y \to v \to x$. We can enumerate such vertices in a constant time, as they are in-neighbors of $x$. Similarly, we can update $n_2(x)$ in a constant time, as the path $x \to y \to z$ only contributes to $n_2(x)$ if $z$ is an in-neighbor of $x$.

There are two ways $n_1(x)$ can be affected by the addition of $x \to y$. It could be that there exists $v \in V(G)$ with $x \to v \to y$. All these vertices are in-neighbors of $y$, and they can be enumerated in constant time. The most complicated case is that there exists a vertex $z$ with $x \to z$ and $y \to z$. Here, we cannot easily enumerate all possibilities for $z$, as we do not have any bound on out-degrees of vertices. Therefore, we need one more piece of information. For each pair of distinct vertices $u, v \in V(G)$, let $n_3(u, v)$ be the number of vertices $w \in V(G)$ with $u \to w$ and $v \to w$. In a hash table, we store

- the number $n_3(u, v)$ for all pairs $u, v \in V(G)$ such that $n_3(u, v) \neq 0$.

Hence, in the last case of the update of $n_1(x)$, we just need to add $n_3(x, y)$.

Let us note that since each vertex has at most 6 in-neighbors, the number $n_3(u, v)$ is non-zero for at most $\binom{6}{2}|V(G)|$ pairs $u, v \in V(G)$, and thus $n_3$ can be stored in linear space. We need to consider how the addition of $x \to y$ affects $n_3$. If $n_3(u, v)$ changes, then by symmetry we can assume that $u = x$ and $v \to y$. Consequently, $v$ is an in-neighbor of $y$, and thus we can enumerate in constant time all (at most 5) pairs $u, v \in V(G)$ such that $n_3(u, v)$ increases.

This finishes the description of the update in the case of edge addition. Edge removal is handled similarly. One problem that we skipped is what to do when the addition of an edge would violate the constraint on the maximum in-degree. We are saved by Brodal and Fagerberg [17], who provided an algorithm for maintaining an orientation with bounded maximum in-degree, which only needs to change orientation of $O(\log |V(G)|)$ edges per update (amortized). In our data structure, the edge reorientations can be handled similarly to edge additions.

Therefore, we have described a data structure for counting the number of triangles in a planar graph (or indeed, any graph with bounded degeneracy), with logarithmic time complexity per update. For a general subgraph $H$, there appear additional complications. The idea of maintaining for each vertex $v$ the number of copies in $H$ in the subgraph reachable by short paths from $v$ (a similar idea appears already in Chrobak and Eppstein [18]) only works for the orientations of $H$ that contain a directed Hamiltonian path. As a first step, we extend this to the case that $H$ contains a spanning outbranching without cross edges, at the expense of only counting homomorphisms from $H$ to $G$ instead of subgraphs. This is not a big problem, as counting subgraphs is equivalent to counting homomorphisms through a standard inclusion-exclusion argument.

However, how to deal with the orientations that do not admit such an outbranching? For this, we use the idea of fraternal augmentations of from [82]. Essentially, we add new edges to $H$ and $G$, obtaining new graphs $H'$ and $G'$, in such a way that we can recover the original number of homomorphisms, but $H'$ contains a spanning outbranching without cross edges. The results of [82] and the assumption of bounded expansion or nowhere-dense class of graphs ensure that $G'$ has bounded degeneracy. This is the most technically complicated part of the argument, formalized in Lemmata 13 and 7.

## 4.2 Definitions and auxiliary results

A fundamental tool for our result are fraternal augmentations, which are special case of augmentations described in 2.3.2. Suppose that $G$ is a directed graph. Vertices $u, v \in V(G)$ form a *fork* if $u$ and $v$ are distinct and non-adjacent and there exists $w \in V(G)$ with $u \to w, v \to w \in E(G)$. Let $G'$ be a graph obtained from $G$ by adding the edge $u \to v$ or $v \to u$ for every pair of vertices $u$ and $v$ forming a fork. Then $G'$ is called a *fraternal augmentation* of $G$. Let us remark that a directed graph can have several different fraternal augmentations, depending on the choices of directions of newly added edges. If $G$ has no fork, then $G$ is called *elder graph*. For an undirected graph $G$, a *$k$-th augmentation of* $G$ is a directed graph $G'$ obtained from an orientation of $G$ by iterating fraternal augmentation (for all forks) $k$ times. Note that $(\binom{|V(G)|}{2} - 2)$-th augmentation of $G$ is an elder graph, because any graph with at most 1 edge is already elder and fraternal augmentation of a non-elder graph adds at least one edge.

The following result from [82] shows that fraternal augmentation preserve bounded expansion and nowhere-denseness.

**Theorem 4.** *There exist polynomials $f_0$, $f_1$, $f_2$, ... with the following property. Let $G$ be a graph with expansion bounded by a function $g$ and let $G_1$ be an orientation of $G$ with in-degree at most $D$. If $G'$ is the underlying undirected graph of a fraternal augmentation of $G_1$, then $G'$ has expansion bounded by the function $g'(r) = f_r(g(2r+1), D)$.*

The fraternal augmentations are a basic tool for deriving properties of graphs with bounded expansion, e.g., existence of low tree-depth colorings. Once such a coloring is found, the subgraph problem can be reduced to graphs with bounded tree-width, where it can be easily solved in linear time by dynamic programing. However, we do not know how to maintain a low tree-depth coloring dynamically

(indeed, not even an efficient data structure for maintaining say a proper 1000-coloring of a planar graph during edge additions and deletions is known). The main contribution of our result is showing that we can count subgraphs using just the fraternal augmentations, which are easier to update.

To maintain orientations of a graph, we use the following result by Brodal and Fagerberg [17]:

**Theorem 5.** *There exists a data structure that, for a graph $G$ with $\nabla_0(G) \leq d$, maintains an orientation with maximum in-degree at most $4d$ within the following bounds:*

- *an edge can be added to $G$ (provided that the resulting graph $G'$ still satisfies $\nabla_0(G') \leq d$) in amortized $O(\log n)$ time, and*
- *an edge can be removed in $O(1)$ time, without affecting the orientation of any other edges.*

*The data structure can be initialized in time $O(|V(G)| + |E(G)|)$. During the updates, the edges whose orientation has changed can be reported in the same time bounds. The orientation is maintained explicitly, i.e., each vertex stores a list of in- and out-neighbors.*

Let us remark that the multiplicative constants of the $O$-notation in Theorem 5 do not depend on $d$, although the implementation of the data structure as described in the paper of Brodal and Fagerberg requires the knowledge of $d$.

Using Theorem 4, we obtain the following modification of the Theorem 5.

**Theorem 6.** *For every $k \geq 0$, there exists an integer $k'$ and a polynomial $g$ with the following property. Let $\mathcal{C}$ be a class of graphs and $h(n, r)$ a computable function such that the expansion of every graph $G \in \mathcal{C}$ is bounded by $f(r) = h(|V(G)|, r)$. There exists a data structure representing a $k$-th augmentation $\tilde{G}_k$ of a graph $G \in G$ with $n$ vertices within the following bounds, where $D = g(h(n, k'))$:*

- *the maximum in-degree of $\tilde{G}_k$ is at most $D$,*
- *an edge can be added to $G$ (provided that the resulting graph still belongs to $\mathcal{C}$) in an amortized $O(D \log^{k+1} n)$ time, and*
- *an edge can be removed in $O(D)$ time, without affecting the orientation of any other edges.*

*The data structure can be initialized in time $O(Dn + t)$, where $t$ is the time necessary to compute $D$. The orientation is maintained explicitly, i.e., each vertex stores a list of in- and out-neighbors.*

*Proof.* Let $q_0(r) = h(n, r)$. We use the data structure of Theorem 5 to provide an orientation $G_0$ of $G'_0 = G$ with maximum in-degree at most $4q_0(0)$. Assume inductively that we have already constructed pairwise edge-disjoint directed graphs $G_0, G_1, \ldots, G_i$ with underlying undirected graphs $G'_0, G'_1, \ldots, G'_i$, such that the expansion of $\tilde{G}'_i = G'_0 \cup \ldots \cup G'_i$ is bounded by a function $q_i(r)$ and $G_i$ has maximum in-degree at most $4q_i(0)$. We define $G'_{i+1}$ to be the graph with vertex set $V(G)$ and with edges corresponding to the forks in $\tilde{G}_i = G_0 \cup \ldots \cup G_i$. Note that

$\tilde{G}_i$ has maximum in-degree at most $d_i = 4(q_0(0) + \ldots + q_i(i))$. By Theorem 4, $\tilde{G}'_{i+1} = G'_0 \cup \ldots \cup G'_{i+1}$ has expansion bounded by $q_{i+1}(r) = f_1(q_i(f_2(r)), d_i)$. We use the data structure of Theorem 5 to provide an orientation $G_{i+1}$ of $G'_{i+1}$ with maximum in-degree at most $4q_{i+1}(0)$.

The data structure maintains the orientations $G_0$, $G_1$, ..., $G_k$ and their union $\tilde{G}_k$. Observe that $\tilde{G}_k$ is a $k$-th augmentation of $G$. Addition of an edge in $G$ may result in change of orientation of $O(\log n)$ edges in $G_0$ (amortized), which may result in addition or removal of $O(d_0 \log n)$ edges in $G'_1$. Each of them results in change of orientation of $O(\log n)$ edges in $G_1$ and consequently addition or removal of $O(d_1 \log n)$ edges in $G_2$, etc. Altogether, addition of an edge may result in $O(d_0 d_1 \ldots d_{k-1} \log^{k+1} n)$ changes, with the same time complexity. Similarly, a removal of an edge may result in $O(d_0 d_1 \ldots d_{k-1})$ changes.

