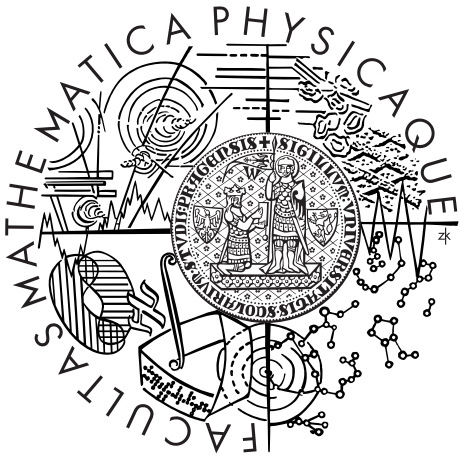
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Martin Koutecký

Solving hard problems on Neighborhood Diversity

Department of Applied Mathematics

Supervisor of the master thesis: doc. Mgr. Petr Kolman Ph.D.

Study programme: Computer science

Specialization: Discrete models and algorithms

Prague 2013

First I would like to thank my supervisor doc. Mgr. Petr Kolman Ph.D. for his valuable help and for always being supportive throughout more than the four years that I have known him.

Next I would like to thank my friend and colleague Mgr. Dušan Knop for providing me with much needed feedback and inspiration.

I am also glad for all new ideas and suggestions made by Michael Lampis in our email conversations. I would also like to thank Daniel Dadush who helped clarify some of his research over email. Finally I want to thank Mgr. Jakub Gajarský for the help he provided over email.

Last but not least I am immensely grateful to my parents for their help and support. The example they set for me is the reason I got this far.

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 Prague on April 12, 2013 Martin Koutecký

Název práce: Obtížné problémy vzhledem k parametru různorodost sousedství

Autor: Martin Koutecký

Katedra: Katedra aplikované matematiky

Vedoucí bakalářské práce: doc. Mgr. Petr Kolman Ph.D.

Abstrakt: Parametrizovaná složitost je oblast teoretické informatiky zabývající se výpočetní složitostí problémů měřenou nikoliv pouze délkou vstupu, ale i nějakým jeho parametrem. „Různorodost sousedství“ je nový strukturální parametr grafu, který je atraktivní především proto, že pro grafy s pevnou různorodostí sousedství se stávají efektivně řešitelnými i některé problémy, jež zůstávají těžké pro jiné parametry s různorodostí sousedství neporovnatelnými. V této práci nově ukazujeme efektivní řešitelnost vzhledem k různorodosti sousedství pro tři problémy těžké vzhledem ke stromové šířce. To tvoří hlavní část této práce a jedná se o náš vlastní výzkum. Dále pak práce obsahuje přehled dalších zajímavých problémů a také shrnutí současného stavu v oblasti parametrů pro řídké a husté grafy.

Klíčová slova: Parametrizovaná složitost, husté grafy, různorodost sousedství

Title: Hard problems on Neighborhood Diversity

Author: Martin Koutecký

Department: Department of Applied Mathematics

Supervisor: doc. Mgr. Petr Kolman, Ph.D.

Abstract: Parameterized complexity is a part of computer science dealing with the computational complexity of problems measured not only by the length of their input but also some parameter of the input. Neighborhood diversity is a recently introduced parameter describing a certain structure of a graph. This parameter is attractive for research especially because some problems which are hard with respect to other parameters that are incomparable with neighborhood diversity become fixed-parameter tractable with respect to neighborhood diversity. In this thesis we show fixed-parameter tractability for three problems that are hard with respect to treewidth. This constitutes the main part of this thesis and it is our original work. Next it contains an overview of other interesting problems and also a survey of the state of the art in the area of parameters for sparse and dense graphs.

Keywords: Parameterized complexity, dense graphs, neighborhood diversity

Contents

1 Introduction

The main focus of computational complexity theory has been for at least 40 years on the study of the relationship between the running time of an algorithm solving some problem and the problem size, like the number of vertices and edges of a graph. This approach led us to the definitions of complexity classes such as P (polynomial time), NP (nondeterministic polynomial time) and EXP (exponential time), and especially the notion of an NP-complete problem, which are all truly foundational for the field of computer science as we know it now. Here we of course have in mind primarily the seminal work done by Cook [16], Karp [51] and Levin [62].

But over time it started to be more and more apparent that a finer approach needs to be taken when dealing with hard problems. In practice the inputs we are feeding to our algorithms are often times nowhere close to random; there might be some underlying structure in our instances that actually allows us to create algorithms that work efficiently for these instances, even though the problem still stays hard in general. In other words, we find that by looking at just the problem size when analyzing the worst-case time complexity we are missing a significant part of the picture, and that it is beneficial to split the input into two parts which are examined separately.

The ideas above are formalized by the quickly growing field of *Parameterized Complexity* [25, 32]. In parameterized complexity we get a *parameter* as a special part of the input; quite often it is just an integer representing the solution size, but generally it can an arbitrary piece of information that gives us more information about the input (its structure etc.). To be clear we are not saying that the problem becomes easier to solve, rather that we measure its complexity in two dimension instead of one, and we contain the hardness to just one of the dimension. In this sense parameterized complexity can be viewed as a tool to rigorously capture *where exactly* lies the complexity of a given problem. Fellows [27] mentions specific cases where parameterized complexity was used to explain the success of some heuristics – these heuristics were implicitly relying on a certain structure of the input graph, which parameterized complexity explicitly captures and analyzes as the parameter.

A classical example of the first case (i.e., a parameter that is simply the solution size) is a parameterized approach to the (NP-complete) Vertex Cover problem – if along with the input graph we are given the size of the solution k, there is an algorithm which solves the problem in time $O(1.271^k + kn)$ [15]. This is still exponential in the solution size, but if that is fixed and we can treat it as a constant, the algorithm runs in time linear with the graph size. We should add that this approach is not attractive just in theory, but the algorithm mentioned above "has been implemented and is quite practical for n of unlimited size and k

up to around 400" (quote from an overview by Fellows [27]).

A classical example of a more complicated parameter is *treewidth*, which can be roughly said to be a measure of how close a graph is to a tree. This parameter arose as a part of the *Graph Minor Project* undertaken by Robertson and Seymour [72, 73]. Just like many hard problems are efficiently solvable on trees by dynamic programming, it was found that the same technique can be used on graphs whose treewidth is bounded by a constant. Moreover, treewidth provides a characterization of some naturally occuring graph classes, such as series-parallel graphs.

Treewidth is also characteristic for describing the development of metaalgorithmic theorems, most famously Courcelle's theorem [17]. The idea of metaalgorithmic theorems is that instead of describing many similar algorithms for similar problems, there is a way to capture a common essence of these, which is done by proving that some logical language is efficiently decidable when the parameter is fixed. This provides great intuition when studying said parameter.

The progress done on treewidth led to the exploration of many other structural parameters, which are usually studied from two perspectives. The first is trying to answer questions such as "How restrictive is this parameter? How many graphs display this structure? What graph classes are captured well by this parameter?" The second perspective is concerned with questions like "Which problems become tractable when we use this parameter? What logical languages become efficiently decidable when this parameter is fixed?"

Instead of asking the first few questions mentioned above (like "What graph classes are captured well by this parameter?") we can take a somewhat "dual" approach and ask "What are good parameters for this class of graphs?" A very vague way to distinguish between graphs is to look at how "dense" or "sparse" a graph is. It can be argued that treewidth is a successful parameter for handling sparse graphs (although we are by no means at the end of the road yet). On the other hand, with dense graphs the situation is still very much open. It is this question, "What is a good parameter for dense graphs?", where this thesis is trying to make some contribution.

The structure of the thesis is following: in Chapter 2 we give a brief overview of parameterized complexity and **provide some basic definitions**. We also define the parameters we will be working with or referencing, namely treewidth, cliquewidth, neighborhood diversity and finite type. Lastly we will describe some techniques from the field of integer programming which we will be using later.

In Chapter 3 we give an **overview of problems** we were interested in, explain the motivation in choosing specifically these problems, and show how to solve some of them efficiently when neighborhood diversity is fixed. This constitutes the main body of this work and also contains our original solutions to three interesting problems.

Finally in Chapter 4 we discuss the current situation in the field of **parameters**, first for sparse and then for dense graphs.

2 Foundations

Before we can describe our results and go into a deeper discussion of the various parameters, we need to define some of the basic terms from the theory of parameterized complexity, and four parameters that are most important for the following chapter: treewidth, cliquewidth, neighborhood diversity and finite type. We assume the reader is familiar with the basics of graph theory, algorithms and computational complexity. For these we refer to the excellent books by Diestel [23] and Dasgupta, Papadimitriou and Vazirani [22].

2.1 Parameterized complexity

The first systematic work on parameterized complexity was done by Downey and Fellows [25], a quick high-level overview can be found in a paper by Fellows [27] and the probably most up-to-date thorough treatment of the topic is given by Flum and Grohe in their book [32].

As we said earlier the goal of parameterized complexity is to provide a classification of NP-hard problems on a finer scale than in the classical setting, where the complexity of a problem is only measured by the length of the input. The two extremes in how successful we can be when analyzing a problem from a parameterized perspective are characterized by the classes FPT and XP. Roughly said, a problem falls into the class FPT if an algorithm running in time $O(f(k)n^{O(1)})$ exists; note that the degree of the polynomial does not depend on the parameter. On the other hand, if the degree of the polynomial *does* depend on the parameter and the algorithm runs in time $O(f(k)n^{g(k)})$, the problem is in the class XP. Let us now define those classes formally.

Definition 1. A problem is said to be *fixed-parameter tractable* (or *FPT*) if an algorithm solving the problem in time $f(k)p(n)$ exists, where f is any computable function depending solely on the parameter k, and $p(n)$ is a polynomial in the total input length n. This is the same as saying that the problem *belongs to the class FPT*.

Definition 2. A problem is said to *belong to the class XP* if an algorithm solving the problem in time $f(k)n^{g(k)}$ exists, where both f and g are computable functions depending solely on the parameter k.

Curiously enough, a hierarchy of complexity classes appears between FPT and XP, called the W-hierarchy. The definition of the class W[i] is not difficult, but for our purposes it is unnecessary to deal with it in detail. For us it suffices to say that the class FPT is the analogue of the class P in classical complexity, W[1]-hardness is the analogue of NP-hardness and XP is the analogue of EXP.

Also just like P $\neq$ NP is usually accepted as a reasonable assumption, so is FPT $\neq$ W[1], and it is known that FPT = W[1] would imply that n variable 3-SAT can be solved in $2^{o(n)}$ time [13].

A simple way to describe the parameterized complexity world as we know it now are the following two inclusion sequences:

$$FPT \subset XP$$
$$FPT \subseteq W[1] \subseteq XP$$

2.1.1 Examples

We have already mentioned in the introduction that the Vertex Cover problem is FPT when parameterized by the solution size [13]. What is probably more interesting are some negative results. It is known that deciding both the k-Clique and the k-Independent Set problems is W[1]-hard when the solution size k is the parameter [25].

2.2 Parameters of interest

Now we would like to formally define four parameters that are interesting for us, two of them well known and established (treewidth and cliquewidth), the other two fairly new whose relevance is yet to be determined (neighborhood diversity and finite type). For completeness we also define the vertex cover number parameter.

We will postpone the discussion of how hard various computational problems are with respect to these parameters to Chapter 3 with the exception of metaalgorithmical results. With these we take interest in two aspects, first, what logic is efficiently decidable, and second, how does the function $f(k)$ (in the running time $O(f(k)n^{O(1)})$) look.

To answer the first question we need to know a bit about *Monadic Second Order* logic, or just MSO. We use this logical language to express properties of graphs which we then want to decide on those graphs, or in the words of finite model theory, we want to solve the model checking problem. As for how a MSO formula looks: it allows us to quantify over vertices and edges (just like *First Order* logic, or FO), but additionally (unlike FO) we can also quantify over *sets* of vertices and sometimes even sets of edges. The last distinction (if we can quantify over sets of edges or not) is precisely how MSO_1 and MSO_2 differ, and this difference is significant because generally MSO_2 is strictly stronger than MSO_1.

Observe that MSO_1 allows us to check for the existence of a k-coloring – we construct a formula with k existential vertex set quantifiers in the beginning and then with a FO formula check that first, these sets are a partition, and second, they are independent sets. In a similar way MSO_2 allows us to check for Hamiltonicity of a graph – we check for the existence of an edge set that is both a 2-factor of the graph, and is connected. Moreover, it can be proved that there is no MSO_1 formula checking the Hamiltonicity of a graph. This is done using techniques from Finite Model Theory, mainly Ehrenfeucht-Fräisé games. For more see Libkin's book [63].

As for the second question, "how does the function $f(k)$ look", we only look at two cases – either it is an exponential tower whose height depends in some way on the formula (its quantifier depth or its alternation depth, which is the number of general-existential quantifier alternations), or the height of the exponential tower is constant. Obviously the first scenario is catastrophic in the worst case for practical use and even though there are indicators suggesting the situation might not be so dire (see Chapter 4.1) the motivation to look for the second scenario is strong.

2.2.1 Treewidth

The first parameter we will define is treewidth. It can be considered a measure of how much a graph is "tree-like".

Definition 3 ([72, 73])**.** A *tree decomposition* of a graph $G = (V, E)$ is a pair $(T, \mathcal{X})$, where T is a tree, $\mathcal{X}$ is a collection of subsets $X \subseteq V$ called *bags* and every bag $X \in \mathcal{X}$ is associated with one *node* $t \in T$, such that two conditions are satisfied. First, every edge is contained within some bag (i.e., $\forall e = \{u, v\} \in E$: $\exists X \in \mathcal{X}$ with $\{u, v\} \subseteq X$) and second, every vertex $v \in V$ induces a connected subtree in T (i.e., $\forall v \in V$ the subgraph induced in T by bags for with $v \in X$ is connected).

The *width* of a tree decomposition is the size of the largest bag minus one. (This is so that the treewidth of trees is one.)

The *treewidth of a graph* G (or just $td(G)$) is the width of its thinnest decomposition.

The treewidth of a clique is its size plus one, because the whole clique has to be contained in one bag. A more interesting observation is that the treewidth of a $n \times n$ grid is at least $n + 1$. This implies that the treewidth of planar graphs is not bounded by a constant; on the other hand the famous planar separator theorem [64] implies that the treewidth of a planar graph will always be bounded by $O(\sqrt{n})$. For a more thorough treatment of treewidth see Diestel's book [23].

As for the metaalgorithmical properties of treewidth, it was shown already in 1990 by Courcelle that both MSO_1 and MSO_2 are FPT with respect to treewidth [17] and this result was quickly extended by Arnborg et al. [2] to optimization variants (i.e., not only decide the *existence* of vertex or edge sets but also find ones of optimal size). On the other hand, Frick and Grohe [35] showed in 2004 that unless P=NP, the MSO_1 model checking problem on graphs of bounded treewidth is not solvable for any elementary function *f*. A special consequence of this is that there also can not be an algorithm with $f(k)$ being an exponential tower of fixed height.

2.2.2 Cliquewidth

The second parameter we will define is cliquewidth.

Definition 4 ([20])**.** The *cliquewidth of a graph* G (or just $cw(G)$) is the minimum number of labels L needed to construct G by means of the following four operations. (Labels come from the set $\{1, \ldots, L\}$.)

1. Creation of a new vertex v with label i,

2. Disjoint union of two labeled graphs H_1 and H_2,

3. Joining by an edge every vertex labeled i to every vertex labeled j,

4. Renaming label i to label j.

It is not difficult to show that cliques of arbitrary size can be constructed using just two labels and so their cliquewidth is bounded, unlike their treewidth. On the other hand it is known [43] that bounded treewidth implies bounded cliquewidth.