Therefore, Theorem 6 holds, since we can choose the integer $k'$ and the polynomial $g$ so that $D \geq \max(d_k, d_0 d_1 \ldots d_{k-1})$. Let us remark that we can assume that $D \leq n^2$, as otherwise the claim of the theorem is trivial; hence, the complexity of performing computations with $D$ (once it was determined during the initialization) does not affect the time complexity of the operations. □

Let $G$ be a directed graph and $S$ a set of its vertices. Let $N^+_d(S)$ denote the set of vertices that are reachable from $S$ by a directed path of length at most $d$, and let $N^+_\infty(S)$ we denote the set of vertices reachable from $S$ by a directed path of any length. Similarly, $N^-_d(S)$ and $N^-_\infty(S)$ denote the sets of vertices from that $S$ can be reached by a directed path of length at most $d$ and by a directed path of any length, respectively. We also use $N^+_d(v)$, $N^+_\infty(v)$, $N^-_d(v)$, $N^-_\infty(v)$ as shorthands for $N^+_d(\{v\})$, $N^+_\infty(\{v\})$, $N^-_d(\{v\})$, $N^-_\infty(\{v\})$, respectively. We say that a directed graph is *connected* if its underlying undirected graph is connected. Similarly, *connected components* of a directed graph are its subgraphs induced by vertex sets of the connected components of its underlying undirected graph.

The key property of elder graphs is that they contain a vertex from that we can reach all other vertices by directed paths. Let us prove a stronger claim that we need in the design of our data structure. A directed tree $T$ with all edges directed away from the root is called an *outbranching*. The root of $T$ is denoted by $r(T)$. Let $H$ be a supergraph of an outbranching $T$ with $V(H) = V(T)$, such that for every edge $(t_1, t_2) \in E(H)$, there exists a directed path in $T$ either from $t_1$ to $t_2$ or from $t_2$ to $t_1$. We call such a pair $(H, T)$ a *vineyard*.

**Lemma 7.** *If $H$ is a connected elder graph, then there exists an outbranching $T \subseteq H$ such that $(H, T)$ is a vineyard.*

*Proof.* The claim is obvious if $|V(H)| \leq 2$. Therefore, suppose that $|V(H)| \geq 3$. By induction, we can assume that the claim holds for all graphs with less than $|V(H)|$ vertices. Let $v$ be a vertex of $H$ such that $v$ is not a cutvertex in the underlying undirected graph of $H$. Note that every induced subgraph of an elder graph is elder, and thus by the induction hypothesis, there exists an outbranching $T'$ such that $(H - v, T')$ is a vineyard. If $(v, r(T')) \in E(H)$, then we can let $T$ consist of $T'$ and the edge $(v, r(T'))$. Therefore, assume that $r(T') \notin N^+_1(v)$.

Consider a vertex $w \in N^+_1(v)$, and let $r(T') = w_0, w_1, \ldots, w_k = w$ be the directed path in $T$ from $r(T')$ to $w$. Since both $(w_{k-1}, w_k)$ and $(v, w_k)$ are edges of an elder graph $H$, it follows that either $(w_{k-1}, v)$ or $(v, w_{k-1})$ is an edge of $H$. In latter case, we can repeat this observation. Since $r(T') \notin N^+_1(v)$, we conclude

that there exists $i$ with $0 \leq i \leq k-1$ such that $w_i \in N_1^-(v)$ and $w_j \in N_1^+(v)$ for all $j$ with $i+1 \leq j \leq k$.

In particular, since $G$ is connected, $N_1^-(v)$ is not empty. If $u_1$ and $u_2$ are distinct vertices in $N_1^-(v)$, then since $H$ is an elder graph, there exists an edge joining $u_1$ with $u_2$. Since $(H,T)$ is a vineyard, there exists a directed path $Q \subseteq T'$ starting in $r(T')$ such that $N_1^-(v) \subseteq V(Q)$ and the endvertex $z$ of $Q$ belongs to $N_1^-(v)$.

Let $T_1, \ldots, T_m$ be all components of $T'-V(Q)$ containing at least one neighbor of $v$. As we observed before, we have $(v, r(T_i)) \in E(H)$ for $1 \leq i \leq m$. Let $T$ be the outbranching obtained from $T'$ by removing the incoming edges of $r(T_1), \ldots, r(T_m)$ and adding the edges $(z,v)$, $(v,r(T_1)), \ldots, (v,r(T_m))$.

All neighbors of $v$ belong either to one of the trees $T_1, \ldots, T_m$ or to $Q$, and thus they are joined to $v$ by a directed path in $T$. Consider an edge $(x,y) \in E(H-v) \setminus E(T)$. If neither $x$ nor $y$ belongs to $X = V(T_1) \cup V(T_2) \cup \ldots \cup V(T_m)$, then the path in $T'$ joining $x$ and $y$ also appears in $T$. If both $x$ and $y$ belong to $X$, then since $(H-v, T')$ is a vineyard, there exists $i$ (with $1 \leq i \leq m$) such that $x, y \in V(T_i)$, and the path joining $x$ and $y$ in $T'$ also appears in $T$. Finally, suppose that say $x$ belongs to $T_1$ and $y$ does not belong to $X$. Since $(H-v, T')$ is a vineyard, we have $y \in V(Q)$, and $x$ and $y$ are joined in $T$ by the path consisting of the subpath of $Q$ from $y$ to $z$, the path $zvr(T_1)$ and the path from $r(T_1)$ to $x$ in $T_1$. Therefore, $(H,T)$ is a vineyard. □

Let $G$ and $H$ be undirected graphs, a mapping $\phi\colon V(H) \to V(G)$ is a *homomorphism* if for every edge $uv \in E(H)$, we have that $\phi(u)\phi(v)$ is an edge of $G$ (in particular, $\phi(u) \neq \phi(v)$). A homomorphism is a *subgraph* if it is injective. It is an *induced subgraph* if it is injective and $\phi(u)\phi(v) \in E(G)$ implies $uv \in E(H)$, for every $u, v \in V(H)$. Let $\hom(H,G)$, $\mathrm{sub}(H,G)$ and $\mathrm{isub}(H,G)$ denote the number of homomorphisms, subgraphs and induced subgraphs, respectively, of $H$ in $G$. Let us note that the definitions of subgraph and induced subgraph distinguish the vertices, i.e., $\mathrm{sub}(H,H) = \mathrm{isub}(H,H)$ is equal to the number of automorphisms of $H$.

Similarly, if $H$ and $G$ are directed graphs, a mapping $\phi\colon V(H) \to V(G)$ is a *homomorphism* if $u \to v \in E(H)$ implies that $\phi(u) \to \phi(v)$ is an edge of $G$, and $\hom(H,G)$ denotes the number of homomorphisms from $H$ to $G$. It turns out to be convenient to work with graphs with colored edges. We do not place any restrictions on the coloring; in particular, edges incident with the same vertex can have the same color. Suppose that $H$ and $G$ are graphs with colored edges. A mapping $\phi\colon V(H) \to V(G)$ is a *homomorphism* if for every edge $uv \in E(H)$, we have that $\phi(u)\phi(v)$ is an edge of $G$ of the same color as $uv$ (and in particular, $\phi(u) \neq \phi(v)$). A homomorphism is a *subgraph* if it is injective. It is an *induced subgraph* if it is injective and $\phi(u)\phi(v) \in E(G)$ implies $uv \in E(H)$, for every $u, v \in V(H)$. Let $\hom(H,G)$, $\mathrm{sub}(H,G)$ and $\mathrm{isub}(H,G)$ denote the number of homomorphisms, subgraphs and induced subgraphs, respectively, of $H$ in $G$. Let us note that the definitions of subgraph and induced subgraph distinguish the vertices, i.e., $\mathrm{sub}(H,H) = \mathrm{isub}(H,H)$ is equal to the number of automorphisms of $H$.

Analogously, if $H$ and $G$ are directed graphs with colored edges, a mapping $\phi\colon V(H) \to V(G)$ is a *homomorphism* if $u \to v \in E(H)$ implies that $\phi(u) \to \phi(v)$

is an edge of $G$ of the same color as $uv$, and $\hom(H, G)$ denotes the number of homomorphisms from $H$ to $G$.

## 4.3 Dynamic data structure for induced subgraphs

In this section, we aim to design the data structure ISub as described in the introduction. More precisely, for any positive integer $k$, a fixed graph $H$ with edges colored by colors $\{1, \ldots, k\}$ and a class $\mathcal{C}$ of graphs, we design a data structure $\mathrm{ISub}_{H,k}(G)$ representing a graph $G \in \mathcal{C}$ with edges colored by $\{1, \ldots, k\}$, supporting the following operations.

- Determine $\mathrm{isub}(H, G)$.
- Change a color of an edge.
- Add an edge, i.e., transform $\mathrm{ISub}_{H,k}(G)$ to $\mathrm{ISub}_{H,k}(G + e)$, under the assumption that $G + \{e\}$ is in $\mathcal{C}$.
- Delete an edge, i.e., transform $\mathrm{ISub}_{H,k}(G)$ to $\mathrm{ISub}_{H,k}(G - e)$.

The complexity of the operations depends on $\mathcal{C}$ and is discussed in more detail in Subsection 4.3.5. To implement the data structure $\mathrm{ISub}_{H,k}(G)$, we first perform several standard transformations, reducing the problem to counting homomorphisms.

### 4.3.1 From induced subgraphs to subgraphs

The data structure $\mathrm{ISub}_{H,k}(G)$ is based on a data structure $\mathrm{Sub}_{H',k}(G)$, which can be used to determine the number of (not necessarily induced) subgraphs of $H'$ in $G$, i.e., the number $\mathrm{sub}(H', G)$. The relationship is based on the following claim.

Let $H(+, i, k)$ denote the set of all graphs which can be obtained from $H$ by adding exactly $i$ new edges and assigning them colors from $\{1, \ldots, k\}$.

**Lemma 8.**

$$\mathrm{isub}(H, G) = \sum_{i=0}^{\binom{|V(H)|}{2}-|E(H)|} (-1)^i \sum_{H' \in H(+,i,k)} \mathrm{sub}(H', G).$$

*Proof.* Let $\overline{E}$ be the set of all unordered pairs of vertices of $H$ that are not adjacent. For each pair $uv \in \overline{E}$ let $A_{uv}$ denote the set of all injective homomorphisms $\phi\colon V(H) \to V(G)$ such that $\phi(u)\phi(v) \in E(G)$. Observe that

$$\mathrm{isub}(H, G) = \mathrm{sub}(H, G) - \left| \bigcup_{uv \in \overline{E}} A_{uv} \right|$$

and that for $1 \leq i \leq |\overline{E}|$,

$$\sum_{H' \in H(+,i,k)} \mathrm{sub}(H', G) = \sum_{X \subseteq \overline{E}, |X|=i} \left| \bigcap_{uv \in X} A_{uv} \right|.$$

The claim of the lemma follows by the principle of inclusion and exclusion. □

The data structure $\mathrm{ISub}_{H,k}(G)$ consists of the collection of the data structures $\mathrm{Sub}_{H',k}(G)$ for all $H' \in \bigcup_{i=0}^{|\overline{E}|} H(+,i,k)$. The additions, removals and recolorings of edges of $G$ are performed in all of the data structures, and $\mathrm{isub}(H, G)$ is determined from the queries for $\mathrm{sub}(H', G)$ using the formula from Lemma 8. The complexity of each operation with $\mathrm{ISub}_{H,k}(G)$ is thus at most $2^{|V(H)|^2} = O(1)$ times the complexity of the corresponding operation with $\mathrm{Sub}_{H',k}(G)$ for some graph $H'$ with $|V(H')| = |V(H)|$.