The metaalgorithmical situation for cliquewidth evolved thusly. Shortly after the introduction of cliquewidth Courcelle, Makowski and Rotics [19] proved that MSO_1 is FPT for cliquewidth. On the other hand, MSO_2 is not because it is hard already on cliques (this is shown in the same paper). Also, the non-elementary lower bounds on the complexity of model checking from Frick and Grohe [35] we mentioned regarding treewidth apply here, too.

2.2.3 Neighborhood diversity

A third parameter we introduce is a generalization of vertex cover and comes from a recent paper by Lampis [58] but the terminology we decided to use is inspired by a master thesis by Gajarský [36] which also introduces the fourth parameter we want to deal with, namely finite type.

Definition 5 ([58])**.** The *neighborhood diversity of a graph* G (or just $nd(G)$) is the number of equivalency classes of the following equivalence: two vertices $u, v \in V$ are equivalent if they have the same neighborhoods except for possibly themselves, i.e. if $N(v) \setminus \{u\} = N(u) \setminus \{v\}$.

It might not obvious that the relation we used to define neighborhood diversity is an equivalence and we refer the reader to Lampis' original paper [58] for proof.

The definition given above is concise, but does not give much intuition about the structure of graphs of neighborhood diversity k. Observe that we can view such a graph like this: the vertices belonging to the same equivalence class form *bags* which are either a clique or an independent set, and every two bags either have a complete bipartite graph between them, or no edges at all.

A way of looking at the observation above is to say that all graphs G of neighborhood diversity k were constructed according to some *template* graph $G_{\mathcal{M}}$ on k vertices. This template graph determines which bags are joined by a complete bipartite graph and the only information missing for a complete description of G are the sizes of bags. It is easy to find the template graph $G_{\mathcal{M}}$ from the original G by simply contracting every bag into a single vertex. This leads us to the following definitions:

Definition 6 ([36])**.** For a graph G with $nd(G) = k$ we define its *metagraph* $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ to be a (possibly annotated) graph on k vertices which correspond to the equivalence classes of neighborhood diversity. There is an edge between two vertices from $V_{\mathcal{M}}$ if there is a complete bipartite graph between the corresponding equivalence classes. Also there is a loop on precisely those vertices from $V_{\mathcal{M}}$ whose vertices correspond to equivalence classes which form a clique.

We call the vertices of $G_{\mathcal{M}}$ *metavertices* and the edges *metaedges*. We call a metavertex which forms a clique a *complete* metavertex, and a metavertex which forms an independent set an *independent* metavertex.

We will use two ways to refer to the metavertices. If we mean just the vertices of the metagraph, we will use lowercase letters, such as $i, j \in V_{\mathcal{M}}$. If we need to work with the original vertices of G associated with said metavertex, we treat the metavertex as a set and use an uppercase V indexed by the metavertex, e.g. $|V_i|$ = number of vertices in the equivalence class i.

The annotation that is possibly attached to the metagraph will depend on the problem we are trying to solve. If only the graph and no additional information is on the input, there is only one piece of information on every metavertex, which is its *size* $|V_i|$.

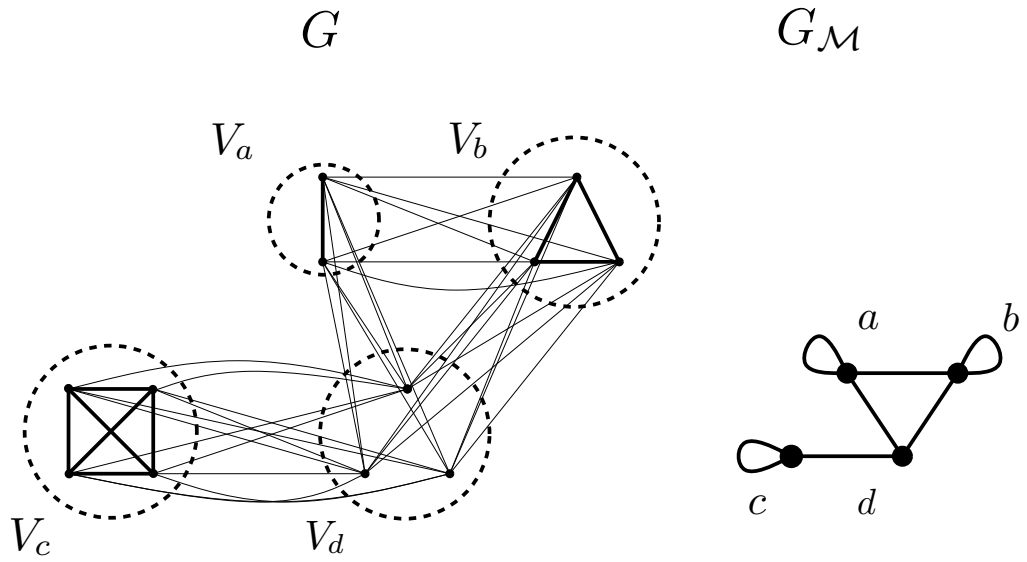


Figure 2.1: **An example of a graph with neighborhood diversity 4, and its metagraph.** On the left side a graph $G = (V, E)$ with $nd(G) = k$ and with neighborhood diversity equivalency classes V_a, V_b, V_c, V_d, the first three forming cliques and the last one forming an independent set. On the right side its metagraph $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ with $V_{\mathcal{M}} = \{a, b, c, d\}$ and $E_{\mathcal{M}} = \{ab, ad, bd, cd, aa, bb, cc\}$. The loops represent that the corresponding equivalency class forms a clique.

The metaalgorithmical situation of neighborhood diversity is much better when compared with cliquewidth, because it allows for MSO_1 model checking with f being only a double exponential [58]. In fact, this was the main contribution of Lampis' original paper. On the other hand the MSO_2 hardness result for cliquewidth carries over to neighborhood diversity too (because like with cliquewidth, cliques have bounded neighborhood diversity).

2.2.4 Finite type

In his thesis Gajarský [36] introduced a generalization of neighborhood diversity which is intuitively easy to grasp when we adopt the view that a graph is constructed according to some *template*, as we did in the previous section. The actual

definition of finite type in his thesis is somewhat technical (for understandable reasons) so we use a hopefully more approachable rephrasing of it.

The basic idea is to inductively iterate the process that is used to construct a graph of bounded neighborhood diversity. However, instead of having just two types of bags, cliques and independent sets, we allow them to be disjoint unions of arbitrarily many graphs of bounded neighborhood diversity. This is the "first level" of finite type, and the same process leads to higher "levels".

Definition 7. We define $\mathcal{M}_k$ to be the collection of all graphs (allowing loops) on k vertices; an element from this collection $M_k \in \mathcal{M}_k$ is a *metagraph*.

Definition 8 ([36])**.** We define *graphs of type* (l, k) and denote this class by $\mathcal{C}^{(l,k)}$. In this notation k is similar to the k from the definition of neighborhood diversity, and l is the "induction" level.

The lowest level is the class $\mathcal{C}^{(0,k)}$ which are precisely graphs of neighborhood diversity at most k. A graph from $\mathcal{C}^{(l,k)}$ is obtained by the following procedure:

1. Choose any metagraph $M_k \in \mathcal{M}_k$ as a template,
2. Replace every vertex of M_k by a disjoint union of arbitrarily many graphs from $\mathcal{C}^{(l-1,k)}$ (the class on the previous level),
3. Put complete bipartite graphs between bags (the disjoint unions created in the previous step) that are joined by an edge in the template metagraph M_k, and turn a bag into a clique if there is a loop on the corresponding metavertex in M_k.

A nice property of this parameter is that if there is an edge between any two vertices $u, v \in V$ of the original graph G, it is determined at just one level and from then on it never changes. The level where this is determined is the level where u and v belong to different metavertices of the same metagraph – if these metavertices are joined by a metaedge, u and v are adjacent in G. They could not have been joined earlier because then they belonged to different metagraphs and we only take disjoint unions of these. Moreover, they can not be joined later because they belong to the same metavertex and we do not add new edges inside these (except for the case where the metavertex has a loop, which is handled easily).

Gajarský does not mention much about the relationships between finite type and other parameters in his thesis, but in an email conversation he explained that it is not hard to find a clique decomposition of bounded width for graphs of bounded finite type. On the other hand finite type of paths is not bounded, but their cliquewidth is. This implies that finite type is a more restrictive parameter than cliquewidth.

The metaalgorithmical situation of finite type is similar to the one of neighborhood diversity: MSO_1 is FPT for finite type and the height of the exponential tower does not depend on the formula. This was the main contribution of Gajarský's thesis [36]. Obviously MSO_2 is hard on finite type since it is already hard on neighborhood diversity, which is a special case of finite type.

2.2.5 Vertex cover

For completeness we briefly introduce vertex cover as a parameter.

Definition 9. The *vertex cover number* of a graph G (or just $vc(G)$) is the size of the smallest set $C \subseteq V$ such that for every edge $uv \in E, u \in C$ or $v \in C$.

Lampis [58] proved that both MSO_1 and MSO_2 model checking is FPT on graphs with bounded vertex cover number and the function $f(k)$ in the complexity bounds of mentioned theorems are double exponential functions.

2.2.6 Relationships between the parameters

As we stated in the introduction there is typically a balance to be struck when we look at structural parameters – either the parameter permits a large portion of inputs and does not allow as many problems to become tractable, or vice versa. Some parameters are generalizations of others in the sense that one is bounded when the other one is but not the other way round. In that case we say that those more restrictive are *stronger*.

As for the relationship between treewidth and cliquewidth we already mentioned that bounded treewidth implies bounded cliquewidth [43]. With regards to neighborhood diversity and its relationship to vertex cover, treewidth and cliquewidth, Lampis [58] showed that for every graph G we have $nd(G) \leq 2^{vc(G)} + vc(G)$ and $cw(G) \leq nd(G) + 1$. Furthermore, there exists a graph with constant treewidth and unbounded neighbourhood diversity (consider a path) and vice versa (consider a clique). This leads us to the following picture, which sums up all those relationships:

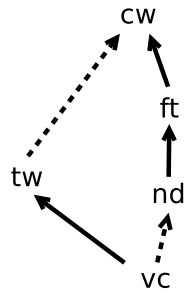


Figure 2.2: **Hierarchy of relevant parameters.** Included are vertex cover, treewidth, neighborhood diversity, finite type and cliquewidth. An arrow implies generalization, for example finite type is a generalization of neighborhood diversity. A dashed arrow indicates that the generalization may increase the parameter exponentialy, for example treewidth k implies cliquewidth at most 2^k.

2.3 Integer programming

The palette of techniques available when approaching problems from a parameterized perspective is fairly wide; for that we point the reader to an already mentioned neat overview by Fellows [27]. In this section we would like to discuss a result that we use extensively in our proofs, and also some of its extensions that might prove useful in the future.

Linear programming is a tool which found wide use in the industry soon after it was formulated at the end of 1940s. After that the important question arose if LP is solvable in polynomial time, which was finally answered positively first by Khachiyan [53] with the ellipsoid method and later by Karmarkar [50] by the practically more useful interior point method. The ellipsoid method is still of great importance theoretically though, and is the key inspiration for most of the results we will mention next.

Now we turn our attention to integer linear programming, that is, optimization with the same constraints as before *and* the requirement that the solution is integer; another equivalent formulation is the problem of deciding whether an intersection of a polytope and a lattice is nonempty. Note that many NP-hard problems can be naturally formulated as an integer linear program (ILP), so the ILP problem itself is NP-hard.

A natural parameter of the ILP problem is the dimension. As we explained in the introduction we can view the parameter as a special part of the input to which we contain the complexity. In ILP the complexity does not seem to lie in the number of rows of the linear program, as these determine in some sense how "fine" the structure of the polyhedron is. Neither does the complexity lie in the constant factors which determine the absolute dimensions of the polyhedron. This is because, roughly said, the polyhedron still remains "just" *something* like a ball (or a cone). The real complexity seems to lie in the number of ways we can branch into when exploring this ball which is precisely the dimension of the ILP. So the question we ask is this: Is the ILP problem FPT with respect to the dimension?

Fortunately for us the answer is positive again, as was proved first by Lenstra in 1983 [61] and later improved (in terms of space complexity) by Kannan [49] and Frank and Tardos [34]. Surprisingly, this result came at a time when the theory of parameterized complexity has not started yet, and later when it did, Lenstra's ILP result went unnoticed for a long time. We quote Niedermeier [68] on this topic:

> *[…] It remains to investigate further examples besides* Closest String *where the described ILP approach turns out to be applicable. More generally, it would be interesting to discover more connections between fixed-parameter algorithms and (integer) linear programming.*

This challenge was answered for example by Fellows et al. [29], who use Lenstra's algorithm to show some positive FPT results for graphs of bounded vertex cover. More currently it was also used by Lampis [58] and especially Ganian [39]; those examples and a specific suggestion made by Lampis were our main motivation to look more deeply into those techniques. Let us now state the results we use:

Definition 10 ([29])**.** p-Variable Integer Linear Programming Feasibility (p-ILP): Given matrices $A \in \mathbb{Z}^{m \times p}$ and $b \in \mathbb{Z}^{m \times 1}$, the question is whether there exists a vector $x \in \mathbb{Z}^{p \times 1}$ satisfying the m inequalities, or exactly $A \cdot x \leq b$. Here $\mathbb{Z}$ stands for the set of all integers.

Theorem 11 ([61, 49, 34])**.** p-Variable Integer Linear Programming Feasibility *can be solved using* $O(p^{2.5p+o(p)} \cdot L)$ *arithmetic operations and space polynomial in* L*, where* L *is the number of bits needed to describe* A *and* b*.*

Note that we will in fact need an optimization version of the p-ILP problem. Observe that the result in Theorem 11 can be extended to optimization through binary search and this is rigorously proved in the aformentioned paper by Fellows et al. [29]. Also, below we discuss other more general results with possibly better running times that also solve this problem. The point here is that we *can* optimize ILPs in FPT time, but we do not need to know the precise complexity bounds.

Definition 12 ([29])**.** p-Variable Integer Linear Programming Optimization (p-Opt-ILP): Let matrices $A \in \mathbb{Z}^{m \times p}$, $b \in \mathbb{Z}^{m \times 1}$ and $c \in \mathbb{Z}^{1 \times p}$ be given. We want to find a vector $x \in \mathbb{Z}^{p \times 1}$ that **minimizes** (or **maximizes**) the objective function $c \cdot x$ and satisfies the m inequalities, that is, $A \cdot x \geq b$.

Theorem 13 ([29])**.** p-Opt-ILP *can be solved using* $O(f(p)poly(L))$ *arithmetic operations and space polynomial in* L*, where* L *is the number of bits in the input.*

The results above were later extended by Khachiyan and Porkolab [52] to semidefinite integer programming. Also, instead of just deciding feasibility this result allows to optimize a convex function. Their algorithm was further improved by Heinz [45] in the specific case of minimizing a polynomial F on the set of integer points described by an inequality system $F_i \leq 0$, $1 \leq i \leq s$ where the F_i are quasiconvex polynomials in p variables with integer coefficients. The time complexity of this algorithm was recently improved by Hildebrand and Köppe [46].

A more general approach is taken by Oertel, Wagner and Weismantel [70] who prove similar results for general convex sets and minimization of convex functions, where the functions defining the convex set are not required to be (quasi)polynomials; instead they are given by three oracles. This might or might not be useful, depending on the situation. A similar path where only a hyperplane separation oracle is needed to define the convex set is taken by Dadush, Peikert and Vempala [21].