### 4.3.2 From subgraphs to homomorphisms

Next, we aim to base the data structure $\mathrm{Sub}_{H,k}(G)$ on a data structure $\mathrm{Hom}_{H',k}(G)$, which counts the number $\hom(H', G)$ of homomorphisms from $H'$ to $G$. Furthermore, we want to restrict our attention to the case that $H'$ is connected.

Consider a graph $H$ with colored edges, and let $P$ be a partition of $V(H)$ such that

- each element of $P$ induces an independent set in $H$, and
- for every $p_1, p_2 \in P$, $u, u' \in p_1$ and $v, v' \in p_2$, if both $uv$ and $u'v'$ are edges of $H$, then $uv$ and $u'v'$ have the same color.

Let $H'$ be the graph obtained from $H$ by identifying the vertices in each part of $P$ and suppressing the parallel edges. We say that $H'$ is a *projection* of $H$. Let $\mathcal{H}^p$ denote the set of all projections $H'$ of $H$.

**Lemma 9.** *For every graph $H$ with colored edges, there exist integer coefficients $\alpha_{H'}$ such that for every graph $G$ with colored edges,*

$$\mathrm{sub}(H, G) = \sum_{H' \in \mathcal{H}^p} \alpha_{H'} \hom(H', G).$$

*Proof.* Let $\phi\colon V(H) \to V(G)$ be a homomorphism. Note that $P = \{\phi^{-1}(v) : v \in \mathrm{dom}(\phi)\}$ is a partition of $V(H)$ that gives rise to a projection $H'$ of $H$, and $H'$ appears as a subgraph in $G$. Conversely, if a projection $H'$ of $H$ (given by a partition $P$ of $V(H)$) is a subgraph of $G$, then it corresponds to a unique homomorphism from $H$ to $G$ that maps all vertices of each element of $P$ to the image of the corresponding vertex of $H'$.

This bijective correspondence shows that

$$\hom(H, G) = \sum_{H' \in \mathcal{H}^p} \mathrm{sub}(H', G).$$

Equivalently,

$$\mathrm{sub}(H,G) = \mathrm{hom}(H,G) - \sum_{H' \in \mathcal{H}^p \setminus \{H\}} \mathrm{sub}(H',G). \tag{4.1}$$

We prove Lemma 9 by induction. Assume that the claim is true for all graphs with fewer vertices than $H$. In particular, for every $H' \in \mathcal{H}^p$ other than $H$, there exist coefficients $\alpha_{H''}^{H'}$ such that

$$\mathrm{sub}(H',G) = \sum_{H'' \in \mathcal{H}'^p} \alpha_{H''}^{H'} \mathrm{hom}(H'',G)$$

for every graph $G$. Note that $\mathcal{H}'^p \subseteq \mathcal{H}^p \setminus \{H\}$. Therefore, Lemma 9 follows from (4.1) by setting $\alpha_H = 1$ and

$$\alpha_{H''} = - \sum_{H' \in \mathcal{H}^p \setminus \{H\}, H'' \in \mathcal{H}'^p} \alpha_{H''}^{H'}$$

for every $H'' \in \mathcal{H}^p \setminus \{H\}$. □

A similar trick allows us to deal with disconnected graphs.

**Observation 10.** *Let $H_1$ and $H_2$ be two graphs. For the disjoint union $H_1 \cup H_2$ it holds that*

$$\mathrm{hom}(H_1 \cup H_2, G) = \mathrm{hom}(H_1,G) \cdot \mathrm{hom}(H_2,G).$$

In the following subsection, we design a data structure $\mathrm{Hom}_{H,k}(G)$ for a connected graph $H$ with edges colored by $\{1,\ldots,k\}$, which counts the number $\mathrm{hom}(H,G)$ of homomorphisms from $H$ to $G$, and allows additions, removals and recolorings of edges in $G$.

The data structure $\mathrm{Sub}_{H,k}(G)$ consists of the collection of data structures $\mathrm{Hom}_{H',k}(G)$ for all connected components of projections of $H$. Edge additions, removals and recolorings in $G$ are performed in all these structures. The number $\mathrm{sub}(H,G)$ is determined from the queries to the structures according to the formula following from Lemmata 9 and 10.

The number of projections of $H$ and their components is bounded by a function of $H$, which we consider to be a constant. Therefore, the complexity of operations with $\mathrm{Sub}_{H,k}(G)$ is the same up to a constant multiplicative factor as the complexity of operations with $\mathrm{Hom}_{H',k}(G)$ with $|V(H')| \leq |V(H)|$.

### 4.3.3 Augmented graphs

In order to implement the data structure $\mathrm{Hom}_{H,k}(G)$, we use fraternal augmentations. Essentially, we would like to find a bijection between homomorphisms from $H$ to $G$ and between homomorphisms from all possible $h$-th augmentations of $H$ to an $h$-th augmentation of $G$, where $h = \binom{|V(H)|}{2} - 2$. However, it turns out that we need to be a bit more careful.

For a graph $F$ with edges colored by colors $\{1,\ldots,k\}$, we define the color of an edge $(u,v)$ of a $t$-th augmentation of $F$ to be the same as the color of $uv$ if $uv \in E(F)$, and to be 0 otherwise (i.e., we introduce a new color for the edges added through the fraternal augmentation). If $F'$ and $F''$ are directed graphs

with edges colored by colors $\{0, 1, \ldots, k\}$, we say that $F''$ is *obtained from* $F'$ *by recoloring zeros* if $F'$ and $F''$ differ only in the colors of edges whose color in $F'$ is 0.

**Lemma 11.** *Let $H$ and $G$ be graphs with edges colored by $\{1, \ldots, k\}$ and let $h \geq 0$ be an integer. Let $\phi\colon V(H) \to V(G)$ be a homomorphism and let $G'$ be an $h$-th augmentation of $G$. There exists a graph $H'$ obtained from an $h$-th augmentation of $H$ by recoloring zeros, such that for every edge $(u, v) \in E(H')$,*

- *if $\phi(u) \neq \phi(v)$, then $(\phi(u), \phi(v)) \in E(G')$, and $(u, v)$ has the same color as $(\phi(u), \phi(v))$; and,*
- *if $\phi(u) = \phi(v)$, then the color of $(u, v)$ is 0.*

*Proof.* We prove the claim by the induction on $h$. If $h = 0$, we let $H'$ be the orientation of $H$ such that each edge $uv \in E(H)$ is oriented towards $v$ if $(\phi(u), \phi(v)) \in G'$ and towards $u$ otherwise (i.e., if $(\phi(v), \phi(u)) \in G'$), with the colors of the edges of $H'$ matching the colors of the corresponding edges of $H$.

Therefore, suppose that $h > 0$. Let $G_1$ be an $(h-1)$-th augmentation of $G$ such that $G'$ is a fraternal augmentation of $G_1$. By induction hypothesis, there exists a directed graph $H_1$ obtained from an $(h-1)$-th augmentation $H$ by recoloring zeros, satisfying the outcome of the lemma.

Let $u$ and $v$ be vertices forming a fork in $H_1$, such that $\phi(u) \neq \phi(v)$. If $(\phi(u), \phi(v)) \in E(G_1)$ or $(\phi(v), \phi(u)) \in E(G_1)$, then we choose the orientation and the color of the edge $uv$ in $H'$ correspondingly. Otherwise, consider a vertex $w$ such that $(u, w), (v, w) \in E(H_1)$, and note that since $\phi(u)$ is not adjacent to $\phi(v)$ in $G_1$, the induction hypothesis implies that $\phi(u) \neq \phi(w) \neq \phi(v)$ and that $(\phi(u), \phi(w)), (\phi(v), \phi(w)) \in E(G_1)$. It follows that $\phi(u)$ and $\phi(v)$ form a fork in $G_1$, and thus $(\phi(u), \phi(v)) \in E(G)$ or $(\phi(v), \phi(u)) \in E(G)$. We choose the orientation of the edge $uv$ in $H'$ correspondingly, and color it by 0.

Finally, for each pair $u, v \in V(H_1)$ forming a fork in $H_1$ such that $\phi(u) = \phi(v)$, we choose an orientation of $uv$ in $H'$ arbitrarily and assign it color 0. Observe that the fraternal augmentation $H'$ of $H_1$ and its coloring satisfy the outcome of Lemma 11 as required. Furthermore, the choices of colors and orientations of edges of $H'$ that are not mapped to a single vertex are uniquely determined by the conditions of the lemma. □

Lemma 11 inspires the following definition. Let $F'$ be a directed graph with edges colored by $\{0, 1, \ldots, k\}$. Let $P$ be a partition of vertices of $F'$ such that

- for every $p \in P$, the subgraph of $F'$ induced by $p$ is connected and contains only edges colored by 0; and
- if $p_1, p_2 \in P$ are distinct, $u, u' \in p_1$, $v, v' \in p_2$ and $(u, v)$ is an edge, then $(v', u')$ is not an edge, and if $(u', v')$ is an edge, then it has the same color as $(u, v)$.

Let $F''$ be the directed graph with edges colored by $\{0, 1, \ldots, k\}$, such that $V(F'') = P$ and $(p_1, p_2) \in E(F'')$ if and only if $(v_1, v_2) \in E(F')$ for some $v_1 \in p_1$ and $v_2 \in p_2$; and in this case, $(p_1, p_2)$ and $(v_1, v_2)$ have the same color. That is,

$F''$ is obtained from $F'$ by identifying the vertices in each part of $P$ and suppressing the parallel edges and loops, and we also remember which vertices of $F'$ correspond to each vertex of $F''$. We say that $F''$ is a 0-*contraction* of $F'$.

We aim to find a bijection between the homomorphisms from an undirected graph $H$ to an undirected graph $G$ and the homomorphisms from all possible 0-contractions of augmentations of $H$ to a fixed augmentation of $G$. We will need the following uniqueness result.

**Lemma 12.** *Let $H$ be a graph with edges colored by $\{1,\dots,k\}$ and let $h = \binom{|V(H)|}{2} - 2$. Let $G'$ be a directed graph with edges colored by $\{0,1,\dots,k\}$, such that each vertex of $G'$ is contained in a loop with color 0, but $G'$ has no other loops or parallel edges. Let $H_1$ and $H_2$ be graphs obtained from $h$-th augmentations of $H$ by recoloring zeros, such that there exists $\phi\colon V(H) \to V(G')$ which is a homomorphism both from $H_1$ and from $H_2$ to $G'$. Let $P_0$ be the partition of $V(H)$ such that two vertices $u, v \in V(H)$ belong to the same part in $P$ if and only if $\phi(u) = \phi(v)$. For $i \in \{1,2\}$, let $P_i$ be the partition of $V(H)$ such that each $p \in P_i$ is the vertex set of a connected component of the subgraph of $H_i$ induced by vertices in some part $p' \in P_0$. Let $H_i'$ be the 0-contraction of $H_i$ corresponding to $P_i$. Then $H_1' = H_2'$.*

*Proof.* Before proceeding with the proof, let us remark that the assumption that $\phi$ is a homomorphism from $H_i$ to $G'$ ensures that the conditions on the partition $P_i$ from the definition of a 0-contraction are satisfied. Furthermore, $H_1' = H_2'$ implies $P_1 = P_2$.

Suppose that $F$ is a directed graph with vertex set $V(H)$ and with edges colored by $\{0,1,\dots,k\}$ such that $\phi$ is a homomorphism from $F$ to $G'$. Let $P(F)$ be the partition of $V(H)$ such that each $p \in P(F)$ is the vertex set of a connected component of the subgraph of $F$ induced by vertices in some part $p' \in P_0$.

Note that both $H_1$ and $H_2$ are elder graphs. Let $H_1^0$ be an orientation of $H$ and let $H_1^0$, $H_1^1$, …, $H_1^k$ be a sequence of directed graphs with edges colored by $\{0,1,\dots,k\}$, such that $H_1 = H_1^k$ and for $1 \le i \le k$, the graph $H_1^i$ is obtained from $H_1^{i-1}$ by adding an edge joining two vertices forming a fork.

We are going to construct a sequence $H_2^0$, $H_2^1$, …, $H_2^k$, where

- $H_2^0$ is an orientation of $H$ and $H_2^i \subseteq H_2$ for $0 \le i \le k$,
- $H_2^i$ is obtained from $H_2^{i-1}$ by repeatedly adding edges joining two vertices forming a fork, for $1 \le i \le k$, and
- $P(H_1^i) = P(H_2^i)$ and the 0-contractions of $H_1^i$ and $H_2^i$ corresponding to this partition are identical, for $0 \le i \le k$.

We set $H_2^0 = H_1^0$. Since $\phi$ is a homomorphism from both $H_1$ and $H_2$ to $G'$ and no edge of $H$ is colored by 0, we have $H_2^0 \subseteq H_2$, hence $H_2^0$ satisfies the required properties.

Suppose now that $1 \le i \le k$ and that we have already constructed $H_2^{i-1}$. Let $u, v, w \in V(H)$ be the vertices such that $u$ and $v$ are not adjacent in $H_1^{i-1}$, $(u,w), (v,w) \in E(H_1^{i-1})$ and $(u,v) \in E(H_1^i)$. Let $P_{i-1} = P(H_1^{i-1}) = P(H_2^{i-1})$. If $u$ and $v$ belong to the same part of $P_{i-1}$, then note that $P(H_1^i) = P_{i-1}$ and the 0-contractions of $H_1^{i-1}$ and $H_1^i$ corresponding to this partition are identical. Therefore, we can set $H_2^i = H_2^{i-1}$.

Suppose now that $u$ and $v$ belong to different parts $p_u, p_v \in P_{i-1}$. Let $p_w \in P_{i-1}$ be the part containing $w$. Note that $p_w \neq p_u$, as otherwise $H_1$ would contain both edges $(u, v)$ and $(v, w)$ with $\phi(u) = \phi(w) \neq \phi(v)$, contrary to the assumption that $\phi$ is a homomorphism from $H_1$ to $G'$. If $p_w = p_v$, then note that $P(H_1^i) = P_{i-1}$ and the corresponding 0-contractions of $H_1^{i-1}$ and $H_1^i$ are identical, hence we can set $H_2^i = H_2^{i-1}$.

Therefore, we can assume that $p_u \neq p_w \neq p_v$. In this case we construct $H_2^i$ by initially setting $H_2^i = H_2^{i-1}$ and then adding edges as described in the rest of this paragraph. Since there exist edges between $w$ and $u$ and $v$, we also have $\phi(u) \neq \phi(w) \neq \phi(v)$. Since the 0-contractions of $H_1^{i-1}$ and $H_2^{i-1}$ corresponding to $P_{i-1}$ are identical, there exist vertices $u' \in p_u$, $v' \in p_v$ and $w_1, w_2 \in p_w$ with $(u', w_1), (v', w_2) \in E(H_2^{i-1})$. Let $w_1 = x_1, x_2, \ldots, x_t = w_2$ be an induced path between $w_1$ and $w_2$ in the underlying undirected graph of the subgraph of $H_2^{i-1}$ induced by $p_w$. Let us also set $x_0 = u'$ and $x_{t+1} = v'$. Since $(x_0, x_1), (x_{t+1}, x_t) \in E(H_2^{i-1})$, observe that there exists $j$ (with $1 \leq j \leq t$) such that $(x_{j-1}, x_j), (x_{j+1}, x_j) \in E(H_2^{i-1})$. Since $H_2$ is an elder graph, we have that either $(x_{j-1}, x_{j+1})$ or $(x_{j+1}, x_{j-1})$ is an edge of $H_2$. We add this edge to $H_2^i$ and consider the path $x_0, x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_{t+1}$. Let us note that if $j = 1$ and $t > 1$, then we added the edge $(x_0, x_2)$, as we have $x_0 \in p_u$, $x_2 \in p_w$, there already exists an edge from $p_u$ to $p_w$, and $\phi$ is a homomorphism from $H_2$ to $G$ that maps all vertices of $p_u$ to $\phi(u)$ and all vertices of $p_w$ to $\phi(w)$. A symmetric argument holds in the case that $j = t > 1$. Therefore, we can repeat this procedure until an edge between $u'$ and $v'$ is added.

If $\phi(u) \neq \phi(v)$, then the last added edge is $(u', v')$, since $\phi$ is a homomorphism from both $H_1$ and $H_2$ to $G$. Furthermore, all other added edges were inside $p_w$, or between $p_u$ and $p_w$, or between $p_v$ and $p_w$, hence $P(H_2^i) = P_{i-1} = P(H_1^i)$. We conclude that the corresponding 0-contractions are identical as required. If $\phi(u) = \phi(v)$, then we similarly conclude that both $P(H_1^i)$ and $P(H_2^i)$ are obtained from $P_{i-1}$ by merging $p_u$ and $p_v$, and that the corresponding 0-contractions of $H_1^i$ and $H_2^i$ are identical.

Therefore, there exists the sequence $H_2^0, \ldots, H_2^k$ with the required properties. Since $H_2^k \subseteq H_2$, we have that $P(H_1) = P(H_2^k)$ is a refinement of $P(H_2)$. By switching the role of $H_1$ and $H_2$ in the argument, we conclude that $P(H_2)$ is a refinement of $P(H_1)$. Therefore, $P(H_1) = P(H_2)$. Since $H_1 = H_1^k$ and $H_2^k$ have the same 0-contraction corresponding to this partition, it follows that $H_1' \subseteq H_2'$. By symmetry, we have $H_2' \subseteq H_1'$, and thus $H_1' = H_2'$. □

Lemmata 11 and 12 enable us to express the number of homomorphisms for undirected graphs in the terms of the homomorphisms of their augmentations. For a graph $H$ with edges colored by $\{1, \ldots, k\}$, let $\mathcal{H}^e$ denote the set of all 0-contractions of graphs obtained by recoloring zeros from $h$-th augmentations of $H$, where $h = \binom{|V(H)|}{2} - 2$.

**Lemma 13.** *Let $H$ and $G$ be graphs with edges colored by $\{1, \ldots, k\}$ and let $h = \binom{|V(H)|}{2} - 2$. If $G'$ is an $h$-th augmentation of $G$, then*

$$\hom(H, G) = \sum_{H' \in \mathcal{H}^e} \hom(H', G').$$

*Proof.* Consider a homomorphism $\phi$ from $H$ to $G$. Let $H_0$ be a graph obtained from an $h$-th augmentation of $H$ by recoloring zeros such that $H_0$ satisfies the outcome of Lemma 11. Let $P_0$ be the partition of $V(H_0)$ such that two vertices $u, v \in V(H_0)$ are in the same part if and only if $\phi(u) = \phi(v)$. Note that every $p \in P_0$ induces a subgraph of $H_0$ whose edges have color 0. Let $P$ be the refinement of $P_0$ such that every $p' \in P$ is the vertex set of a connected component of the underlying undirected graph of the subgraph induced in $H_0$ by some $p \in P_0$. Let $H'$ be the 0-contraction corresponding to the partition $P$ (which satisfies the assumptions from the definition of a 0-contraction since $\phi$ is a homomorphism). Let $\phi' \colon V(H') \to V(G)$ be the mapping such that $\phi'(p) = \phi(v)$ for every $p \in V(H')$ and $v \in p$. Observe that $\phi'$ is a homomorphism from $H'$ to $G'$. This defines a mapping $\Phi(\phi) = (H', \phi')$ which assigns a graph $H' \in \mathcal{H}^e$ and a homomorphism $\phi' \colon V(H') \to V(G')$ to each homomorphism $\phi \colon V(H) \to V(G)$.

We need to prove that $\Phi$ is a bijection. Note that if $\Phi(\phi) = (H', \phi')$, then for each $v \in V(H)$, we have $\phi(v) = \phi'(p)$, where $p$ is the vertex of $H'$ such that $v \in p$. Therefore, $\Phi$ is an injection, and it suffices to argue that $\Phi$ is surjective.