The research in the last paragraph might seem superfluous with the results of Khachiyan, Porkolab, Heinz, etc. at hand. But the motivation of the second group of authors is different from the first (those we just mentioned) because they are not focusing on generalizing the original Lenstra's result. Instead they want to get a better time complexity. There is a conjectured $\Omega(p)^p$ lower bound for the general Convex Integer Programming problem in p variables and the current state of the art (represented by the paper from Dadush et al.) almost attains this bound – they present a randomized algorithm which runs in $O(p)^p$ expected time. Moreover Dadush stated in an email conversation that it is possible to derandomize this algorithm.

In this thesis we will only make use of the result in Theorem 13 but we would like to draw more attention to the recent results mentioned above, as they seem to be powerful generalizations of the celebrated Lenstra's result, just waiting for the right problem to come.

3 Problems

In the previous chapter we have seen that neighborhood diversity and treewidth are incomparable. As such it is interesting to look at problems that are hard with respect to treewidth and see if they become FPT with respect to neighborhood diversity. That is what we do in this chapter.

The focus of this section is on several problems which are either W[1]-hard with respect to treewidth or even NP-complete for graphs with low treewidth (e.g. trees or series-parallel graphs). We tried to solve some of these problems with respect to neighborhood diversity, coming up with several positive results, and some partial ideas and pointers for future research. These results provide a finer understanding of the differences between treewidth, neighborhood diversity and related parameters. This chapter is our main contribution and contains three original results in Theorems 26, 33 and 43.

3.1 Warm-up

3.1.1 Simple problems

To get an idea of what working with graphs of bounded neighborhood diversity feels like, we will show two simple algorithms for the optimization variants of the Vertex Cover and Dominating Set problems. Both of them are FPT with respect to treewidth by the EMSOL result of Arnborg et al. [2].

Let us define them now:

Definition 14 ([27])**.** Vertex Cover
Input: A graph $G = (V, E)$
Output: A smallest set of vertices $C \subseteq V$ such that for every edge $uv \in E$ either $u \in C$ or $v \in C$.

Definition 15 ([27])**.** Dominating Set
Input: A graph $G = (V, E)$
Output: A smallest set of vertices $D \subseteq V$ such that $\forall u \in (V \setminus D) : u \in N(v)$ for some $v \in D$. (By $N(v)$ we denote the neighborhood set of a vertex v.)

The following two theorems have not appeared in print. We attribute that to the fact that they are fairly easy to see (which is why we include them in the warm-up section). Still it is worth pointing out that we are the first to give these proofs explicitly.

Let us now turn to the first mentioned problem, Vertex Cover.

Theorem 16. *The* Vertex Cover *problem can be solved on graphs of neighborhood diversity at most k in time $O(2^k + n + m)$.*

Proof. Let $G = (V, E)$ be a graph with $nd(G) = k$ and let $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ be its metagraph. We claim that for every vertex cover $C \subseteq V$ there is another vertex cover $C' \subseteq V$ with $|C'| \leq |C|$ that has the following properties:

1. For every complete metavertex $V_i \in V_{\mathcal{M}}$ either all of its vertices are in C', or all but one.

2. For every independent metavertex $V_i \in V_{\mathcal{M}}$ either all of its vertices are in C', or none of them.

We will construct C' using C. In the beginning let $C' := C$.

To see the first point, we will show that C already has the first property. For every metavertex V_i take the restriction of C to V_i, $C \cap V_i$. To cover all edges in V_i (which is a clique) at least $|V_i| - 1$ vertices need to be in C; otherwise there would be an edge uv with $u \notin C$ and $v \notin C$.

Now to the second point. Iterate over all independent metavertices V_i. If $C' \cap V_i \in \{V_i, \emptyset\}$ we can move on because V_i already has the second property. Thus $(C' \cap V_i) \subsetneq V_i$. In that case set $C' := C \setminus V_i$. To see that C' is still a vertex cover take any $u \in (V_i \setminus C')$. Because C' was a vertex cover all edges going to u were covered. The only way that could happen is that all neighbors of u were in C'. But u has the same neighbors as all other vertices in V_i, so all their edges are covered by neighbors of u as well and it was safe to remove them. Observe that after we iterated over all independent $V_i \in V_{\mathcal{M}}$ they all have the second property.

The algorithm then is to simply check all 2^k solutions of the form above (at independent metavertices: select all / no vertices; at complete metavertices: select all / all but one vertices) and return the one which is a vertex cover and uses the least number of vertices. The $n + m$ factors are for reading the input. □

The algorithm for the Dominating Set problem is similar.

Theorem 17. *The* Dominating Set *problem can be solved on graphs of neighborhood diversity at most k in time $O(3^k + n)$.*

Proof. Let $G = (V, E)$ be a graph with $nd(G) = k$ and let $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ be its metagraph. This time we claim that for every dominating set $D \subseteq V$ there is another dominating set $D' \subseteq V$ with $|D'| \leq |D|$ that has the following properties. For every metavertex $V_i \in V_{\mathcal{M}}$ one of the following holds:

1. No vertices from V_i are selected to be in D.

2. Just one vertex $v \in V_i$ is selected to be in D.

3. All vertices from V_i are selected to be in D.

First, set $D' := D$. Next, iterate over all $V_i \in V_{\mathcal{M}}$. There are three options of what to do depending on how $D' \cap V_i$ looks like.

1. If $D' \cap V_i = \emptyset$ then V_i represents the first case and we continue.

2. If $(D' \cap V_i) \subsetneq V_i$ pick arbitrary $v \in (D' \cap V_i)$ and set $D' := (D' \setminus V_i) \cup \{v\}$. D' is still a dominating set because the vertex which dominates v also dominates all other vertices $u \in V_i$ since u and v have the same neighborhood. Now V_i represents the second case and we continue.

3. If $(D' \cap V_i) = V_i$ then V_i represents the third case and we continue. Note that the only reason why we would not be able to make D' smaller in this case is that V_i would be an independent metavertex because its vertices can not dominate each other.

The algorithm then is to try all 3^k solutions of the form above (for every complete metavertex: pick zero / one vertices, for every independent metavertex: pick zero / one / all vertices), discard those that are not dominating sets, and return the smallest of the rest. □

Note that we already see a general pattern which is common to all our proofs and also the proofs from Lampis [58] and Ganian [39]. The space of all solutions of a hard problem is big (otherwise it would be possible to do a brute force check of all solutions). The way to overcome this is to show that it suffices to only look at a much smaller space of solutions of some special form because it either already contains an optimal solution, or the solutions in it can be extended to some optimal solution quickly.

3.1.2 Chromatic number

A successful technique for handling problems on graphs of bounded neighborhood diversity is Lenstra's ILP algorithm that we described in Section 2.3. In the context of neighborhood diversity it was first used already in the paper from Lampis where he introduces neighborhood diversity [58]. The problem Lampis successfully solved using this technique is the classical Chromatic Number problem (note that it is W[1]-hard for clique-width [33] but FPT for treewidth [3]). Later Ganian [39] used the ILP technique to solve two more interesting problems: p-Vertex-Disjoint Paths, which is NP-hard for clique-width [44] but FPT for treewidth [74], and Precoloring Extension, which is W[1]-hard already for treewidth [28]. We review Lampis' approach and show three more applications for coloring problems and one for the Capacitated dominating set problem.

Definition 18 ([23]). Chromatic Number (Proper coloring)
Input: A graph $G = (V, E)$
Output: A function $c : V \to \{1, \ldots, l\}$ with l as small as possible which assigns colors to vertices such that no adjacent vertices have the same color, i.e., $\forall uv \in E : c(u) \neq c(v)$.

In the rest of the thesis we will be dealing with various coloring problems and their respective colorings. To emphasize the distinction between them and the problem (and its assigned coloring) above we sometimes refer to it as the *classical* or *proper* coloring problem.

Theorem 19 ([58]). *The* Chromatic Number *problem can be solved in time* $O(f(k) \cdot poly(n))$ *on graphs of neighborhood diversity at most* k*, where* $f(k)$ *is the* f *from Lenstra's algorithm for* $p = 2^k$.

Proof. This proof is originally from Lampis' paper [58].

First observe that if V_i is an independent metavertex we can delete all of its vertices except for one because there always exists an optimal coloring where all

vertices of V_i take the same color. Thus from now on we can assume that all metavertices are complete, some of order 1.

The key observation now is that in any coloring of G every color class intersects each metavertex in at most one vertex (since all metavertices are cliques). In other words, every color class coincides with an independent set of the metagraph $G_{\mathcal{M}}$. Let $\mathcal{I}$ be the set of all independent sets of $G_{\mathcal{M}}$. (Note here that we do *not* consider the loops in this construction or in any of the following coloring problems.) Now consider the following ILP with a variable x_I for each $I \in \mathcal{I}$ (there are at most 2^k of these):

$$\begin{aligned} &\text{minimize} && \sum_{I \in \mathcal{I}} x_I \\ &\text{subject to } \forall i \in V_{\mathcal{M}} && \sum_{I : i \in I} x_I = |V_i| \end{aligned}$$

The intuition is that the variable x_I encode how many different color classes coincide with the independent set I of $G_{\mathcal{M}}$ in a coloring of G. The dimension of this ILP is at most 2^k, i.e. bounded by the neighborhood diversity of the graph so we can use Theorem 13 to solve it.

The rest of the proof is fairly simple: we need to argue that every solution found by the ILP can be transformed into a proper coloring of G and that at least one optimal coloring of G corresponds to a solution of the ILP. For details we refer the reader to Lampis' paper [58]; the main idea we wanted to get across was the observation that first, color classes of G coincide with independent sets of $G_{\mathcal{M}}$, and second, it is possible to encode this structure into an ILP. □

We will get back to this result in the following section and expand on the ideas contained in its proof to solve another problem. In Subsection 3.4.1 we will also show how to extend this proof to graphs of bounded finite type.

3.2 Coloring problems

Now we turn our attention to some coloring problems. We have already mentioned that Chromatic Number is FPT both on graphs of bounded treewidth [3] and on graphs of bounded neighborhood diversity [58] as stated in Theorem 19. All other problems we discuss in this section are hard with respect to treewidth. For the convenience of the reader we provide the exact definitions of those problems we do not deal with closely in Appendix A.1.

First, we mention those that become easy on graphs of bounded neighborhood diversity. Ganian shows this for Precoloring Extension [39]; the treewidth hardness result was given by Fellows et al. [28].

L(0,1)- and L(1,1)-coloring are W[1]-hard with respect to treewidth as shown by Fiala, Golovach and Kratochvíl [31]; we show they are FPT with respect to neighborhood diversity in Subsection 3.2.1.

The Achromatic Number problem is NP-complete already on trees [26]; we show it is FPT with respect to neighborhood diversity in Subsection 3.2.2.

Second we mention those problems whose complexity on graphs of bounded neighborhood diversity is still open. They are Equitable Coloring (W[1]-hard with respect to treewidth [28]), L(2,1)-coloring (NP-complete for graphs

with treewidth ≥ 2 [30]) and Weighted Coloring (NP-complete for graphs with treewidth ≥ 3 [65]).

A special place is taken by List Coloring – this problem is W[1]-hard not only with respect to treewidth [28] but already on graphs of bounded vertex cover [29] which also implies hardness on graphs of bounded neighborhood diversity.

Let us now finally move to the first two original positive results.

3.2.1 L(0,1)- and L(1,1)-coloring

The L(p,q)-coloring problem (or *distance constrained coloring*) is a generalization of the classical coloring problem motivated by the Frequency Assignment Problem. "In it the colors are nonnegative integers and requirements are posed on the difference of labels assigned to vertices that are close to each other." (quote from paper by Fiala et al. [31], for more see surveys by Calamoneri [14] and Yeh [77]).

The connection to the Frequency Assignment Problem can be seen when we use WiFi networks as an example. There each network operates on one of 13 channels (for Europe) and for multiple networks to operate in the same area it is desirable that networks close to each other operate on channels that are as much apart as possible. The L(p,q)-coloring problem then is to find the best assignment of channels to networks with constraints specifying the channel distance of networks which are in "physical" distances one and two.

Definition 20 ([31]). L(p,q)-coloring
Input: Graph $G = (V, E)$
Output: A function $c : V \to \{1, \ldots, l\}$ with l as small as possible which assigns labels to vertices such that $\forall u, v \in V$ at distance one (or two), it holds that $|c(v) - c(u)| \geq p$ (or q).

It holds for graphs in general that for p and q of values 0 and 1 the L(p,q)-coloring problem instance can be transformed into a Chromatic Number instance by adding and deleting edges based on the distance of vertices. The intuition (formalized in Lemma 23) is that color difference 0 means that vertices can share colors and color difference 1 means that vertices must have a different color. This in turn translates into a "are not adjacent" and "are adjacent" relation for the classical Chromatic Number problem for some new graph on the same vertex set. The reason this idea can not be used for all parameters in general is that the transformation might change the parameter.

We show how to solve the L(0,1)-coloring and L(1,1)-coloring variants, which are known to be W[1]-hard with respect to treewidth [31], in FPT time with respect to neighborhood diversity. The surprising observation is that the transformation described above preserves neighborhood diversity; for treewidth this is not true, so even though Chromatic Number is FPT with respect to treewidth it does not immediately follow that the two variants we solve would be too, and in fact they are not.

As for the other two options of $p = 1, q = 0$ and $p = 0, q = 0$ note that L(1,0)-coloring is the classical Chromatic Number problem and L(0,0)-coloring is trivial, because it gives no constraints on colors at all (it is equivalent to coloring an independent set). Later we will also talk about the related L(2,1)-coloring problem, which is NP-complete for graphs of treewidth two [30].

Let us now formalize and prove the ideas just described.

Definition 21. The *L(0,1)-transformation* of a graph $G = (V, E)$ is a graph $G' = (V, E')$ where $E' = \{uv | u, v \in V \text{are at distance 2}\}$.

Definition 22. The *L(1,1)-transformation* of a graph $G = (V, E)$ is a graph $G' = (V, E \cup E')$ with E' from the L(1,0)-transformation of G.

Note that even if G was connected, G' might not be; this does not really bother us as we can consider every component separately anyway. The following lemma is folklore and is for instance implicitly found in the paper by Fiala et al. [31].

Lemma 23 ([31])**.** *A proper coloring of the L(0,1)-transformation (or L(1,1)-transformation) of G is an L(0,1)-coloring (or L(1,1)-coloring) of G.*

Proof. Classical coloring states that no two adjacent vertices have the same color. An L(0,1)-coloring states that no two vertices at distance 2 have the same color. The L(0,1)-transformation has an edge between every two such vertices and nowhere else, so proper colorings of the L(0,1)-transformation of G correspond precisely to L(0,1)-colorings of G.

Similarly L(1,1)-coloring states that no two adjacent vertices *and* no two vertices at distance 2 have the same color. The L(1,1)-transformation has an edge between every two vertices satisfying either of the previous two conditions and nowhere else, so again proper colorings of the L(1,1)-transformation of G correspond precisely to L(1,1)-colorings of G. □

Lemma 24. *For any graph G of neighborhood diversity at most k, the L(0,1)-transformation of G has neighborhood diversity also at most k.*

Proof. The process which leads to the L(0,1)-transformation of G can be reformulated as removing all original edges and adding edges between every two vertices that were originally at distance 2 of each other. We will show that this translates nicely into a similar transformation of the metagraph $G_{\mathcal{M}}$.

First, every complete metavertex becomes an independent metavertex – all its vertices were adjacent to each other, so we had to delete these edges. Analogously, all independent metavertices except for isolated ones become complete – all vertices in an independent metavertex that is not isolated have a common neighbor in some neighboring metavertex, so they are all at distance 2 from each other. The case where an independent metavertex has no neighbor is not interesting because we can use any color for this metavertex (it is just a bunch of isolated vertices). Simply said, complete metavertices become independent and vice versa.