Consider arbitrary $H' \in \mathcal{H}^e$ and a homomorphism $\phi'$ from $H'$ to $G'$. Since $H' \in \mathcal{H}^e$, there exists a graph $H'_0$ obtained from an $h$-th augmentation of $H$ by recoloring zeros such that $H'$ is a 0-contraction of $H'_0$. Let $\phi \colon V(H) \to V(G)$ be the mapping defined by $\phi(v) = \phi'(p)$, where $p$ is the vertex of $H'$ such that $v \in p$. Note that if $uv \in E(H)$, then $(u, v) \in E(H'_0)$ or $(v, u) \in E(H'_0)$ and this edge has nonzero color, and thus there exist distinct vertices $p_u, p_v \in V(H')$ with $u \in p_u$ and $v \in p_v$ such that an orientation of $p_u p_v$ is an edge of $H'$ of the same color. Since $\phi'$ is a homomorphism, we conclude that $\phi'(p_u)\phi'(p_v) = \phi(u)\phi(v)$ is an edge of $G$ of the same color. It follows that $\phi$ is a homomorphism from $H$ to $G$.

We need to prove that $\Phi(\phi) = (H', \phi')$. Suppose that $\Phi(\phi) = (H'', \phi'')$. Let $H_0$ be the graph from the definition of $\Phi(\phi)$. Since $\phi$ is a homomorphism from both $H_0$ and $H'_0$ to the graph obtained from $G'$ by adding loops of color 0 to each vertex, Lemma 12 implies that $H'' = H'$. Since the homomorphism from $H'$ to $G'$ is uniquely determined by $\phi$, it also follows that $\phi'' = \phi'$, as required. Therefore, $\Phi$ is indeed a bijection, and the equality of the lemma follows. □

Furthermore, taking 0-contractions preserves elderness.

**Lemma 14.** *If $H$ is an elder graph with edges colored by $\{0, 1, \ldots, k\}$ and $H'$ is a 0-contraction of $H$, then $H'$ is an elder graph.*

*Proof.* Let $P$ be a partition of $V(H)$ that gives rise to $H'$. Suppose that vertices $u', v' \in V(H')$ form a fork, i.e., they are non-adjacent and there exists a vertex $w' \in V(H')$ with $(u', w'), (v', w') \in E(H')$.

By the definition of a 0-contraction, there exist vertices $u \in u'$, $v \in v'$ and $w_1, w_2 \in w'$ such that $(u, w_1), (v, w_2) \in E(H)$. Furthermore, the underlying undirected graph of the subgraph of $H$ induced by $w'$ is connected, hence it contains a path $Q = x_1 x_2 \ldots x_t$ with $x_1 = w_1$ and $x_t = w_2$. Let us choose the vertices $w_1$, $w_2$ and the path $Q$ so that the length of $Q$ is minimal.

Since $u'$ and $v'$ are not adjacent, it follows that $u$ and $v$ are not adjacent, and since $H$ is an elder graph, we conclude that $w_1 \neq w_2$. Note that $Q$ is an induced path. Since $H$ is an elder graph, it follows that for every $i$ with $2 \leq i \leq t-1$, either $(x_{i-1}, x_i) \notin E(H)$ or $(x_{i+1}, x_i) \notin E(H)$. This implies that if $(x_1, x_2) \in E(H)$, then $(x_i, x_{i+1}) \in E(H)$ for $1 \leq i \leq t-1$. Therefore, either $(x_2, w_1) \in E(H)$ or

$(x_{t-1}, w_2) \in E(H)$. By symmetry, we assume the former. Since $H$ is an elder graph, this implies that either $(u, x_2) \in E(H)$ or $(x_2, u) \in E(H)$. Since $H'$ is a 0-contraction of $H$ arising from the partition $P$, $(u, w_1) \in E(H)$ and $w_1, x_2 \in P_w$, it follows that the edge between $u$ and $x_2$ cannot be oriented towards $u$, and thus $(u, x_2) \in E(H)$. However, the path between $x_2$ and $w_2$ is shorter than $Q$. Since $Q$ was chosen so that its length is minimal, this is a contradiction. □

In the following subsection, we design a data structure $\mathrm{AHom}_{(H',T'),k,D}(G')$ for an elder vineyard $(H', T')$ and a directed graph $G'$ of maximum in-degree at most $D$, where both $H'$ and $G'$ have edges colored by $\{0, 1, \ldots, k\}$. The data structure AHom counts the number $\hom(H', G')$ of homomorphisms from $H'$ to $G'$ and allows additions, removals, reorientations and recolorings of edges in $G'$.

The data structure $\mathrm{Hom}_{H,k}(G)$ consists of

- an $h$-th augmentation $G'$ of $G$ (where $h = \binom{|V(H)|}{2} - 2$) maintained as described in Theorem 6, and
- the collection of data structures $\mathrm{AHom}_{(H',T'),k,D}(G')$ for each $H' \in \mathcal{H}^e$, where $D$ is the bound from Theorem 6 for the class of graphs containing $G$ and $T'$ is an outbranching in $H'$ such that $(H', T')$ is an elder vineyard (which exists by Lemma 7).

Note that $|\mathcal{H}^e|$ is bounded by a function of $H$ and $k$ only, and thus its size is constant.

Edge additions and removals are first performed in the data structure for the $h$-th augmentation $G'$ of $G$. Each addition results in $O(D \log^{h+1} |V(G)|)$ changes in $G'$, each removal results in $O(D)$ such changes (amortized). A recoloring in $G$ only affects one edge of $G'$. In all the cases, the changes of $G'$ are performed in all the AHom substructures. The number $\hom(H, G)$ is determined by summing the results of the queries to these substructures, as follows from Lemma 13.

### 4.3.4 Homomorphisms of elder graphs

In this subsection, we describe the data structure AHom, thus finishing the design of the data structure for subgraphs.

Let $(H, T)$ be an elder vineyard. A *clan* is a subset $C$ of vertices of $H$ such that $N_1^+(C) = C$ and the subgraph $T'$ of $T$ induced by $C$ is an outbranching. Let $r(C)$ denote the root of this outbranching $T'$. The *ghosts* of a clan $C$ are the vertices $N_1^-(C) \setminus C$.

**Lemma 15.** *Let $(H, T)$ be an elder vineyard.*

1. *For every $v \in V(H)$, the set $N_\infty^+(v)$ is a clan.*
2. *The ghosts of a clan $C$ are exactly the vertices in $N_1^-(r(C)) \setminus C$.*
3. *All ghosts of a clan $C$ are on the path from $r(H)$ to $r(C)$ in $T$.*

*Proof.* Let us prove the claims separately:

1. Let $C = N^+_\infty(v)$. Clearly, $N^+_1(C) = C$. Note that the subgraph $H[C]$ of $H$ induced by $C$ is connected. If the subgraph of $T$ induced by $C$ is not an outbranching, then it contains two components $T_1$ and $T_2$ joined by an edge of $H[C]$. Observe that no directed path in $T$ contains a vertex both in $T_1$ and $T_2$. This contradicts the assumption that $(H, T)$ is a vineyard.

2. Suppose that $v$ is a ghost of $C$, i.e., there exists an edge $(v, w) \in E(H)$ for some $w \in C$. Let $w$ be such a vertex whose distance from $r(C)$ in $T$ is minimal. If $w \neq r(C)$, then consider the in-neighbor $z$ of $w$ in $T$. Since $H$ is an elder graph, $v$ and $z$ are adjacent in $H$. Since $v$ does not belong to $C$, we have $(z, v) \notin E(H)$, and thus $(v, z) \in E(H)$. However, the distance from $r(C)$ to $z$ in $T$ is smaller than the distance to $w$, which is a contradiction. Therefore, we have $w = r(C)$ as required.

3. This follows from the definition of vineyard.

□

The *extended clan* $C^*$ for a clan $C$ is obtained from the subgraph of $H$ induced by $C$ and its ghosts by removing the edges joining pairs of ghosts.

Let $G$ be a directed graph and let $(H, T)$ be an elder vineyard, where the edges of $G$ and $H$ are colored by colors $\{0, 1, \ldots, k\}$. let $C$ be a clan with ghosts $g_1, \ldots, g_m$ listed in the increasing order by their distance from $r(C)$ in $T$ and let $v$ and $w_1, \ldots, w_m$ be (not necessarily distinct) vertices of $G$. Note that $g_1$ is the in-neighbor of $r(C)$ in $T$. Let $\hom_{(H,T)}(C, v, w_1, \ldots, w_m, G)$ denote the number of homomorphisms from $C^*$ to $G$ such that $r(C)$ maps to $v$ and $g_1, \ldots, g_m$ map to $w_1, \ldots, w_m$ in order. Let $\hom((H, T), G, v)$ denote the number of homomorphisms from $H$ to $G$ such that $r(T)$ maps to $v$.

**Theorem 16.** *Let $(H, T)$ be an elder vineyard with edges colored by $\{0, 1, \ldots, k\}$ and let $D$ be an integer. There exists a data structure $\mathrm{AHom}_{(H,T),k,D}(G)$ representing a directed graph $G$ with edges colored by $\{0, 1, \ldots, k\}$ and maximum in-degree at most $D$ supporting the following operations in $O(D^{|V(H)|^2})$ time.*

1. *Addition of an edge $e$ to $G$ such that the maximum indegree of $G + e$ is at most $D$.*

2. *Reorientation of an edge in $G$ such that the maximum indegree of the resulting graph is at most $D$.*

3. *Removal or recoloring of an edge.*

*The data structure can be used to determine $\hom((H, T), G, v)$ for a vertex $v \in V(G)$, as well as $\hom(H, G)$, in $O(1)$. The data structure can be built in time $O(D^{|V(H)|^2+1}|V(G)|)$ and has space complexity $O(D^{|V(H)|}|V(G)|)$.*

*Proof.* We store the following information:

- For each clan $C \neq V(H)$ with $m$ ghosts and each $m$-tuple of vertices $w_1, \ldots, w_m$ of $G$ we record the number

$$S(C, w_1, \ldots, w_m) = \sum_{v \in N^+_1(w_1)} \hom_{(H,T)}(C, v, w_1, \ldots, w_m),$$

that is the number of homomorphisms of $C^*$ to $G$ such that the ghosts of $C$ map to $w_1, \ldots, w_m$ and $r(C)$ maps to some outneighbor $v$ of $w_1$.