Second, we argue that all original metaedges are deleted and all distance-2 metaedges are added. Take any two metavertices that are adjacent in the metagraph: $V_iV_j \in E_{\mathcal{M}}$. We see that every vertex $u \in V_i$ and every vertex $v \in V_j$ are adjacent. Thus the L(0,1)-transformation deletes all of these edges and now all these u and v are non-adjacent – hence there is a meta-nonedge between V_i and V_j. Similarly for any two metavertices V_i, V_j that are at distance 2 in the metagraph, all of their $u \in V_i, v \in V_j$ are at distance 2 from each other, so they

become adjacent – which is the same as saying that a metaedge between V_i and V_j was added, or $V_iV_j \in E'_{\mathcal{M}}$.

Note that some neighborhood classes can coalesce in the process, but that only lowers neighborhood diversity. □

Lemma 25. *For any graph G of neighborhood diversity at most k, the L(1,1)-transformation of G also has neighborhood diversity at most k.*

Proof. The proof is analogous to the proof of the previous lemma. The only difference is that the L(1,1)-transformation process keeps the complete metavertices as they are, and also keeps the original metaedges. □

Theorem 26. *There is an algorithm solving the* L(0,1)-COLORING *and* L(1,1)-COLORING *problems in time $O(f(k) \cdot poly(n))$ on graphs of neighborhood diversity at most k.*

Proof. Solve the CHROMATIC NUMBER problem for the L(0,1)- or L(1,1)-transformation of G. This can be done in the stated time by Theorem 19. It gives the correct answer according to Lemma 23 and the bound on neighborhood diversity is preserved as stated in Lemmas 24 and 25. □

We return to the L(0,1)- and L(1,1)-COLORING problems in Subsection 3.4.2 and show that the second is FPT on graphs of bounded finite type.

3.2.2 Achromatic number

A *complete coloring* is a special kind of coloring which is minimal in the sense that it cannot be transformed into a proper coloring with fewer colors by merging some color classes. The *achromatic number* $\psi(G)$ of a graph is the maximum number of color classes of such a complete coloring.

Definition 27. COMPLETE COLORING (ACHROMATIC NUMBER)
Input: A graph $G = (V, E)$
Output: A partition of V into l disjoint sets $V_1, V_2, \ldots, V_l$ for l as large as possible, such that each V_i is an independent set for G and for each pair of distinct sets V_i, V_j they *touch*, meaning that $V_i \cup V_j$ is not an independent set. The highest possible l is the *achromatic number* of G.

Note that a complete coloring is defined as a partition of the vertex set. The typical way of defining a coloring as a mapping from vertices to colors would not be as comfortable to work with here, so we do not use it.

A useful way to look at the achromatic number is that it gives an upper bound on how badly can the greedy algorithm for coloring do on a graph – if it found a coloring with more than $\psi(G)$ colors there would always be a pair of colors whose union is an independent set and so they can be safely merged into just one color, which is something the greedy algorithm would have done, which in turn is a contradiction.

This problem is hard on some special graph classes, for us most importantly trees [26] and hence graphs of bounded treewidth. We will show that it is FPT on graphs of bounded neighborhood diversity.

The proof goes roughly like this. We see that the solution is again some collection of independent sets of the metagraph, that it has to give a different color

to every vertex in complete metavertices, but unlike in a classical coloring, it makes sense to color independent metavertices with many colors. The key insight is that even though there might be many options how to color an independent metavertex, what really only matters are the colors used, not how many vertices are colored by which color. In other words, all complete colorings of graphs with neighborhood diversity k fall into few (i.e., only depending on k) equivalence classes for a suitable equivalence. Moreover it is possible to express these equivalence classes as the solution space of an ILP. The other key idea is that since the number of all independent sets of the metagraph is bounded, we can safely go over all such subsets of them where every two independent sets touch and for these solve an ILP. We will now state this formally.

Definition 28. Let $G = (V, E)$ be a graph with neighborhood diversity k and let $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ be its metagraph. Disjoint subsets $W_1, W_2, \ldots, W_l$ of V are a *semifinished coloring* if they are all independent sets in G, every two of them touch, and all complete metavertices are *covered* by them (i.e., for every complete $V_i \in V_{\mathcal{M}}$ it holds that $V_i \subseteq W = \bigcup W_i$). Specifically, we do not require that $W = V$ and that independent metavertices are fully covered.

Definition 29. We call a semifinished coloring $W_1, \ldots, W_l$ of a graph G with metagraph $G_{\mathcal{M}}$ *fundamental* if the following two conditions are satisfied:

1. Every color intersects every independent metavertex in *at most one* of its vertices, i.e., for every independent metavertex $V_i \in V_{\mathcal{M}}$ and every color W_j, $|V_i \cap W_j| \leq 1$

2. Every independent metavertex contains *at least one* colored vertex, i.e., for every independent metavertex $V_i \in V_{\mathcal{M}}$ there is at least one W_j with $|V_i \cap W_j| = 1$.

Definition 30. Let $W_1, \ldots, W_l$ be a complete coloring of a graph G with $nd(G) = k$ and let $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ be its metagraph. The *reduction of a color class* W_j is a set $W'_j \subseteq W_j$ such that for every metavertex $V_i \in V_{\mathcal{M}}$ where $|V_i \cap W_j| \geq 1$ it has $|V_i \cap W'_j| = 1$.

A *reduction of a complete coloring* $W_1, \ldots, W_l$ are sets $W'_1, \ldots, W'_l$ where for every $j = 1, \ldots, l$, W'_j is a reduction of W_j.

Two complete colorings $W_1, \ldots, W_l$ and $U_1, \ldots, U_l$ that use the same number of colors are *equivalent* if there is a permutation π on the vertex set V and a permutation σ on the color classes such that for every $j = 1, \ldots, l$, $W'_j = \pi(U'_{\sigma(j)})$ where W'_j and U'_j are the reductions of these color classes.

Observe that all equivalent complete colorings reduce to the same fundamental coloring (up to isomorphism). We will show that even though we do not consider *every* complete coloring, for every complete coloring we consider the fundamental coloring associated to it and thus in a certain way all complete colorings that are equivalent to it.

Lemma 31. *Let G be a graph with $nd(G) = k$ and let $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ be its metagraph. For every complete coloring $W_1, \ldots, W_l$ of a graph G its reduction is a fundamental coloring. Moreover, every fundamental coloring $U_1, \ldots, U_l$ of G can be extended into a complete coloring which uses the same number of colors.*

Proof. The first part is a trivial observation.

For the second part we basically reverse the reduction. Given a fundamental coloring $U_1, U_2, \ldots, U_l$ which is not complete, look at all independent metavertices $V_i \in V_{\mathcal{M}}$ with $|V_i \setminus U| \geq 1$, where $U = \bigcup U_j$. Pick an arbitrary color U_j for which $|V_i \cap U_j| = 1$ and add all remaining vertices from V_i to this color, i.e., $U_j := U_j \cup (V_i \setminus U)$. There has to be such a color because every fundamental coloring intersects every independent set in at least one vertex.

The modified sets U_j still form a semifinished coloring since all of the added vertices were independent. They also form a complete coloring since there are no uncolored vertices and every two color classes still touch. □

The algorithm we give now works by first trying to guess which independent sets of the metagraph will be used in the optimal complete coloring (remember that only those collections of independent sets where every two touch are allowed) and then optimizing over all fundamental colorings that use these independent sets; we have just seen that if we find a fundamental coloring with the largest number of colors, we can extend it to an optimal complete coloring.

The algorithm then works in the following four steps.

1. Compute $\mathcal{I}$, the collection of all independent sets of the metagraph $G_{\mathcal{M}}$.

2. Iterate over all maximal subsets $\mathcal{I}' \subseteq \mathcal{I}$ where every two $I_1, I_2 \in \mathcal{I}'$ touch. Denote I_{ind} those sets I that are only made of independent metavertices.

3. For every such $\mathcal{I}'$ construct and solve the following ILP:

$$\begin{aligned}
&\text{maximize} && \sum_{I \in \mathcal{I}'} x_I \\
&\text{subject to } \forall I \in \mathcal{I}' \ \forall i \in \text{comp}(I) && \sum_{I : i \in I} x_I = |V_i| \\
&\qquad \forall I \in \mathcal{I}' \ \forall i \in \text{ind}(I) && \sum_{I : i \in I} x_I \leq |V_i| \\
&\qquad \forall I \in \mathcal{I}' \ \forall i \in \text{ind}(I) && \sum_{I : i \in I} x_I \geq 1 \\
&\qquad \forall\ I_{\text{ind}} \in \mathcal{I}' && x_{I_{\text{ind}}} \leq 1
\end{aligned}$$

   In the ILP $\text{ind}(I)$ stands for the set $I_1 \subseteq I$ containing all independent metavertices of I. Similarly, $\text{comp}(I)$ stands for the set $I_2 \subseteq I$ containing all complete metavertices of I.

   Keep the solution of that ILP that attained the maximum over all sets $\mathcal{I}'$.

4. Interpret said solution as a fundamental coloring. Extend it to a complete coloring as described in Lemma 31. Return this coloring.

The time complexity of this algorithm is in FPT: the first step computes $\mathcal{I}$ for which we know that $|\mathcal{I}| \leq 2^k$ and we have to check at most 2^k options to

get it. The second step iterates over all maximal subsets of $\mathcal{I}$ satisfying an easy to check condition; there are at most $2^{|\mathcal{I}|} = 2^{2^k}$ of these sets $\mathcal{I}'$. The third step solves the p-Opt-ILP problem with $p = 2^k$ for each $\mathcal{I}'$. The fourth step takes at most $O(n)$ time. The resulting time complexity is $O(2^{2^k} \cdot f(2^k)poly(n))$ for f from Lenstra's algorithm which is obviously FPT.

Now we turn to the correctness of the algorithm. To show that we only need two insights. First, every complete coloring reduces to some fundamental coloring (see Lemma 31). Thus, it is sufficient to search in the space of fundamental colorings since we also know how to extend it, which is what we do as the fourth step of the algorithm. We formulate the second insight as a lemma.

Lemma 32. *Every fundamental coloring corresponds to at least one feasible solution of an ILP for at least one $\mathcal{I}' \subseteq \mathcal{I}$ in the algorithm above. Moreover, every feasible solution corresponds to a fundamental coloring.*

Proof. To see the first direction, let $W_1, \ldots, W_l$ be any fundamental coloring. Its color classes are independent sets that have the property that every two of them touch. Thus, they have to be all included in some $\mathcal{I}' \subseteq \mathcal{I}$. Construct a solution to the ILP $x \in \mathbb{N}^{\mathcal{I}'}$ in the following way. For every $I \in \mathcal{I}'$ set x_I to be the number of color classes W_j for which $|W_j \cap V_i| = 1 \Leftrightarrow i \in I$, i.e., W_j and I represent the same independent set of the metagraph.

Now we look at the constraints of the ILP and check that the ILP solution x is feasible. **The first constraint** says that no two vertices in a complete metavertex share a color, which is a condition every fundamental coloring satisfies. **The second constraint** says that in any independent metavertex we cannot color more vertices than there actually are; every fundamental coloring satisfies this as well, obviously. **The third constraint** says that there has to be at least one colored vertex in every independent metavertex; fundamental colorings are defined such that this is satisfied as well. **The fourth constraint** says that we cannot use an independent set composed of only independent metavertices twice as a color because then those two colors would not touch, which is a condition all fundamental colorings satisfy.

As for the second part, we prove that every feasible solution corresponds to a fundamental coloring. First we construct the fundamental coloring according to a feasible solution x. Create colors classes W_j^I, $1 \leq j \leq x_I$, where every W_j^I takes precisely one vertex from every metavertex in I in a greedy way. It follows from the constraints of the ILP that there is sufficiently many vertices to do so. We also see that all complete metavertices are fully covered, every independent metavertex contains at least one colored vertex and from the construction of $\mathcal{I}'$ and the last constraint it follows that every two sets $I_1, I_2 \in \mathcal{I}'$ touch. Finally observe that no color has two vertices in any metavertex. Thus the coloring we constructed is fundamental. □

The last step of the algorithm is to interpret the solution of the ILP as a fundamental coloring. We do that as described in the proof of the lemma above and we know by Lemma 31 that it is possible to extend that fundamental coloring to a complete coloring with the same number of colors which we in turn know to be optimal thanks to the ILPs.

The exposition above concludes the proof of the following theorem.

Theorem 33. *There is an algorithm solving the* Complete Coloring *problem in FPT time on graphs of neighborhood diversity at most k.*

3.3 Capacitated dominating set

In this section we turn our attention to the Capacitated Dominating Set (CDS) problem. This problem is different from the previous three in the sense that we get more information than just the graph on the input. We will talk about why this is significant in Section 3.5.

As the problems we have dealt with, CDS is known to be W[1]-hard with respect to treewidth, as was shown by Dom et al. [24]. To be able to define it we first need to define the notion of a capacitated graph:

Definition 34 ([24])**.** A *capacitated graph* is a graph $G = (V, E)$ together with a capacity function $c : V \to \mathbb{N}$ such that $1 \leq c(v) \leq d(v)$, where $d(v)$ is the degree of the vertex v.

Definition 35 ([24])**.** Capacitated Dominating Set
Input: A capacitated graph $G = (V, E)$
Output: The smallest dominating set (see Definition 15) $D \subseteq V$ for which there is a mapping $f : (V \setminus D) \to D$ which maps every vertex in $(V \setminus D)$ to one of its neighbors in D in such a way that the total number of vertices mapped by f to any vertex $v \in D$ does not exceed $c(v)$.

First we make a simple yet important observation.

Proposition 36. *Let $G = (V, E)$ be a capacitated graph with $nd(G) = k$ and let $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ be its metavertex. Let $u, v \in V_i$ be two vertices from the same metavertex with $c(u) \geq c(v)$. For every CDS solution D with $v \in D$, either also $u \in D$, or $D' = (D \cup \{u\}) \setminus \{v\}$ is solution too.*

Proof. We only need to argue the second case. If $v \in D$ and $u \notin D$, we take the mapping f testifying that D is indeed a CDS and show how to create f' for D' based on the original f.

In f' we leave all as it was in f except that we take vertices assigned to v and assign them to u, i.e., $\forall w \in \{w | f(w) = v\} : f'(w) := u$. We can do that because u has the same neighbors as v, and it has sufficient capacity because $c(u) \geq c(v)$. Now we are only left with v possibly not being dominated (if u was not dominated by v), and there we let it be dominated by the vertex which originally dominated u, since u does not need to be dominated anymore, so we set $f'(v) := f(u)$. Again, this is possible because u and v have the same neighbors. □

This tells us that when solving the CDS problem, we can consider vertices from a metavertex ordered by their capacities. In other words, a solution can be characterized merely by *how many* vertices it selects for D from every metavertex, and not at all *which specific* vertices – a solution of the same size can be immediately inferred using the ordering. This is a crucial observation for the main proof of this section. We further formalize it by the following two definitions and a lemma:

Definition 37. A *capacity ordering* of a set of vertices $S \subseteq V$ is an ordering of the vertices of S by their capacities in a descending order.

Definition 38. A *fundamental solution* to the CDS problem on a graph $G = (V, E)$ with $nd(G) = k$ and a metagraph $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ is such a solution D, where for every metavertex V_i the sets $D_i = D \cap V_i$ are composed of precisely the first $s_i = |D_i|$ vertices from V_i in some capacity ordering of V_i. (We say *some* capacity ordering because we do not specify the order of vertices with equal capacities.)

Lemma 39. *Let D be an optimal CDS solution on a graph G with $nd(G) = k$, and let $G_{\mathcal{M}}$ be the metagraph of G. Define D_i for every $i \in V_i$ as $D_i = D \cap V_i$ and $s_i = |D_i|$. Construct $D' = \bigcup D'_i$ where for every $i \in V_{\mathcal{M}}$ we define D'_i as a set of first s_i vertices of V_i in its capacity ordering.*

Then D' is an optimal fundamental CDS solution.

Proof. By induction on the symmetric difference $|D \triangle D'|$ using Proposition 36. □

What the previous lemma tells us is that it is sufficient to search for an optimal solution only among fundamental solutions, which select the "best" (with respect to capacity) vertices in every metavertex. At this point it might help to try to look at the metavertices as opaque "blobs" (we can't see inside of them). For each of them we know its size and we have a function that computes how much "domination capacity" these blobs can deliver for a given cost. Eventually, we will construct an ILP where we force these blobs to dominate each other, and such a "domination capacity" function will be useful as an upper bound. Let us define it formally.

We use the following notation. For every natural number $n \in \mathbb{N}$, by $[n]$ we denote the set $\{1, \ldots, n\}$. For any set of vertices $U \in V$, by $d(U)$ we denote the sum of degrees of vertices of that set, that is, $d(U) = \sum_{v \in U} d(v)$ where $d(v)$ is the degree of a vertex v.

Definition 40. For each metavertex $V_i \in V_{\mathcal{M}}$ we define the *domination capacity function* $f_i : [|V_i|] \to [d(V_i)]$ for $m = 1, \ldots, |V_i|$ as follows:

$$f_i(m) = \sum_{j=1}^{m} c(v_j)$$

where $v_1, \ldots, v_m$ are vertices from the metavertex V_i taken in its capacity ordering.

We interpret its value as an answer to the question "how many vertices at most can the metavertex i dominate in its neighbors if we select m vertices from it to be in D?"

Now that we know that we can represent a solution simply as the number of vertices selected from each metavertex (Lemma 39), and we have the notion of a domination capacity function, we use them to construct an ILP. Let $G = (V, E)$ be a capacitated graph with $nd(G) = k$ and let $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ be its metagraph. For the purposes of this section we consider the metagraph as an oriented graph with symmetrical edges, that is, for each original $ij \in E_{\mathcal{M}}$ we now have both $ij \in E_{\mathcal{M}}$ and $ji \in E_{\mathcal{M}}$. Also remember that complete metavertices have a

loop on them. The loops represent the fact that the complete metavertices can dominate their own vertices, unlike independent metavertices.. We will get to the reason why we need oriented edges soon.

First, for each $i \in V_{\mathcal{M}}$ there is an integer variable x_i. These variables encode how many vertices to select from each metavertex. We also need $|E_{\mathcal{M}}|$ auxiliary integer variables x_{ij}, for every oriented metaedge $ij \in E_{\mathcal{M}}$ one. Those variables encode the distribution of domination capacity from i to its neighbors: x_{ij} is the number of vertices in metavertex j dominated by vertices from metavertex i.

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in V_{\mathcal{M}}} x_i \\ \text{subject to } \forall i \in V_{\mathcal{M}} \quad & \sum_{ij \in E_{\mathcal{M}}} x_{ij} \leq f_i(x_i) \qquad (*) \\ \forall j \in V_{\mathcal{M}} \quad & \sum_{ij \in E_{\mathcal{M}}} x_{ij} \geq |V_j| - x_j \\ \forall i \in V_{\mathcal{M}} \quad & x_i \leq |V_i| \end{aligned}$$

The reader has probably noticed that the constraint containing the function f_i is not linear. We will show how it can be translated into a set of linear constraints, but in the rest of this exposition we will use the integer program above, for the sake of clarity.

How to deal with the constraint containing the function f_i? One way would be to verify that the constraint is convex and refer to the general convex minimization results that we mentioned. But we do not even need to go that far – instead we expand that constraint into $O(n)$ new ones that together define the region under the function f_i. Remember that the complexity of Lenstra's algorithm is "bad" only in the dimension, not in the number of constraints. Thus, as long as there are polynomially many of them, the running time is FPT.

$$\forall i \in V_{\mathcal{M}} \; \forall m \in [|V_i|] \sum_{ij \in E_{\mathcal{M}}} x_{ij} \leq f_i(m-1) + c(v_m)(x_i - m + 1) \; (**)$$

The right hand side of this new constraint is constructed as follows. First, $c(v_m)x_i$ would be the domination capacity function if all vertices in metavertex $i \in V_{\mathcal{M}}$ had capacity $c(v_m)$. But we know that in the beginning we can do better (or at least as well), so for the first $m-1$ vertices instead of factoring in that many times $c(v_m)$, we subtract $(m-1)c(v_m)$ and add $f_i(m-1)$. That is $c(v_m)x_i - (m-1)c(v_m) + f_i(m-1) = f_i(m-1) + c(v_m)(x_i - m + 1)$ which is the right hand side as we claimed.

Let us get back to the correspondence between the ILP and fundamental CDS solutions. We will prove both directions of this correspondence, i.e., that every solution to our ILP translates to a fundamental CDS solution, and that every fundamental CDS solution translates to some feasible ILP solution. These two directions are proven in the following two lemmas.

Let us first go quickly over the ideas contained in the formulation of the CDS ILP that we use to prove these lemmas. In the ILP the CDS constraints translate naturally. First, for fixed i all x_{ij} have to sum to at most $f_i(x_i)$ because the sum of variables x_{ij} is how many vertices we expect i to dominate in its neighbors, which cannot be more than i *can* dominate, which in turn is precisely what the function f_i tells us. Second, for fixed j all x_{ij} have to sum to at least $|V_j| - x_j$

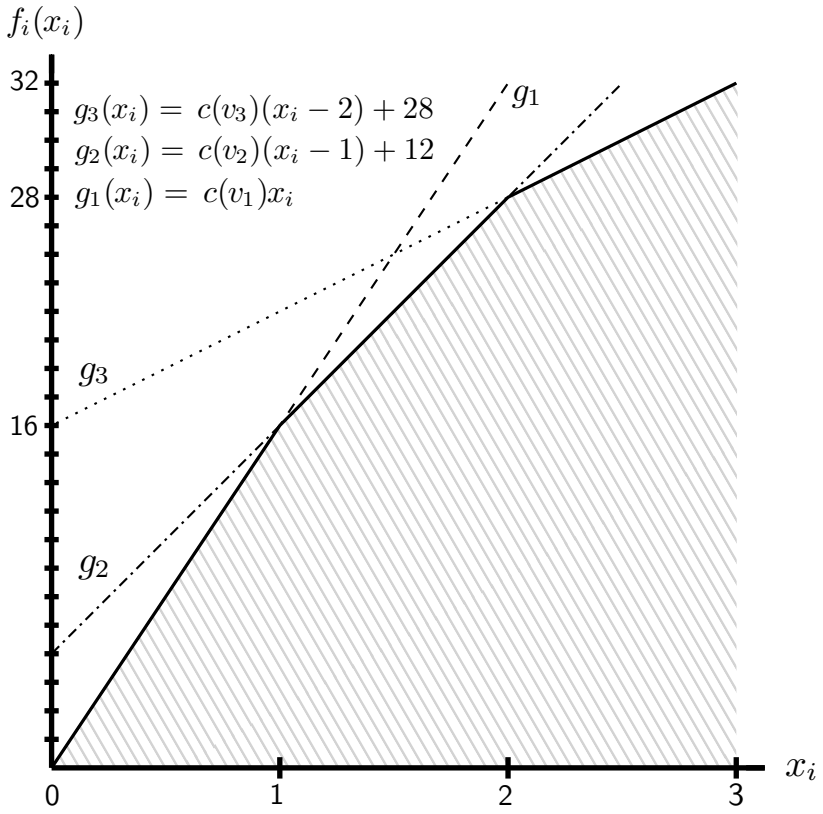


Figure 3.1: **Dealing with the constraint containing the domination capacity function.** This plot shows that the region defined by the constraint $(*)$ is indeed the same as the region defined by the set of constraints $(**)$. We use an example with three vertices v_1, v_2 and v_3 with capacities 16, 12 and 4. The region under $(*)$ is a region defined by an extension of $f_i(x_i)$ to a piece-wise linear function, drawn in bold. Using the technique described in the previous paragraphs we decompose $(*)$ into three constrains represented by the functions g_1, g_2 and g_3, drawn as dashed, dash-dotted and dotted lines.

because to dominate all of V_j at least that amount of domination capacity is needed, but those vertices in V_j that have been selected for D do not need to be dominated and there is x_j of these. Third, every x_i is upper bounded by $|V_i|$ – it is not possible to select more vertices from V_i than there actually are.

Lemma 41. *Let $G = (V, E)$ be a graph with $nd(G) = k$ and $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ be its metagraph. Every solution x to the CDS ILP determines a fundamental CDS solution D of size $\sum_{i \in V_{\mathcal{M}}} x_i$.*

Proof. The fundamental CDS solution D is constructed easily: $D = \bigcup D_i$, where for every $i \in V_{\mathcal{M}}$ set D_i to be the first x_i vertices from V_i in its capacity ordering. This is possible because $x_i \leq |V_i|$ for all metavertices. What is left now is to construct the function f testifying that D is indeed a CDS. There we make use of the helper variables x_{ij}.

Now we need to set $f(v)$ for every $v \in (V \setminus D)$ and show that it respects capacities (Definition 35). The idea here is simple and we already described it, but in this stage we want to be precise; for that we introduce some new notation.

To represent the capacities of vertices in D we construct for every metavertex $j \in V_{\mathcal{M}}$ a set $D_j^c = \{v_\alpha^l \mid v_\alpha \in D_j, 1 \leq l \leq c(v_\alpha)\}$. Let $D^c = \bigcup_{j \in V_{\mathcal{M}}} D_j^c$. Observe

that D^c is basically D with $c(v_\alpha)$ copies of every v_α. We will define a function $\psi : (V \setminus D) \to D^c$, show that it is defined for all $v \in V$ and is injective. Then for every $u \in (V \setminus D)$ we define $f(u)$ to be that $v_\alpha \in D$ for which $\psi(u) = v_\alpha^j$ (i.e., in the image of ψ, for every vertex v_α we identify its copies v_α^j to just one, the original v_α). Because ψ was injective, we know that f respects capacities.

To define ψ we define a mapping $\psi_{ij} : (V_i \setminus D) \to D_j^c$ for every metaedge $ij \in E_\mathcal{M}$. Then set $\psi(v)$ as the only $\psi_{ij}(v)$ that is defined (i.e., ψ is a union of mappings ψ_{ij} over all metaedges). Next we show that the domains of mappings ψ_{ij} are disjoint, and their union is $(V \setminus D)$.

For every metavertex $i \in V_\mathcal{M}$ partition $V_i \setminus D$ into sets V_{ij} such that the size of every V_{ij} is at most x_{ji}. (To explain the index reversal: V_{ij} is the set of vertices in i dominated by j; x_{ji} is the number of vertices j dominates in i.) This is possible, since all x_{ji} sum to to at least $|V_i| - x_i = |(V_i \setminus D)|$. The sets V_{ij} are the domains of mappings ψ_{ij}.

Now for the other part, the sets D_j^c. We will partition them into sets D_{ji}^c denoting which vertices from D_j dominate which vertices in V_i. First observe that $|D_j^c| = f_j(x_j)$. Next, as in the previous paragraph, we want every D_{ji}^c to be of size at least x_{ji}. This is possible, since all x_{ji} sum to at most $f_i(x_i) = |D_j^c|$.

Finally we see that for all $ij \in E_\mathcal{M} : |V_{ij}| \leq |D_{ji}^c|$. Because all vertices from V_{ij} see all vertices from D_{ji}^c, they can be dominated by them. Thus any $u \in V_{ij}$ can be dominated by any $v_\alpha^l \in D_{ji}^c$. Also, since $|V_{ij}| \leq |D_{ji}^c|$ there exists an injective mapping from V_{ij} to D_{ji}^c. Let ψ_{ij} be this mapping.

Because for all mappings ψ_{ij} both their domains and their images are disjoint, ψ is defined well (for every $u \in (V \setminus D)$ precisely one ψ_{ij} is defined) and is injective (every v_α^l is in the image of at most one ψ_{ij}, and those are injective). We have already shown how to construct f respecting capacities if ψ is well defined and injective, so this concludes the proof. □

Lemma 42. *Let $G = (V, E)$ be a graph with $nd(G) = k$ and let $G_\mathcal{M} = (V_\mathcal{M}, E_\mathcal{M})$ be its metagraph. Every fundamental CDS solution D determines a feasible solution x to the CDS ILP.*

Proof. Use Lemma 39 to get a fundamental solution D' of the same size as D. Now translate it to the ILP solution in the following manner. Set $x_i = |D_i'|$ for $D_i' = D' \cap V_i$. Next set $x_{ij} = |\{u | u \in V_j, v \in V_i, f(u) = v\}|$. We will verify this x satisfies all of the constraints.

The first constraint states that for every metavertex $i \in V_\mathcal{M}$ $\sum_{ij \in E_\mathcal{M}} x_{ij} \leq f_i(x_i)$. This constraint represents that to vertices from D_i' no more than $f_i(x_i)$ vertices are assigned by f. This holds, because f respects capacities, and vertices from D_i' can not dominate more vertices than what is the sum of their capacities, $f_i(x_i)$.

The second constraint states that for every metavertex $j \in V_\mathcal{M}$ $\sum_{ij \in E_\mathcal{M}} x_{ij} \geq |V_j| - x_j$. The left hand side of this constraint represents the amount of vertices in metavertex j for which f is defined, that is how many of them are dominated. The right hand side is precisely $|V_j \setminus D|$, that is how many vertices have to be dominated. Since f is defined for all $V_j \setminus D$, this constraint is also satisfied.

The third constraint is satisfied trivially – D contains from every V_i at most all of its vertices, that is $|V_i|$ of them. □

From this we see that every feasible solution to our ILP translates to *some* fundamental CDS solution, and from Lemma 39 we know that an optimal *fundamental* solution exists too, so our ILP has to find it. As for the running time, observe that we are solving the p-Opt-ILP problem for $p \leq k+k^2$, because there are k variables x_i (for every metavertex one) and at most k^2 variables x_{ij} (for every oriented metaedge one). The $n+m$ factor is needed just to read the input.

This concludes the proof of Theorem 43 and also this section.

Theorem 43. *The* Capacitated Dominating Set *problem can be solved in time $O(f(k+k^2)+n+m)$ on graphs of neighborhood diversity at most k for the function f from Lenstra's algorithm (Theorem 13).*

3.4 Finite type coloring

In Subsections 3.1.2 and 3.2.1 dealing with the Chromatic Number and L(1,1)- and L(0,1)-coloring problems we promised to return to them in the context of graphs of bounded finite type. Here we sketch how to prove that the first two problems are FPT on graphs of bounded finite type. We believe the ideas below are solid, but extending them into full rigorous proofs is beyond the scope of this thesis.

Before we turn to the aformentioned problems we define a useful notion of finite type decomposition tree. Note that this is a simplification of what Gajarský [36] describes when he says that graphs of finite type can be alternatively characterized in terms of modules as defined by Courcelle and Delhommé [18].

Remember that $\mathcal{M}_k$ is the collection of all graphs on k vertices with loops, the metagraph template set.

Definition 44. A *finite type decomposition tree* of a graph $G = (V, E) \in \mathcal{C}^{(l,k)}$ is a pair $(\mathcal{X}, T)$ where T is a rooted tree and $\mathcal{X} \subseteq 2^V$, along with two mappings $\mu : T \to \mathcal{M}_k \cup \{\texttt{nil}\}$ and $\tau : T \to \mathcal{X}$ satisfying the following conditions:

1. All leaves of T are at distance $l+2$ from the root; we say they are *at level 0*,
2. The mapping τ is *modular*, that is, for every child t' of t, $\tau(t') \subseteq \tau(t)$,
3. For every level $i \in \{0, \ldots, l+2\}$ the sets $\tau(t)$ for $t \in T$ on level i form a partition of V (i.e., they are disjoint and their union is V),
4. Every node on an odd level has k children,
5. To every node on an odd level μ assigns a metagraph,
6. μ is defined as `nil` on nodes on even levels.