- For each $v \in V(G)$, the number $\hom((H,T),G,v)$.
- The sum $\hom(H,G)$ of these numbers over all vertices of $G$.

The number $S(C, w_1, \ldots, w_m)$ is only stored for those combinations of $C$ and $w_1, \ldots, w_m$ for that it is non-zero. The values are stored in a hash table (see e.g. [20] for implementation details), so that they can be accessed in a constant time. By Lemma 15, if $\hom_{(H,T)}(C, v, w_1, \ldots, w_m)$ is non-zero, then $w_1$, ..., $w_m$ are in-neighbors of $v$ in $G$. Since the maximum indegree of $G$ is at most $D$, each vertex $v$ contributes at most $D^{|V(H)|}$ non-zero values (and each of the numbers is smaller or equal to $|V(G)|^{|V(H)|}$), thus the space necessary for the storage is $O(D^{|V(H)|}|V(G)|)$. Queries can be performed in a constant time by returning the stored information.

The addition of an edge $(x,y)$ to $G$ is implemented as follows. We process the clans of $(H,T)$ in the decreasing order of size, i.e., when we use the information stored for the smaller clans, it still refers to the graph $G$ without the new edge. Let us consider a clan $C \neq V(H)$ with ghosts $g_1, \ldots, g_m$. For each non-empty set $X$ of edges of $C^*$ which have the same color as $(x,y)$, we are going to find all vertices $v$ and $w_1, \ldots, w_m$ such that there exists a homomorphism of $C^*$ mapping $r(C)$ to $v$ and the ghosts of $C$ to $w_1, \ldots, w_m$ which maps precisely the edges of $X$ to $(x,y)$. We will also determine the numbers of such homomorphisms, and decrease the number $S(C, w_1, \ldots, w_m)$ by this amount. Note that the number of choices of $X$ is constant (bounded by a function of $H$).

Consider now a fixed set $X$. Let $M$ be the set of vertices $z \in V(C^*)$ such that there exists a directed path in $C^*$ from $z$ to the head of an edge of $X$. Note that $r(C)$ and all ghosts of $C$ belong to $M$. Let $C_1$, ..., $C_t$ be the vertex sets of connected components of $C^* - M$, and observe that they are clans. Now, let $\mathcal{F}$ be the set of all homomorphisms from the subgraph of $C^*$ induced by $M$ to $G + (x,y)$ such that exactly the edges of $X$ are mapped to $(x,y)$. Note that if $z$ is an image of a vertex of $M$ in such a homomorphism, then $G$ contains a directed path from $z$ to $y$ of length at most $|V(H)|$, thus there are only $O(D^{|V(H)|})$ vertices of $G$ to that $M$ can map, and consequently only $O(D^{|V(H)|^2})$ choices for the homomorphisms. Each such choice fixes the image of $r(C)$ as well as all the ghosts.

Consider $\phi \in \mathcal{F}$. We need to determine in how many ways $\phi$ extends to a homomorphism of $C^*$ that maps no further edges to $(x,y)$ (this number is then added to the value $S(C, \phi(g_1), \ldots, \phi(g_m))$). Note that for $1 \leq i \leq t$, the ghosts of $C_i$ are contained in $M$, and thus their images are fixed by the choice of $\phi$. Therefore, if ${g^i}_1, \ldots, {g^i}_{m_i}$ are the ghosts of $C_i$, then the number of the homomorphisms extending $\phi$ is

$$\prod_{i=1}^{t} S(C_i, \phi(g_1^i), \ldots, \phi(g_{m_i}^i)).$$

Here, we use the fact that the values $S(C_i, \ldots)$ were not updated yet, and thus in the homomorphisms that we count, no other edge maps to $(x,y)$. These products can be determined in a constant time.

The values $\hom((H,T),G,v)$ are updated similarly, before the values $S(C,\ldots)$ are updated. The changes in the values of $\hom((H,T),G,v)$ are also propagated to the stored value of $\hom(H,G)$. The complexity of the update is given by the number of choices of partial homomorphisms $\mathcal{F}$, i.e., $O(D^{|V(H)|^2})$.

Edge removal works in the same manner, except that the information is subtracted in the end, and that the clans are processed in the opposite direction, i.e., starting from the inclusion-wise smallest clans, so that the values for the graph without the edge are used in the computations.

Change of the orientation of an edge or its recoloring can be implemented as subsequent deletion and addition. The data structure can be initialized by adding edges one by one, starting with the data structure for an empty graph $G$ whose initialization is trivial. □

### 4.3.5 Induced subgraphs

The data structure $\mathrm{ISub}_{H,k}(G)$ consists essentially of the data structure for maintaining the $h$-th augmentation $G'$ of $G$ (where $h = \binom{|V(H)|}{2} - 2$) and of a constant (bounded by a function of $k$ and $H$) number of data structures AHom, to that we have to propagate all the changes in $G'$. Therefore, if $D$ is the bound from the data structure from Theorem 6, then the data structure $\mathrm{ISub}_{H,k}(G)$ has the following complexities (amortized).

- Edge addition: $O(D^{|V(H)|^2+1}\log^{\binom{|V(H)|}{2}-1}|V(G)|)$.
- Edge removal: $O(D^{|V(H)|^2+1})$.
- Edge recoloring: $O(D^{|V(H)|^2})$.
- Initialization: $O(D^{|V(H)|^2+1}|V(G)|+t)$.
- Space: $O(D^{|V(H)|^2}|V(G)|)$.

For dense graphs, the bound on $D$ is too large for the data structure to be useful. However, if $G$ is kept within some class $\mathcal{C}$ of graphs with bounded expansion, then the function $h(n,r)$ from Theorem 6 can be chosen to be constant, and we obtain $D$ constant. Therefore, when applied to such a class of graphs, the complexities are as follows.

- Edge addition: $O(\log^{\binom{|V(H)|}{2}-1}|V(G)|)$.
- Edge removal and recoloring: $O(1)$.
- Initialization: $O(|V(G)|)$.
- Space: $O(|V(G)|)$.

Similarly, if $\mathcal{C}$ is nowhere-dense, then the function $h(n,r)$ is $O(n^{\varepsilon'})$ for any fixed $r$ and any $\varepsilon'>0$. Therefore, given any $\varepsilon>0$, we can choose $\varepsilon'$ to be less than $\varepsilon/(|V(H)|^2+1)$ and the complexities of the data structure are as follows.

- Edge addition, removal and recoloring: $O(|V(G)|^{\varepsilon})$.
- Initialization and space: $O(|V(G)|^{1+\varepsilon})$.

## 4.4 Extensions

Although we have for simplicity formulated the data structure ISub for a graph $G$ with a fixed vertex set, there is no problem with adding or removing isolated vertices to/from $G$ in a constant time.

One can ask about a number of possible extensions to the data structure ISub. Can we allow directed edges? Or colors of vertices? Or hyperedges? All these can be expressed as relational structures, recall the definition from 2.4. We work with colors here, note that colors of vertices and edges can be represented by additional unary and binary relations, respectively.

We define $|S|$ as $|V(S)| + \sum_{R \in \sigma} |R^S|$, and for a structure $S$ work with its incidence graph $G^i_S$. Note that for structures with bounded expansion, the maximum average degree of corresponding Gaifman graph $G_S$ is bounded by a constant ($|V(S)|^\varepsilon$ for every $\varepsilon > 0$, respectively). Consequently, the number of cliques in $G_S$ of size bounded by the maximum arity of a relation symbol of $S$ is $O(|V(S)|)$ ($O(|V(S)|)^{1+\varepsilon}$ for every $\varepsilon > 0$, respectively), see [111]. It follows that we have $|S| = O(|V(G)|)$ ($O(|V(S)|)^{1+\varepsilon}$ for every $\varepsilon > 0$, respectively).

As promised in 2.4, we prove that Gaifman graphs and incidence graphs define the same density.

**Lemma 17.** *If a class of structures $\mathcal{S}$ has bounded expansion (is nowhere-dense), then the class $\mathcal{S}^i = \{G^i_S;\ S \in \mathcal{S}\}$ has bounded expansion (is nowhere dense, respectively).*

*Proof.* We present the proof for bounded expansion. The argument for the nowhere-dense case is analogical.

For $t > 0$ and a class $\mathcal{C}$ of graphs, let $\mathcal{C} \widetilde{\nabla} t$ denote the set of all graphs $G$ such that there exists a graph $G' \in \mathcal{C}$ and a graph obtained from $G$ by subdividing each edge at most $t$ times is a subgraph of $G'$. As was shown in [35], $\mathcal{C}$ has bounded expansion if and only if for every $t > 0$, there exists a constant $c_t$ such that all graphs in $\mathcal{C} \widetilde{\nabla} t$ have average degree at most $c_t$.

Suppose that there exists $t$ such that we can find arbitrarily dense graphs in $\mathcal{S}^i \widetilde{\nabla} t$. Since $\mathcal{S}^i \widetilde{\nabla} t$ is closed on subgraphs, it also contains graphs of arbitrarily large minimum degree. Let $k$ be the maximum arity of the symbols in the dictionary of $S$ and let $H$ be a graph in $\mathcal{S}^i \widetilde{\nabla} t$ with minimum degree at least $k+2$, and let $S \in \mathcal{S}$ be a relational structure such that a graph $H'$ obtained from $H$ by subdividing each edge at most $t$ times appears as a subgraph of $G^i_S$.

The branching vertices of $H'$ in $G^i_S$ must correspond to the vertices of $V(S)$, since all other vertices of $G^i_S$ have degree at most $k+1$. However, for every path of length two in $G^i_S$ between two vertices $u$ and $v$ corresponding to vertices of $V(S)$, there is an edge in $G_S$ between $u$ and $v$. We conclude that $H \in \{G_S;\ S \in \mathcal{S}\} \widetilde{\nabla} \lceil t/2 \rceil$. Therefore, $\{G_S;\ S \in \mathcal{S}\} \widetilde{\nabla} \lceil t/2 \rceil$ would contain graphs of arbitrarily large minimum degree, contradicting the assumption that $\mathcal{S}$ has bounded expansion. □

To count the number of appearances of a fixed relational structure $S_0$ as an induced substructure of a relational structure $S$, it suffices to count the number of appearances of $G^i_{S_0}$ in $G^i_S$ as an induced subgraph (assuming that every vertex

of $S_0$ belongs to at least one relation; the case that $S_0$ contains isolated vertices can be dealt with by introducing a new unary relation satisfied for all vertices). A change (addition or removal of a tuple to/from a relation) in $S$ results in only a constant number of changes in $G_S^i$, thus ISub can be used to represent relational structures through this transformation.