The interpretation of the above is this. On an odd level $2i+1$, the mapping τ gives the subgraph of G which is a graph from $\mathcal{C}^{(i,k)}$, and μ gives its metagraph $M_k \in \mathcal{M}_k$. On an even level $2i$, τ gives the subgraph of G which is a graph from $\mathcal{C}^{(i-1,k)}$ and μ is not defined. On odd levels nodes can only have k children as every child represents a metavertex from the M_k assigned to that node by μ. On even levels nodes can have arbitrarily many children as these represent the disjoint union of arbitrarily many graphs from the previous level which are assigned to a metavertex on the next level.

3.4.1 Chromatic number and finite type

The idea used to prove that the Chromatic Number problem is FPT with respect to neighborhood diversity (Theorem 19) can be extended to graphs of bounded finite type. The key observation is that we can solve it first for all graphs on level 1 of the finite type decomposition tree (we know how – they are graphs of neighborhood diversity k) and then do some processing on level 2 which collapses the first two levels, and we repeat the same procedure.

We will demonstrate this procedure and explain why it works on an example. Let G be any graph from $\mathcal{C}^{(1,k)}$. Its finite tree decomposition has 4 levels. Level 0 contains k children for every node from level 1, and these are either cliques or independent sets. Those stand for the metavertices of graphs from $\mathcal{C}^{(0,k)}$. Level 1 are these graphs from $\mathcal{C}^{(0,k)}$, arbitrarily many of them under every node from level 2. Every level 2 node stands for one metavertex of a graph from $\mathcal{C}^{(1,k)}$ (remember that metavertices are disjoint unions of graphs from the previous level) and there is precisely k of these nodes. At level 3 there is only the root r. Its children are joined to each other by complete bipartite graphs wherever the metagraph $\mu(r)$ has an edge.

Now look at the metagraph at the top level. Observe that no two neighboring metavertices can share any colors since they are joined by a complete bipartite graph. This reminds us of the situation in graphs with bounded neighborhood diversity, except there we knew precisely how many colors every metavertex needs (its size in cliques, and one in independent sets).

But there is a way to find out how many colors every metavertex needs. Since every metavertex $i \in M_k$ is a disjoint union of graphs H_j of bounded neighborhood diversity and for those we can solve the Chromatic Number problem, we solve it for all H_j and see that $\chi(\tau(i)) = \max\{\chi(H_j) | H_j \in i\}$. This way we find out precisely how many colors are needed to color each metavertex.

Now we claim that we can safely replace every metavertex i with a clique of order $\chi(\tau(i))$, resulting in a graph G' with $\chi(G') = \chi(G)$. Moreover, $G' \in \mathcal{C}^{(0,k)}$ since it is just k cliques joined by metaedges according to the metagraph $\mu(r)$. And since G' has bounded neighborhood diversity, we can find a coloring by the original Lampis' algorithm [58] (Theorem 19).

The biggest leap in the argument above is the claim that we can replace a metavertex by a clique of the same chromatic number and that this does not increase the chromatic number of the whole graph. To that end observe the following.

Take any independent set I of $G \in \mathcal{C}^{(1,k)}$ and partition it into disjoint sets $I_i = I \cap V_i$. Let I' be a set of these metavertices i for which I_i is nonempty. Notice that I' is an independent set of the metagraph $G_{\mathcal{M}}$. On the other hand every I_i is an independent set of the subgraph induced by the metavertex and as such is a union of independent sets on the graphs from lower levels. The coloring of these graphs is independent of each other and also independent of the rest of $G_{\mathcal{M}}$ for a fixed I' (independent set of the metagraph); by "independent" we mean that two colorings of a metavertex using the same number of colors are interchangeable. Remember that the ILP used to solve Chromatic Number on graphs of bounded neighborhood diversity considers as color classes all such I' independent sets of the metagraph.

To state the sketch above rigorously we would need to prove that every coloring

can be transformed into some special form while preserving the number of colors used, and that the proposed algorithm considers all of these colorings. For the purposes of this thesis though we consider the exposition above as sufficient. We close with a sketch of the algorithm above for graphs of general finite type.

1. On the input we have $G \in \mathcal{C}^{(l,k)}$ along with its finite type decomposition tree.

2. For $i = 1, 2, \ldots, 2l + 2$ do the following. If i is odd, solve the Chromatic Number problem on all graphs induced by nodes on level i. If i is even, for all nodes replace the metavertex a induced by this node with a clique of order $c = \max\{\chi(H_j)|H_j \in a\}$.

3. Return the result of solving Chromatic Number on $\tau(r)$ for r the root of the finite type decomposition tree.

This algorithm runs in FPT time since it solves the Chromatic Number problem at most n times on graphs of neighborhood diversity at most k.

3.4.2 L(0,1)- and L(1,1)-coloring and finite type

Now we turn our attention to the two variants of L(p,q)-coloring for which we have described algorithms running in FPT time on graphs of bounded neighborhood diversity in Subsection 3.2.1. A crucial observation helped us there – the L(0,1)- and L(1,1)-transformation of a graph preserves neighborhood diversity. Does the same hold for graphs of bounded finite type?

In the first case the answer is positive. Let us look at the L(1,1)-transformation. There the situation is almost trivial, since every metavertex turns into a clique. This is because the L(1,1)-transformation adds edges between all vertices at distance 2, and since any two vertices from the same metavertex share a neighbor in a neighboring metavertex, they are at distance 2. Thus any $G \in \mathcal{C}^{(l,k)}$ turns into a graph of neighborhood diversity at most k.

With the L(0,1)-transformation the question is open, because we have to add edges between the arbitrarily many disjoint graphs on even levels and we have to delete edges inside of them. It is not obvious that the resulting graph would be of bounded finite type.

3.5 Possible future research directions

In the introduction to this chapter we gave a brief overview of problems hard with respect to treewidth. Afterwards we have shown how to solve some of them efficiently on graphs of bounded neighborhood diversity. Now, after taking a short detour to graphs of bounded finite type, we will discuss the (limited) progress we have made in solving these problems.

Before we start going into specifics, let us take a view from a higher-level perspective. In an email conversation Michael Lampis conjectured that all problems whose input is only the graph G are FPT with respect to neighborhood diversity.

> *[…] The reason is that a graph with n vertices and neighborhood diversity k can be described with $O(k \log n)$ bits. If we could reduce say,*

k-Clique, to a problem parameterized by neighborhood diversity we would in effect be compressing $\Omega(n)$ bits of input to a much smaller quantity.

The key here is the compression of information – no problem is known to be W[1]-hard with respect to neighborhood diversity *if only the graph is on the input*. On the other hand we have already mentioned that the List Coloring problem is W[1]-hard with respect to vertex cover [29] and thus also with respect to neighborhood diversity.

We will now discuss two problems. The first one is FPT with respect to treewidth, while the second one is W[1]-hard. For these problems we have made some partial progress. Afterwards we will sum up the situation for other problems and with that conclude this chapter.

3.5.1 Edge OCT

We turn to the Edge Odd Circuit Traversal (or Edge OCT) problem which is known to be FPT with respect to treewidth (for example via the EMSOL result of Arnborg et al. [2]).

Definition 45 ([56])**.** Edge OCT
Input: A graph $G = (V, E)$
Output: An edge set $F \subseteq E$ as small as possible such that $G' = (V, E \setminus F)$ is bipartite.

To prove that this problem is FPT with respect to neighborhood diversity we tried to show that there is an optimal solution for which the graph $G' = (V, E \setminus F)$ has neighborhood diversity bounded by some function $g(k)$. This would mean that we could limit our search to bipartite subgraphs of G which are composed of of at most $g(k)$ *pieces* that form equivalency classes in the sense of neighborhood diversity. There piece means a subset of the vertices of a metavertex. For example if we knew that all metavertices fall apart into at most k (or some other function of k) pieces we would have for each metavertex $V_i \in V_{\mathcal{M}}$ its partition into pieces $V_i^1, \ldots, V_i^k$, some possibly empty. There is some experimental indication behind this conjecture but the reason we talk about it is different.

Assume for now that such a claim holds – for simplicity say that every metavertex V_i falls apart into at most k pieces. We see it is possible in FPT time to try all possible ways these pieces can be connected to each other and the rest of the graph such that they form a bipartite graph, since the number of these options is the number of bipartite graphs on $O(k^2)$ vertices which is still some function of k. This gives us a set of "templates" for a candidate solution.

But there is a catch – we also need to specify the sizes of pieces, that is, if V_i falls into three pieces, we have to show how many vertices from V_i go to the first, the second, and the third piece. Even if we guessed the candidate solution correctly the brute force algorithm would lie in the class XP because it would need to check $O(n^{g(k)})$ options. We want to do better.

In a situation like that we usually found that formulating the problem as an ILP is sufficient to solve it. This problem is interesting in that it can not be easily expressed as an ILP because the objective function that we maximize (the number of remaining edges) is a sum of products (the number of edges in a

complete bipartite graph between two pieces is the product of their sizes). We hoped that the function would at least be convex but since already $x \cdot y$, the hyperbolic paraboloid, is not a convex function, we are afraid this is a dead end. Still, let us formulate the integer program we would like to use.

Let $G = (V, E)$ be a graph with $nd(G) = k$ and let $G_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ be its metagraph. Let $H_{\mathcal{M}} = (W_{\mathcal{M}}, F_{\mathcal{M}})$ be the metagraph of the candidate solution without sizes of metavertices. We assume that $H_{\mathcal{M}}$ has at most k^2 metavertices. We do not know how H (the solution) looks like because we have yet to determine the sizes of metavertices of $H_{\mathcal{M}}$. We construct a separate integer program for each candidate solution which is some partition of metavertices into pieces which means that the bipartiteness condition is already satisfied.

$$\begin{aligned} &\text{maximize} && \sum_{a_i b_j \in F_{\mathcal{M}}} x_i^a \cdot x_j^b \\ &\text{subject to } \forall V_i \in V_{\mathcal{M}} && \sum_{W_j \subseteq V_i} x_i^j = |V_i| \end{aligned}$$

The reason we mention the above is that it could still be of some use. As we already noted, Edge OCT is easily formulated as an MSO_2 optimization problem which is known to be solvable in FPT time on graphs with bounded treewidth. There is a variant of this problem with a slightly different objective function which we do *not* know how to express in MSO_2 (though it is still FPT with respect to treewidth [55]) – the Fair Edge OCT.

Definition 46 ([55])**.** Fair Edge OCT
Input: A graph $G = (V, E)$
Output: An edge set $F \subseteq E$ with the smallest maximum degree such that $G' = (V, E \setminus F)$ is bipartite.

On graphs of bounded treewidth this problem was proven to be FPT by Kolman et al. [55]. The authors mention an intriguing questions: is there a metaalgorithmical result for these "fair" problems, i.e., problems that do not optimize simply the size of a set but instead how it behaves with respect to individual vertices? We have not investigated this question but the following might give some indication that these problems could be solved on graphs with bounded neighborhood diversity using the ILP method.

If the conjecture about optimal solutions with few pieces we stated above was right for Fair Edge OCT is it possible to formulate an ILP for it? The answer is positive when we introduce an auxiliary variable Y that we use as an upper-bound on the maximum degree of F.

$$\begin{aligned} &\text{minimize} && Y \\ &\text{subject to } \forall V_i \in V_{\mathcal{M}} && \sum_{W_j \subseteq V_i} x_i^j = |V_i| \\ &\forall a_i \in W_{\mathcal{M}} && \sum_{ij \in E_{\mathcal{M}}} |V_j| - \sum_{a_i b_j \in F_{\mathcal{M}}} x_j^b \leq Y \end{aligned}$$

We hope the ideas above are eventually helpful to someone even though they did not bear any fruit at the moment.

3.5.2 Equitable Coloring

Definition 47 ([28]). Equitable Coloring
Input: A graph $G = (V, E)$
Output: A proper coloring using as few colors as possible, such that the size of all color classes is either l or $l + 1$ for some l.

The idea of reusing the original Chromatic Number algorithm immediately comes to mind. However, we have to answer the question how to encode sizes of color classes when we contracted every independent metavertex into just one vertex. In the classical Chromatic Number problem these independent metavertices played a rather insignificant role; here their role is significant, because they provide the means to balance the sizes of color classes.

As in the previous problems we would like to find some "fundamental" form of a solution which would be narrow enough to be expressed as an ILP, yet wide enough to also express some optimal solution. In this direction it seems we got stuck half way.

Remember from Subsection 3.2.2 (about Achromatic Number) that every color class in a coloring of G corresponds to an independent set in the metagraph $G_{\mathcal{M}}$. This independent set takes at most one vertex from each complete metavertex and arbitrarily many from each independent metavertex. Our question is: Given an optimal equitable coloring, can two independent sets I_1 and I_2 that correspond to the same independent set of the metagraph behave significantly differently on the independent metavertices? (By "significantly different behavior" we mean for instance that I_1 selects all but one vertices from V_i and just one from V_j and I_2 vice versa.) The answer is yes, they can, but then there is also a solution using the same number of colors where they behave nicely. (By "behave nicely" we mean that in every metavertex they select either the same number of vertices, or they differ by just one.)

To see that, take any equitable solution and let $I_1, I_2, \ldots, I_l$ be color classes that correspond to the independent set I of the metagraph. For every $j \in [l]$ and for every metavertex $i \in V_{\mathcal{M}}$ let $I^i_j = I_j \cap V_i$. Let $s_i = \sum_{1 \leq j \leq l} |I^i_j|$ and define new sets $\bar{I}^i_j$ such that they have either $\lceil \frac{s_i}{l} \rceil$ or $\lfloor \frac{s_i}{l} \rfloor$ vertices; choose between those two options in such a way that their sizes sum to precisely s_i. With a little caution it is possible to make these choices across the metavertices in such a way that first, the sets $\bar{I}^i_j$ differ in size by at most one from each other and second, they also differ by at most one from other colors.

In the original Chromatic Number ILP we had a variable x_I for every independent set I of the metagraph. Here it seems appropriate to have additional variables x^i_I for every independent set I of the metagraph and for every independent metavertex i; this variable would encode how many vertices from i are selected for the coloring for every choice x_I. The problem is that this leads to a constraint of the form $\forall i \in V_{\mathcal{M}} \ \sum_{I \in \mathcal{I}} x_I \cdot x^i_I = |V_i|$ which is not convex so we can not use any of the results we mentioned. (This is not a problem in the results we tried to use – such an integer program simply is not convex, so there is little hope if we insist on this form.)

We tried different ways of encoding the described "fundamental" solutions to no avail.

3.5.3 Miscellaneous

The other problems we tried to deal with fall into two categories.

The first one are those where only the graph is on the input and so there is much hope that they will turn out to be FPT with respect to neighborhood diversity. Two problems remain in this category that we have not mentioned yet. The first one is the L(2,1)-coloring, which was proven by Fiala, Golovach and Kratochvíl [30] to be NP-complete already for series-parallel graphs. The second one is the p-Edge-disjoint paths problem, which was also proven to be NP-complete already for series-parallel graphs by Nishizeki, Vygen and Zhou [69].

The second category are those problems with some additional information on the input. We have already mentioned that the List Coloring problem is W[1]-hard with respect to vertex cover and thus also with respect to neighborhood diversity, but we have also shown in Section 3.3 that the Capacitated dominating set problem is FPT with respect to neighborhood diversity. This gives us the two ends of the spectrum of what we can hope for. There are two problems for which this remains open. First, the Weighted coloring problem, proven to be NP-complete for graphs of treewidth three and more by McDiarmid and Reed [65]. Second, the Capacitated Vertex Cover problem, proven to be W[1]-hard with respect to treewidth by Dom et al. [24]

Let us conclude with a table which sums up this chapter. By [*] we denote results presented in this thesis.

Problem	**Treewidth**	**Neighborhood diversity**
Edge OCT	FPT [2]	Open
Fair Edge OCT	FPT [55]	Open
Chromatic Number	FPT [2]	FPT [58]
Precoloring extension	W[1]-hard [28]	FPT [39]
Achromatic number	NP-c on trees [26]	FPT [*]
L(0/1,1)-coloring	W[1]-hard [31]	FPT [*]
L(2,1)-coloring	NP-c for TW ≥ 2 [30]	Open
p-Edge-disjoint paths	NP-c for TW ≥ 2 [69]	Open
Equitable coloring	W[1]-hard [28]	Open
List coloring	W[1]-hard [28]	W[1]-hard [29]
Weighted coloring	NP-c for TW ≥ 3 [65]	Open
CDS	W[1]-hard [24]	FPT [*]
CVC	W[1]-hard [24]	Open

4 Parameters

In this chapter we make a short survey of the state of the art in the world of parameters for sparse graphs and dense graphs. We realize that it is limited and there are parameters that we do not mention. The main message of this chapter is that the situation of parameters suitable for dense graphs is much more clouded and with less consensus than the situation of parameters suitable for sparse graphs, and even that is not at all finished. For the convenience of the reader, we provide the exact definitions of parameters discussed in this chapter in Appendix A.2.

We admit that our understanding of the words "sparse" and "dense" in the following sections is quite limited and specific to the scenario we are focusing on. A relevant recent paper dealing with the difficulty of the concept of sparsity (and related concepts) is due to Nešetřil and de Mendez [67]. Typically, graphs with $O(n)$ edges are defined as sparse and graphs with $\Omega(n^2)$ edges as dense. With these definitions there are many graph classes that do not fall into either category.

One of the problems with these definitions in our scenario is that no parameter subsumes either class completely. Already planar graphs (which have $O(n)$ edges) do not have bounded treewidth. The situation for cliquewidth is similar [48]. Only the other direction holds: *all* graph classes of bounded treewidth have $O(n)$ edges, and *some* graph classes of bounded cliquewidth have $\Omega(n^2)$ edges. There are other problems that we will mention.

Thus, the way we use these words is mostly motivated by practice and how they are used in literature.

In the following sections we will be using the notion of equivalence of parameters. Two parameters A and B are said to be *equivalent* if for every graph G the parameter A is bounded by a constant if and only if the parameter B is bounded by a constant.

4.1 Sparse graphs

In this section we explain why we think that treewidth is a successful parameter for a significant part of sparse graphs. We use a survey from Bodlaender [8] extensively in the following paragraphs.

The notion of treewidth was introduced as a part of the *Graph Minor Project* of Roberson and Seymour [72, 73] in the 80s. In hindsight we see that there are other equivalent definitions of treewidth which were studied by other authors independently, some even before Roberson and Seymour. Graphs with bounded treewidth gathered interest because many problems that are intractable (e.g., NP-hard) become linear time (or polynomial time with polynomials of reasonable

degree) solvable when restricted to graphs of bounded treewidth. Bodlaender [8] says:

> *Such algorithms have been found for many combinatorial problems and also have been employed for problems from computational biology, constraint satisfaction, probabilistic networks and other areas. […] In other words: many graph problems become fixed-parameter tractable when parameterized by the treewidth of the input graph.*

Also, since the end of the 80s and the beginning of the 90s treewidth gathered attention from the fields of automata theory, database theory and finite model theory, resulting in the famous metaalgorithmical theorem of Courcelle [17]. This theorem states that the model checking problem of MSO_2 and CMSO logics is FPT with respect to treewidth. (CMSO, or counting MSO, is an extension of the MSO logic to also express the cardinality of sets modulo p.) This was later extended to optimization variants (EMSO) by Arnborg et al. [2] and recently to some other more specialized logics, for instance MSO with local cardinality constraints (Szeider [76]). The interest from database theory is demonstrated for example by papers from Gottlob et al. [42] who show how to translate MSO formulas into Datalog queries. As for finite model theory, it is where the methods Gottlob et al. and many others use come from; for more see Libkin's book [63].

For a parameter to be useful in practice there has to be a way to efficiently (in FPT time) determine the parameter and afterwards obtain the relevant decomposition. This is because we usually do not know the parameter exactly, only that it is somehow bounded on the input graph.

Both of these are possible for treewidth thanks to the extensive work done by Bodlaender et al. in the past approximately 15 years. The first important step was that Bodlaender [7] showed in 1996 that for fixed k it is possible to decide if a graph G has treewidth at most k in linear time, and if the answer is positive, to return a tree decomposition of width k. This algorithm was shown to be impractical by Röhrig [75], but heuristics which work reasonably were found instead. In the following years much research was done in this area and is summed up in a series of papers. The first two papers due to Bodlaender and Koster show polynomial heuristics for computing upper bounds [10] and lower bounds [11] of the treewidth of a graph. The third paper is yet to come out but it will be dedicated to exact algorithms for computing treewidth; some progress is described in a paper by Bodlaender et al. [9] which describes exact exponential algorithms running in space polynomial in the parameter (unlike the first algorithm from 1996). Those algorithms combined with aformentioned heuristics provide the needed tools to obtain good (almost optimal) tree decompositions quickly.

A great example demonstrating how far the research done on treewidth got is the recent paper "Evaluation of a MSO solver" by Langer et al. [59]. In it the authors evaluate an implementation of a new kind of proof for Courcelle's theorem which they introduced the previous year [54]. In spite of the fact that the parameterized complexity community still considers metaalgorithmical theorems like Courcelle's theorem to be merely of theoretical interest (see Niedermeier's book [68]), the MSO solver implemented by Langer et al. wildly outperforms the industry standard ILP solver CPLEX on a real-world instance of the Bus Stop Location problem.

We see the following three reasons as the main reason why treewidth is a successful parameter:

1. The class of graphs with bounded treewidth is wide enough to capture well many real-world instances for many practical problems which become FPT with respect to treewidth.

2. The model checking problem for a variety of logical languages becomes FPT on graphs of bounded treewidth.

3. It is possible to determine treewidth and find an optimal tree decomposition in FPT time with respect to treewidth. Moreover, practical heuristics exist.

The biggest theoretical problem of treewidth are the lower bounds on the complexity of model checking for MSO logics given by Frick and Grohe [35] in 2004. Because of the positive results due to Langer et al. [59, 54] we described above we do not think those lower bounds play a big role in practice. Still, there was interest in finding some answer to this obstacle.

To that end attention turned to *tree-depth*, a parameter introduced in 2006 by Nešetřil and de Mendez [66], which creates a finer hierarchy between graphs of bounded vertex cover and bounded treewidth. Instead of explaining this relationship between treewidth and tree-depth we refer to a paper by Blumensath and Courcelle [6] which proves an equivalence between tree-depth and *l-depth treewidth*, a parameter that is easier to describe.

A graph G has l-depth treewidth bounded by k if a tree decomposition of width k exists for G and also it is possible to choose a root in this decomposition such that all leaves are in depth at most l. The equivalence between those two parameters is such that tree-depth $\leq k$ implies k-depth treewidth k, and l-depth treewidth $\leq k$ implies tree-depth lk. Treewidth is often described as a measure of how close a graph is to a tree; in this sense tree-depth is a measure of how close a graph is to a star.

Why is tree-depth important? Gajarský and Hliněný [37] have recently proven that MSO_2 is FPT with respect to tree-depth *and the function f has elementary dependence on the formula* (i.e., the height of the exponential tower does not depend on the formula). Intuitively, this construction allows us to trade the nonelementary dependence on the formula for a bound on the depth of the tree decomposition. This seems like a good answer to the lower bounds of Frick and Grohe from a theoretical standpoint.

Even when we factor in the limitations of our use of the word "sparse", judging by the amount of interest treewidth generated both in theory and practice we conclude that it successfully answers many important questions.

4.2 Dense graphs

In this section we explain why we think that the search for a good parameter for "dense" graphs is still very much an ongoing effort.

The notion which is probably the closest to treewidth with regards to "dense" (again, for a suitable definition thereof) graphs is the parameter cliquewidth (see

Definition 4). A good starting point for this parameter is the survey conducted by Kaminski et al. [48]. Cliquewidth was introduced by Courcelle and Oum [20] in 2000 as a generalization of treewidth, as bounded treewidth implies bounded cliquewidth [43]; on the other hand there are graphs with constant cliquewidth but unbounded treewidth. Many important hard problems are FPT on graphs of bounded cliquewidth (for more specifics see the introduction chapter of a recent paper by Ganian, Hliněný and Obdržálek [41]). On the other hand, since more graphs have bounded cliquewidth than bounded treewidth it is not surprising that not all problems that are FPT with respect to treewidth are also FPT with respect to cliquewidth. For example the Chromatic number problem is W[1]-hard with respect to cliquewidth [33].

As for metaalgorithmical results, already in 2000 Courcelle, Makowski and Rotics [19] proved the equivalent of Courcelle's theorem for graphs of bounded cliquewidth with one significant difference: only MSO_1 (not MSO_2) model checking is FPT with respect to cliquewidth. Another important result were the nonelementary lower bounds on the complexity of the function $f(k)$ of MSO_1 model checking given by Frick and Grohe [35].

We are already starting to see some significant disadvantages of cliquewidth when compared to treewidth, although some might be attributed simply to the fact that cliquewidth is younger by about 15 years. We have not yet discussed the third "component of success" of treewidth, that is the FPT algorithm determining the treewidth of a graph and computing its decomposition.

Upon seeing that it is possible to solve some hard problems efficiently on graphs of bounded cliquewidth, the problem of determining (or at least approximating) cliquewidth gained importance. In 2006, Oum and Seymour [71] proved that if a graph G has cliquewidth at most k, it is possible to find a $(2^{k+1}-1)$-expression (the equivalent of a tree decomposition for cliquewidth) defining G in FPT time. This discrepancy between orders of k and 2^k was the main motivation for the introduction [47] of a parameter called *rank-width*.

It holds [71] that for any graph G, $rw(G) \leq cw(G) \leq 2^{rw(G)+1}-1$, that is, the parameters are equivalent and cliquewidth is always at most roughly $2^{rw(G)}$. It still remains open if there is an FPT algorithm that would, for a graph G with $cw(G)=k$, construct its k-expression. For these reasons rank-width is often more popular for algorithm design. It is worth mentioning that the ideas used in the alternative proof of Courcelle's theorem for treewidth given by Kneis et al. [54] were used in the proof of the metaalgorithmical theorem on rank-width by Langer et al. [60].

Another parameter equivalent to cliquewidth (and thus rank-width) is *boolean-width*. It was introduced by Bui-Xuan et al. [12] in 2009 and further investigated by Adler et al. [1] and Belmonte and Vatshelle [5]. The reason why boolean-width is interesting is the combination of two of its properties. First, boolean-width can be exponentially smaller than rank-width. More interestingly, many graph classes (see Figure 1 in the paper by Belmonte and Vatshelle [5]) have boolean-width $O(\log n)$. It is not typical in parameterized complexity to look at graph classes with *logarithmic* values of the parameter, but here it makes sense. The reason is that, second, a wide class of problems (see the paper by Adler et al. [1]) can be solved on graphs of boolean-width w in time $2^{O(w)}\mathrm{poly}(n)$ (i.e., with $f(w)$ singly exponential). When those two properties are combined the result are FPT

algorithms for those problems on the classes with boolean-width $O(\log n)$. The authors also show that for the relevant graph classes boolean decompositions can be obtained in polynomial time.

Observe that in this sense boolean-width is *not* equivalent to cliquewidth or rank-width, because graphs with boolean-width $O(\log n)$ can have cliquewidth much bigger (even doubly exponentially), so even if we were able to use the same algorithms as for boolean-width, we would not get a runtime polynomial in n. Note that this is the only case known to the aformentioned authors and to us too where this technique is used; typically only classes of graphs with a parameter bounded by a constant are studied.

Above we summed the progress motivated by the original question of determining cliquewidth efficiently – the notions of rank-width and boolean-width both proved to carry significant advantages over clique-width even though they are all bounded by some constant for the same graph classes. Now we turn to the research motivated by the lower bounds for model checking due to Frick and Grohe [35].

The first parameter for which model checking of MSO_1 with elementary f was proved was neighborhood diversity. This recent result is due to Lampis [58]. Gajarský [36] generalized neighborhood diversity to finite type and proved a similar result in his thesis. The most interesting result in this regard is one analogous to the one we mentioned in the previous section for tree-depth. Just like tree-depth fills the gap between vertex cover and treewidth, *shrub-depth* (introduced by Ganian et al. [40]) fills the gap between vertex cover and cliquewidth. Interestingly, graphs with shrub-depth 1 are precisely graphs of bounded neighborhood diversity [38]. Gajarský also noted in an email conversation that graphs of finite type have bounded shrub-depth, but not vice versa. The analogy between tree-depth and shrub-depth is that tree-depth allows MSO_2 model checking with elementary f in the time complexity and shrub-depth allows MSO_1 model checking with elementary f.

Still, there remain problems with shrub-depth and related parameters. Ganian et al. [40] state: "Primarily, we do not know yet how to efficiently (in FPT) construct decompositions related to our depth parameters (in this respect our situation is similar to that of clique-width)."

In this thesis we did the main part of our work on graphs of bounded neighborhood diversity. Even though this parameter is easy to work with, easy to determine (in time $poly(n)$, unlike all other parameters we mentioned) and theoretically attractive (since it is incomparable with treewidth), we have to realistically admit that it is probably not very useful in practice. Recently, Lambert [57] investigated the values of $vc(G)$ and $nd(G)$ for some graphs which are encountered in practice. Even though we have our reservations towards the choice of these graphs (there are better samples available online) it is apparent that unlike with treewidth it is unusual to encounter graphs of bounded neighborhood diversity in practice.

Thus we conclude that there is no one parameter for "dense" graphs that would fit all three criteria from the previous section – let many hard problems become tractable on many graph classes, allow for efficient model checking and allow for efficient finding of decompositions – so the search is still very much open. We sum this chapter with a figure from Subsection 2.2.6 which we updated

to contain all of the parameters discussed in this chapter.

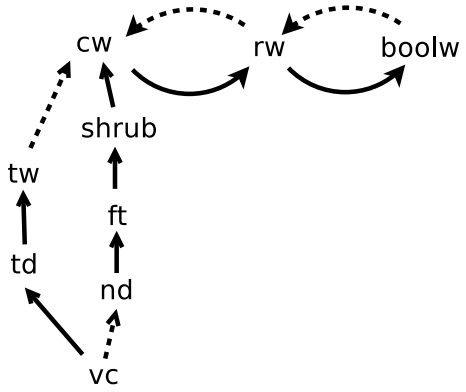


Figure 4.1: **Extended hierarchy of discussed parameters.** Included are vertex cover, tree-depth, treewidth, neighborhood diversity, finite type, shrub-depth, cliquewidth, rankwidth and booleanwidth. An arrow implies generalization, for example finite type is a generalization of neighborhood diversity. A dashed arrow indicates that the generalization may increase the parameter exponentialy, for example treewidth k implies cliquewidth at most 2^k.

5 Conclusion

The area of parameterized complexity is still young and as such still has to try many new trails. Neighborhood diversity is a recently introduced parameter that might be viewed as such a trail.