A seemingly more general question is testing existential first order properties, i.e., properties which can be defined by closed first-order formulas using only non-negated existential quantifiers. Such a formula $\phi$ can be considered to be in the disjunctive normal form, i.e.

$$\phi = \bigvee_{i=1}^{t} \exists x_1, \ldots, \exists x_l \phi_i,$$

where each $\phi_i$ is a conjunction of finitely many terms of the form $R(x_{i_1}, \ldots, x_{i_k})$ or $\neg R(x_{i_1}, \ldots, x_{i_k})$ for a relation $R \in \sigma \cup \{=\}$ of arity $k$. For example, existence of an induced subgraph, subgraph or homomorphism from a graph $H$ of a bounded size can be expressed this way, by a formula having one variable for each vertex of $H$ and describing the required adjacency, non-adjacency and non-equality relations between them.

In order to decide whether a structure $S$ satisfies the given formula $\phi$, we need to find a set of witnessing vertices $x_1, \ldots, x_l$ which satisfies the subformula $\phi_i$ for some $1 \leq i \leq t$. For every such $\phi_i$ there is only a finite number of structures $S_i$ on at most $l$ vertices which satisfy $\phi_i$. Hence, it suffices to check whether one of these structures is an induced substructure of $S$. Therefore, using the data structure ISub, we can decide arbitrary existential first order properties with a bounded number of variables on classes of structures with bounded expansion (or nowhere-dense), within the same time bounds.

# Chapter 5

# Conclusion

In this thesis we have gathered important and influential results from the areas of parameterised complexity, structural graph theory and model theory; and showed rich interplay between these areas. Although the development has been rich, in many places our knowledge is not full and many gaps remain to be filled.

We have contributed with two new data structures, showing how results and tools can be lifted from their static variants and applied to dynamic problems. We conclude with open problems tightly related to our work.

As for the Subgraph problem, a natural question is whether one can design a fully dynamic data structure to decide properties expressible in First Order Logic on graphs with bounded expansion. For this purpose, it would be convenient to be able to maintain low tree-depth colourings of [80], which however appears to be difficult.

Possibly a much easier problem is the following. We have described a dynamic data structure that enables us to count the number of appearances of $H$ as an induced subgraph of $G$, for graphs from a class with bounded expansion. If this number is non-zero, can we find such an appearance? Getting this from our data structure is not entirely trivial, due to the use of the principle of inclusion and exclusion.

By the result Courcelle, any property expressible in $\mathbb{MSO}_2$ can be tested for graphs of bounded tree-width in linear time. Can one design a dynamic data structure for this problem? It is not even clear how to maintain a tree decomposition of bounded width dynamically.

As for the Dynamic treedepth decomposition, there is a natural question whether we can provide a dynamic decomposition for other graph classes, say, with bounded treewidth.

# Bibliography


[1] K. A. Abrahamson, R. G. Downey, and M. R. Fellows, *Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues*, Annals of pure and applied logic, 73 (1995), pp. 235–276.

[2] G. Adel'son-Vel'skii and E. Landis, *An algorithm for the organization of information*, Doklady Akademii Nauk SSSR, 146 (1962).

[3] J. Alber, H. Fan, M. R. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, and U. Stege, *A refined search tree technique for dominating set on planar graphs*, Journal of Computer and System Sciences, 71 (2005), pp. 385–405.

[4] N. Alon and J. H. Spencer, *The probabilistic method*, Wiley, 2004.

[5] K. Appel and W. Haken, *Every planar map is four colorable*, Bulletin of the American mathematical Society, 82 (1976), pp. 711–712.

[6] D. Archdeacon and P. Huneke, *A kuratowski theorem for nonorientable surfaces*, Journal of Combinatorial Theory, Series B, 46 (1989), pp. 173–231.

[7] S. Arnborg, *Efficient algorithms for combinatorial problems on graphs with bounded decomposability–a survey*, BIT Numerical Mathematics, 25 (1985), pp. 1–23.

[8] S. Arnborg, D. G. Corneil, and A. Proskurowski, *Complexity of finding embeddings in a k-tree*, SIAM Journal on Algebraic Discrete Methods, 8 (1987), pp. 277–284.

[9] S. Arora and B. Barak, *Computational complexity: a modern approach*, Cambridge University Press, 2009.

[10] L. W. Beineke and R. E. Pippert, *The number of labeled k-dimensional trees*, Journal of Combinatorial Theory, 6 (1969), pp. 200–205.

[11] H. L. Bodlaender, *A tourist guide through treewidth*, Technical report RUU-CS, 92 (1993).

[12] ——, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM Journal on computing, 25 (1996), pp. 1305–1317.

[13] ——, *Discovering treewidth*, in SOFSEM 2005: Theory and Practice of Computer Science, Springer, 2005, pp. 1–16.

[14] ——, *Treewidth: characterizations, applications, and computations*, in Graph-theoretic concepts in computer science, Springer, 2006, pp. 1–14.

[15] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks, *Approximating treewidth, pathwidth, frontsize, and shortest elimination tree*, Journal of Algorithms, 18 (1995), pp. 238–255.

[16] H. L. Bodlaender, A. Grigoriev, and A. M. Koster, *Treewidth lower bounds with brambles*, Algorithmica, 51 (2008), pp. 81–98.

[17] G. Brodal and R. Fagerberg, *Dynamic representations of sparse graphs*, in Algorithms and Data Structures, vol. 1663 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 1999, pp. 773–773.

[18] M. Chrobak and D. Eppstein, *Planar orientations with low out-degree and compaction of adjacency matrices*, Theoretical Computer Science, 86 (1991), pp. 243–266.

[19] S. A. Cook, *The complexity of theorem-proving procedures*, in Proceedings of the third annual ACM symposium on Theory of computing, ACM, 1971, pp. 151–158.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, The MIT Press, 2009.

[21] B. Courcelle, *The monadic second-order logic of graphs. I. Recognizable sets of finite graphs*, Information and computation, 85 (1990), pp. 12–75.

[22] B. Courcelle, J. A. Makowsky, and U. Rotics, *Linear time solvable optimization problems on graphs of bounded clique-width*, Theory of Computing Systems, 33 (2000), pp. 125–150.

[23] ——, *On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic*, Discrete Applied Mathematics, 108 (2001), pp. 23–52.

[24] B. Courcelle and S. Olariu, *Upper bounds to the clique width of graphs*, Discrete Applied Mathematics, 101 (2000), pp. 77–114.

[25] A. Dawar, M. Grohe, and S. Kreutzer, *Locally excluding a minor*, in Logic in Computer Science, 2007. LICS 2007. 22nd Annual IEEE Symposium on, IEEE, 2007, pp. 270–279.

[26] E. D. Demaine, D. Harmon, J. Iacono, and M. Patrascu, *Dynamic optimality-almost*, SIAM Journal on Computing, 37 (2007), pp. 240–251.

[27] M. DeVos, G. Ding, B. Oporowski, D. P. Sanders, B. Reed, P. Seymour, and D. Vertigan, *Excluding any graph as a minor allows a low tree-width 2-coloring*, Journal of Combinatorial Theory, Series B, 91 (2004), pp. 25–41.

[28] R. Diestel, *Graph theory*, Springer, Heidelberg, 1997.

[29] G. Ding, B. Oporowski, D. P. Sanders, and D. Vertigan, *Surfaces, tree-width, clique-minors, and partitions*, Journal of Combinatorial Theory, Series B, 79 (2000), pp. 221–246.

[30] G. A. Dirac, *A property of 4-chromatic graphs and some remarks on critical graphs*, Journal of the London Mathematical Society, 1 (1952), pp. 85–92.

[31] R. G. Downey, *The birth and early years of parameterized complexity*, in The Multivariate Algorithmic Revolution and Beyond, Springer, 2012, pp. 17–38.

[32] R. G. Downey and M. R. Fellows, *Fixed-parameter tractability and completeness I: Basic results*, SIAM Journal on Computing, 24 (1995), pp. 873–921.

[33] ——, *Parameterized computational feasibility*, in Feasible Mathematics II, Birkhauser, 1995, pp. 219–244.

[34] ——, *Parameterized Complexity*, Springer Verlag, 1999.

[35] Z. Dvořák, *On forbidden subdivision characterizations of graph classes*, European Journal of Combinatorics, 29 (2008), pp. 1321–1332.

[36] Z. Dvořák, A. C. Giannopoulou, and D. M. Thilikos, *Forbidden graphs for tree-depth*, European Journal of Combinatorics, 33 (2012), pp. 969–979.

[37] Z. Dvorak, D. Král, and R. Thomas, *Deciding first-order properties for sparse graphs*, in Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on, IEEE, 2010, pp. 133–142.

[38] Z. Dvorak, M. Kupec, and V. Tuma, *Dynamic data structure for tree-depth decomposition*, arXiv preprint arXiv:1307.2863, (2013).

[39] Z. Dvořák and V. Tůma, *A dynamic data structure for counting subgraphs in sparse graphs*, in Algorithms and Data Structures Symposium, WADS 2013, London (Ont.), Proceedings, Springer, 2013. To appear.

[40] H.-D. Ebbinghaus, *Mathematical logic*, Springer, 1994.

[41] H.-D. Ebbinghaus and J. Flum, *Finite model theory*, Springer, 2005.

[42] F. Eisenbrand and F. Grandoni, *On the complexity of fixed parameter clique and dominating set*, Theoretical Computer Science, 326 (2004), pp. 57–67.

[43] J. Ellis, H. Fan, and M. Fellows, *The dominating set problem is fixed parameter tractable for graphs of bounded genus*, in Algorithm TheorySWAT 2002, Springer, 2002, pp. 180–189.

[44] D. Eppstein, *Subgraph isomorphism in planar graphs and related problems*, J. Graph Algorithms Appl., 3 (1999), pp. 1–27.

[45] D. Eppstein, M. Goodrich, D. Strash, and L. Trott, *Extended dynamic subgraph statistics using h-index parameterized data structures*, in Proceedings of the 4th international conference on Combinatorial optimization and applications - Volume Part I, COCOA'10, Springer-Verlag, 2010, pp. 128–141.

[46] D. Eppstein and E. S. Spiro, *The h-index of a graph and its application to dynamic subgraph statistics*, in Algorithms and Data Structures, Springer, 2009, pp. 278–289.

[47] R. Fagin, *Generalized first-order spectra and polynomial-time recognizable sets*, in Complexity of Computation, SIAM-AMS Proceedings,, vol. Vol. 7, 1974.

[48] J. Flum and M. Grohe, *Fixed-parameter tractability, definability, and model-checking*, SIAM Journal on Computing, 31 (2001), pp. 113–145.

[49] ——, *Parameterized complexity theory*, vol. 3, Springer Heidelberg, 2006.

[50] F. V. Fomin, P. A. Golovach, D. Lokshtanov, and S. Saurabh, *Intractability of clique-width parameterizations*, SIAM Journal on Computing, 39 (2010), pp. 1941–1956.

[51] M. Frick and M. Grohe, *Deciding first-order properties of locally tree-decomposable structures*, Journal of the ACM (JACM), 48 (2001), pp. 1184–1206.

[52] ——, *The complexity of first-order and monadic second-order logic revisited*, in Logic in Computer Science, 2002. Proceedings. 17th Annual IEEE Symposium on, IEEE, 2002, pp. 215–224.

[53] J. Gajarský and P. Hliněný, *Faster Deciding MSO Properties of Trees of Fixed Height, and Some Consequences*, in IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012), D. D'Souza, T. Kavitha, and J. Radhakrishnan, eds., 2012, pp. 112–123.

[54] R. Ganian, P. Hliněný, A. Langer, J. Obdržálek, P. Rossmanith, and S. Sikdar, *Lower bounds on the complexity of MSO1 model-checking*, arXiv preprint arXiv:1109.5804, (2011).

[55] M. R. Garey and D. S. Johnson, *Computers and intractability*, Freeman New York, 1979.

[56] M. Grohe, *Algorithmic metatheorems*, in Graph-Theoretic Concepts in Computer Science, Springer, 2008.

[57] M. Grohe and S. Kreutzer, *Methods for algorithmic metatheorems*, Model Theoretic Methods in Finite Combinatorics, 558 (2011), pp. 181–206.

[58] P. J. Heawood, *Map-colour theorem*, 24 (1890), pp. 332–339.

[59] P. Hliněný, S.-i. Oum, D. Seese, and G. Gottlob, *Width parameters beyond tree-width and their applications*, The computer journal, 51 (2008), pp. 326–362.

[60] W. Hodges, *Shorter model theory*, Cambridge university press, 1997.

[61] N. Immerman, *Descriptive complexity*, Springer, 1999.

[62] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, Springer, 1972.

[63] T. Kloks, D. Kratsch, and H. Müller, *Finding and counting small induced subgraphs efficiently*, Information Processing Letters, 74 (2000), pp. 115–121.

[64] A. V. Kostochka, *The minimum Hadwiger number for graphs with a given mean degree of vertices*, Metody Diskret. Analiz, 38 (1982), pp. 37–58.

[65] M. Koutecký, *Solving hard problems on neighborhood diversity*, in Masters thesis, Charles University in Prague, 2013.

[66] S. Kreutzer, *Algorithmic metatheorems*, in Parameterized and Exact Computation, Springer, 2008.

[67] ——, *On the parameterised intractability of monadic second-order logic*, in Computer Science Logic, Springer, 2009, pp. 348–363.

[68] S. Kreutzer and S. Tazari, *Lower bounds for the complexity of monadic second-order logic*, in Logic in Computer Science (LICS), 2010 25th Annual IEEE Symposium on, IEEE, 2010, pp. 189–198.

[69] J. B. Kruskal, *Well-quasi-ordering, the tree theorem, and Vazsonyi's conjecture*, Transactions of the American Mathematical Society, 95 (1960), pp. 210–225.

[70] C. Kuratowski, *Sur le probleme des courbes gauches en topologie*, Fundamenta mathematicae, 15 (1930), pp. 271–283.

[71] M. Lampis, *Algorithmic meta-theorems for restrictions of treewidth*, Algorithmica, 64 (2012), pp. 19–37.

[72] M. Lampis, *Model checking lower bounds for simple graphs*, CoRR, abs/1302.4266 (2013).

[73] L. Libkin, *Elements of finite model theory*, Springer, 2004.

[74] L. Lovász, *Graph minor theory*, Bulletin of the American Mathematical Society, 43 (2006), pp. 75–86.

[75] J. A. Makowsky and J. Mariño, *Tree-width and the monadic quantifier hierarchy*, Theoretical computer science, 303 (2003), pp. 157–170.

[76] T. Milenkoviæ and N. Pržulj, *Uncovering biological network function via graphlet degree signatures*, Cancer informatics, 6 (2008), p. 257.

[77] B. Mohar, *A linear time algorithm for embedding graphs in an arbitrary surface*, SIAM Journal on Discrete Mathematics, 12 (1999), pp. 6–26.

[78] B. Mohar and C. Thomassen, *Graphs on surfaces*, Johns Hopkins University Press Baltimore, 2001.

[79] J. Nešetřil and P. Ossona de Mendez, *Colorings and homomorphisms of minor closed classes*, in Discrete and Computational Geometry, B. Aronov, S. Basu, J. Pach, and M. Sharir, eds., vol. 25 of Algorithms and Combinatorics, Springer Berlin Heidelberg, 2003, pp. 651–664.

[80] ——, *Linear time low tree-width partitions and algorithmic consequences*, in Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, STOC '06, ACM, 2006, pp. 391–400.

[81] ——, *Tree-depth, subgraph coloring and homomorphism bounds*, European Journal of Combinatorics, 27 (2006), pp. 1022–1041.

[82] ——, *Grad and classes with bounded expansion I. Decompositions*, European Journal of Combinatorics, 29 (2008), pp. 760–776.

[83] ——, *First order properties on nowhere dense structures*, J. Symbolic Logic, 75 (2010), pp. 868–887.

[84] ——, *On nowhere dense graphs*, European J. Combin., 32 (2011), pp. 600–617.

[85] ——, *Sparsity: Graphs, Structures, and Algorithms*, vol. 28, Springer, 2012.

[86] J. Nešetřil and S. Poljak, *Complexity of the subgraph problem*, Comment. Math. Univ. Carol., 26 (1985), pp. 415–420.

[87] R. Niedermeier, *Invitation to fixed-parameter algorithms*, Oxford University Press Oxford, 2006.

[88] M. H. Overmars, *The design of dynamic data structures*, Springer, 1983.

[89] G. Ringel and J. Youngs, *Solution of the Heawood map-coloring problem*, Proceedings of the National Academy of Sciences of the United States of America, 60 (1968), p. 438.

[90] N. Robertson, D. Sanders, P. Seymour, and R. Thomas, *The four-colour theorem*, Journal of Combinatorial Theory, Series B, 70 (1997), pp. 2–44.

[91] N. Robertson and P. D. Seymour, *Graph minors. I. Excluding a forest*, Journal of Combinatorial Theory, Series B, 35 (1983), pp. 39–61.

[92] ——, *Graph minors. III. Planar tree-width*, Journal of Combinatorial Theory, Series B, 36 (1984), pp. 49–64.

[93] ——, *Graph minors – a survey*, Surveys in combinatorics, 103 (1985), pp. 153–171.

[94] ——, *Graph minors. II. Algorithmic aspects of tree-width*, Journal of algorithms, 7 (1986), pp. 309–322.

[95] ——, *Graph minors. V. Excluding a planar graph*, Journal of Combinatorial Theory, Series B, 41 (1986), pp. 92–114.

[96] ——, *Graph minors. VIII. A Kuratowski theorem for general surfaces*, Journal of Combinatorial Theory, series B, 48 (1990), pp. 255–288.

[97] ——, *Graph minors. X. Obstructions to tree-decomposition*, Journal of Combinatorial Theory, Series B, 52 (1991), pp. 153–190.

[98] ——, *Graph minors. XIII. The disjoint paths problem*, Journal of Combinatorial Theory, Series B, 63 (1995), pp. 65–110.

[99] ——, *Graph minors. XVII. Taming a vortex*, Journal of Combinatorial Theory, Series B, 77 (1999), pp. 162–210.

[100] G. Robins and M. Morris, *Advances in exponential random graph ($p^*$) models*, Social Networks, 29 (2007), pp. 169–172.

[101] H. Röhrig, *Tree decomposition: A feasibility study*, in Masters thesis, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1998.

[102] P. D. Seymour and R. Thomas, *Graph searching and a min-max theorem for tree-width*, Journal of Combinatorial Theory, Series B, 58 (1993), pp. 22–33.

[103] D. D. Sleator and R. Endre Tarjan, *A data structure for dynamic trees*, Journal of computer and system sciences, 26 (1983), pp. 362–391.

[104] D. D. Sleator and R. E. Tarjan, *Self-adjusting binary search trees*, Journal of the ACM (JACM), 32 (1985), pp. 652–686.

[105] T. Tao, *Structure and randomness in combinatorics*, in Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on, IEEE, 2007, pp. 3–15.

[106] S. Tarkowski, *On the comparability of dendrites*, Bull. Acad. Polon. Sci. Math. Astronom. Phys, 8 (1960), pp. 39–41.

[107] A. Thomason, *An extremal function for contractions of graphs*, Math. Proc. Cambridge Philos. Soc, 95 (1984), pp. 261–265.

[108] C. Thomassen, *The graph genus problem is NP-complete*, Journal of Algorithms, 10 (1989), pp. 568–576.

[109] C. Thomassen, *Five-coloring graphs on the torus*, J. Combin. Theory, Ser. B, 62 (1994), pp. 11–33.

[110] M. Y. Vardi, *The complexity of relational query languages*, in Proceedings of the fourteenth annual ACM symposium on Theory of computing, ACM, 1982, pp. 137–146.

[111] D. Wood, *On the maximum number of cliques in a graph*, Graph. Comb., 23 (2007), pp. 337–352.

# List of Figures

# List of Tables