In this thesis we have shown that three problems become fixed-parameter tractable with respect to neighborhood diversity. What makes these results interesting is the fact that these problems are hard with respect to treewidth. The problems for which we have shown the aformentioned results are L(0,1)- and L(1,1)-coloring, Complete coloring and Capacitated Dominating Set. This is the main contribution of this thesis.

While working on this thesis we have also gathered a list of other problems that are hard with respect to treewidth. We hope this list serves as inspiration for future research investigating the finer differences between treewidth and neighborhood diversity. Especially interesting are problems that contain additional information besides the graph as their input because they have big potential to prove hard with respect to neighborhood diversity, thus enhancing our understanding of it. We also gave a brief overview of some other relevant parameters. In this overview we see signs of which paths lead in good directions and which had to be adjusted over time.

It remains to be seen where will research of neighborhood diversity and related parameters lead us. At this point it is too early to judge, but even if this path only leads to the discovery of a dead end we have learned something valuable not only for us, but also for the rest of the scientific community.

A Appendix

A.1 Problem definitions

Here we provide the definitions of those computational problems that we mentioned in Chapter 3 but did not deal with any further.

Definition 48 ([39])**.** Precoloring extension
Input: A graph $G = (V, E)$ and a partial proper coloring $c' : V \to \mathbb{N}$
Output: A proper coloring c of G which extends c' (i.e., it agrees with c' on all vertices for which c' was defined) and uses as few colors as possible.

Definition 49 ([29])**.** List coloring
Input: A graph $G = (V, E)$ and for each vertex $v \in V$ a list $L(v)$ of permitted colors.
Output: A proper coloring c of G with $c(v) \in L(v)$ for every $v \in V$.

Definition 50 ([65])**.** Weighted coloring
Input: A graph $G = (V, E)$ and for each edge $uv \in E$ its length $l(uv) \in \mathbb{N}$.
Output: A mapping $c : V \to \{1, \ldots, l\}$ for l as small as possible such that for every edge $uv \in E$, $|c(u) - c(v)| \geq l(uv)$.

Definition 51 ([69])**.** p-Edge-disjoint paths
Input: A graph $G = (V, E)$ and a set of p pairs of vertices $(s_1, t_1), \ldots, (s_p, t_p)$ in G.
Output: If they exist, p edge-disjoint paths $P_1, \ldots, P_p$ in G such that P_i joins s_i and t_i for $i = 1, \ldots, p$, or an empty set $\emptyset$ if they do not exist.

Definition 52 ([24])**.** Capacitated Vertex Cover
Input: A capacitated graph $G = (V, E)$ (see Definition 34)
Output: The smallest vertex cover (see Definition 14) $C \subseteq V$, for which there is a mapping $f : E \to C$ which maps every edge in E to one of its two endpoints such that the total number of edges mapped by f to any vertex $v \in C$ does not exceed $c(v)$.

A.2 Parameter definitions

Here we have provide the definitions of those parameters that we only mentioned in Chapter 4. All of them are direct quotes from the cited papers.

Definition 53 ([37, 66])**.** The *closure* $cl(F)$ of a rooted forest F is the graph obtained from F by adding from each node all edges to its descendants. The *tree-depth* $td(G)$ of a graph G is the smallest height (distance from the root to all leaves) of a rooted forest F such that $G \subseteq cl(F)$.

Definition 54 ([48, 47])**.** For a cut $C = (A, B)$ (partition of the vertex set into two disjoint subsets A, B), the *cut-rank* of C, denoted $\text{cutrk}_G(C)$, is the linear rank of the $|A| \times |B|$ submatrix of the adjacency matrix of G corresponding to the set of pairs (a, b) with $a \in A$ and $b \in B$, where the matrix is viewed over $GF(2)$.

If T is a tree of maximum degree 3 and L is a bijection from the set of leaves of T to the vertices of G, then (T, L) is a *rank-decomposition* of G. For each edge e of T, the two components of $T - e$ define a cut C_e in G. The *width* of the edge e is $\text{cutrk}_G(C_e)$. The *width* of (T, L) is the maximum width of all the edges of T. The *rank-width* of G, denoted $rw(G)$, is the minimum width of all rank-decompositions of G.

The definition of boolean-width is involved and requires auxiliary definitions.

Definition 55 ([4])**.** A *decomposition tree* of a graph G is a pair (T, δ) where T is a tree having internal nodes of degree three and $n = |V(G)|$ leaves, and δ is a bijection between the vertices of G and the leaves of T. Every edge of T defines a cut $(A, \overline{A})$ of the graph, i.e., a partition of $V(G)$ in two parts, namely the two parts given, via δ, by the leaves of the two subtrees of T we get by removing the edge. Let $f : 2^V \to \mathbb{R}$ be a symmetric function, i.e., $f(A) = f(\overline{A})$ for all $A \subseteq V(G)$, also called a *cut function*. The *f-width* of (T, δ) is the maximum value of $f(A)$, taken over all cuts $(A, \overline{A})$ of G given by an edge uv of T. The *f-width* of G is the minimum f-width over all decomposition trees of G.

Definition 56 ([4])**.** Let G be a graph and $A \subseteq V(G)$. Two vertex subsets $X \subseteq A$ and $X' \subseteq A$ are *neighborhood equivalent* with respect to A, denoted by $X \equiv_A X'$, if $\overline{A} \cap N(X) = \overline{A} \cap N(X')$.

Definition 57 ([4])**.** The *cut-bool* $: 2^{V(G)} \to \mathbb{R}$ *function* of a graph G is

$$\textit{cut-bool}(A) = \log_2 |\{S \subseteq \overline{A} : \ \exists X \subseteq A \wedge \ S = \overline{A} \cap \bigcup_{x \in X} N(x)\}|$$

Using Definition 55 with $f =$ cut-bool we define the *boolean-width of a decomposition tree*, denoted $boolw(T, \delta)$, and the *boolean-width of a graph*, denoted $boolw(G)$.

The definition of shrub-depth requires one auxiliary definition.

Definition 58 ([40])**.** We say that a graph G has a tree-model of m labels and depth d if there exists a rooted tree T (of height d) such that

1. the set of leaves of T is exactly $V(G)$,
2. the length of each root-to-leaf path in T is exactly d,
3. each leaf of T is assigned one of m labels (T is m-labelled),
4. and the existence of a G-edge between $u, v \in V(G)$ depends solely on the labels of u, v and the distance between u, v in T.

The class of all graphs having a tree-model of m labels and depth d is denoted by $\mathcal{TM}_m(d)$.

Definition 59 ([40])**.** A class of graphs $\mathcal{G}$ has shrub-depth d if there exists m such that $\mathcal{G} \subseteq \mathcal{TM}_m(d)$, while for all natural m it is $\mathcal{G} \nsubseteq \mathcal{TM}_m(d-1)$.

Bibliography

[1] Adler, I., Bui-Xuan, B.-M., Rabinovich, Y., Renault, G., Telle, J. A., and Vatshelle, M. On the Boolean-Width of a Graph: Structure and Applications. In *WG* (2010), D. M. Thilikos, Ed., vol. 6410 of *Lecture Notes in Computer Science*, p. 159–170.

[2] Arnborg, S., Lagergren, J., and Seese, D. Easy Problems for Tree-Decomposable Graphs. *J. Algorithms 12*, 2 (1991), 308–340.

[3] Arnborg, S., and Proskurowski, A. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics 23*, 1 (1989), 11–24.

[4] Belmonte, R., and Vatshelle, M. On graph classes with logarithmic boolean-width. *CoRR abs/1009.0216* (2010).

[5] Belmonte, R., and Vatshelle, M. Graph Classes with Structured Neighborhoods and Algorithmic Applications. In *WG* (2011), P. Kolman and J. Kratochvíl, Eds., vol. 6986 of *Lecture Notes in Computer Science*, Springer, p. 47–58.

[6] Blumensath, A., and Courcelle, B. On the Monadic Second-Order Transduction Hierarchy. *Logical Methods in Computer Science 6*, 2 (2010).

[7] Bodlaender, H. L. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput. 25*, 6 (1996), 1305–1317.

[8] Bodlaender, H. L. Fixed-Parameter Tractability of Treewidth and Pathwidth. In *The Multivariate Algorithmic Revolution and Beyond* (2012), H. L. Bodlaender, R. Downey, F. V. Fomin, and D. Marx, Eds., vol. 7370 of *Lecture Notes in Computer Science*, Springer, p. 196–227.

[9] Bodlaender, H. L., Fomin, F. V., Koster, A. M. C. A., Kratsch, D., and Thilikos, D. M. On Exact Algorithms for Treewidth. In *ESA* (2006), Y. Azar and T. Erlebach, Eds., vol. 4168 of *Lecture Notes in Computer Science*, Springer, p. 672–683.

[10] Bodlaender, H. L., and Koster, A. M. C. A. Treewidth computations I. Upper bounds. *Inf. Comput. 208*, 3 (2010), 259–275.

[11] Bodlaender, H. L., and Koster, A. M. C. A. Treewidth computations II. Lower bounds. *Inf. Comput. 209*, 7 (2011), 1103–1119.

[12] Bui-Xuan, B.-M., Telle, J. A., and Vatshelle, M. Boolean-Width of Graphs. In *IWPEC* (2009), J. Chen and F. V. Fomin, Eds., vol. 5917 of *Lecture Notes in Computer Science*, Springer, p. 61–74.

[13] Cai, L., and Juedes, D. W. Subexponential Parameterized Algorithms Collapse the W-Hierarchy. In *ICALP* (2001), F. Orejas, P. G. Spirakis, and J. van Leeuwen, Eds., vol. 2076 of *Lecture Notes in Computer Science*, Springer, p. 273–284.

[14] Calamoneri, T. The L(h, k)-Labelling Problem: A Survey and Annotated Bibliography. *Comput. J. 49*, 5 (2006), 585–608.

[15] Chen, J., Kanj, I. A., and Jia, W. Vertex Cover: Further Observations and Further Improvements. In *WG* (1999), P. Widmayer, G. Neyer, and S. Eidenbenz, Eds., vol. 1665 of *Lecture Notes in Computer Science*, Springer, p. 313–324.

[16] Cook, S. A. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on the Theory of Computing* (New York, 1971), ACM, p. 151–158.

[17] Courcelle, B. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput. 85*, 1 (1990), 12–75.

[18] Courcelle, B., and Delhommé, C. The modular decomposition of countable graphs. Definition and construction in monadic second-order logic. *Theor. Comput. Sci. 394*, 1-2 (2008), 1–38.

[19] Courcelle, B., Makowsky, J. A., and Rotics, U. Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width. *Theory Comput. Syst. 33*, 2 (2000), 125–150.

[20] Courcelle, B., and Olariu, S. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics 101*, 1-3 (2000), 77–114.

[21] Dadush, D., Peikert, C., and Vempala, S. Enumerative Lattice Algorithms in any Norm Via M-ellipsoid Coverings. In *FOCS* (2011), R. Ostrovsky, Ed., IEEE, p. 580–589.

[22] Dasgupta, S., Papadimitriou, C. H., and Vazirani, U. V. *Algorithms.* McGraw-Hill, 2008.

[23] Diestel, R. *Graph Theory*, 4th ed. Graduate Texts in Mathematics, Volume 173. Springer-Verlag, Heidelberg, July 2010.

[24] Dom, M., Lokshtanov, D., Saurabh, S., and Villanger, Y. Capacitated Domination and Covering: A Parameterized Perspective. In *IWPEC* (2008), M. Grohe and R. Niedermeier, Eds., vol. 5018 of *Lecture Notes in Computer Science*, Springer, p. 78–90.

[25] Downey, R., and Fellows, M. *Parameterized complexity*, vol. 19. Springer New York, 1999.

[26] Edwards, K., and McDiarmid, C. The Complexity of Harmonious Colouring for Trees. *Discrete Applied Mathematics 57*, 2-3 (1995), 133–144.

[27] Fellows, M. R. Parameterized Complexity: The Main Ideas and Some Research Frontiers. In *ISAAC* (2001), P. Eades and T. Takaoka, Eds., vol. 2223 of *Lecture Notes in Computer Science*, Springer, p. 291–307.

[28] Fellows, M. R., Fomin, F. V., Lokshtanov, D., Rosamond, F. A., Saurabh, S., Szeider, S., and Thomassen, C. On the complexity of some colorful problems parameterized by treewidth. *Inf. Comput. 209*, 2 (2011), 143–153.

[29] Fellows, M. R., Lokshtanov, D., Misra, N., Rosamond, F. A., and Saurabh, S. Graph Layout Problems Parameterized by Vertex Cover. In *ISAAC* (2008), S.-H. Hong, H. Nagamochi, and T. Fukunaga, Eds., vol. 5369 of *Lecture Notes in Computer Science*, Springer, p. 294–305.

[30] Fiala, J., Golovach, P. A., and Kratochvíl, J. Distance Constrained Labelings of Graphs of Bounded Treewidth. In *ICALP* (2005), L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds., vol. 3580 of *Lecture Notes in Computer Science*, Springer, p. 360–372.

[31] Fiala, J., Golovach, P. A., and Kratochvíl, J. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theor. Comput. Sci. 412*, 23 (2011), 2513–2523.

[32] Flum, J., and Grohe, M. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[33] Fomin, F. V., Golovach, P. A., Lokshtanov, D., and Saurabh, S. Intractability of Clique-Width Parameterizations. *SIAM J. Comput. 39*, 5 (2010), 1941–1956.

[34] Frank, A., and Tardos, . An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica 7*, 1 (1987), 49–65.

[35] Frick, M., and Grohe, M. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic 130*, 1-3 (2004), 3–31.

[36] GAJARSKÝ, J. Efficient solvability of graph MSO properties [online]. Master's thesis, Masarykova univerzita, Fakulta informatiky, 2012 [cit. 2013-03-22].

[37] Gajarský, J., and Hliněný, P. Deciding Graph MSO Properties: Has it all been told already? *CoRR abs/1204.5194* (2012).

[38] Gajarský, J., and Hliněný, P. Faster Deciding MSO Properties of Trees of Fixed Height, and Some Consequences. In *FSTTCS* (2012), D. D'Souza, T. Kavitha, and J. Radhakrishnan, Eds., vol. 18 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, p. 112–123.

[39] Ganian, R. Using Neighborhood Diversity to Solve Hard Problems. *CoRR abs/1201.3091* (2012).

[40] Ganian, R., Hliněný, P., Nešetřil, J., Obdržálek, J., de Mendez, P. O., and Ramadurai, R. When Trees Grow Low: Shrubs and Fast MSO1. In *MFCS* (2012), B. Rovan, V. Sassone, and P. Widmayer, Eds., vol. 7464 of *Lecture Notes in Computer Science*, Springer, p. 419–430.

[41] Ganian, R., Hliněný, P., and Obdržálek, J. A unified approach to polynomial algorithms on graphs of bounded (bi-)rank-width. *Eur. J. Comb. 34*, 3 (2013), 680–701.

[42] Gottlob, G., Pichler, R., and Wei, F. Monadic datalog over finite structures of bounded treewidth. *ACM Trans. Comput. Logic 12* (November 2010), 3:1–3:48.

[43] Gurski, F., and Wanke, E. The Tree-Width of Clique-Width Bounded Graphs Without Kn, n. In *WG* (2000), U. Brandes and D. Wagner, Eds., vol. 1928 of *Lecture Notes in Computer Science*, Springer, p. 196–205.

[44] Gurski, F., and Wanke, E. Vertex Disjoint Paths on Clique-Width Bounded Graphs. In *LATIN* (2004), M. Farach-Colton, Ed., vol. 2976 of *Lecture Notes in Computer Science*, Springer, p. 119–128.

[45] Heinz, S. Complexity of integer quasiconvex polynomial optimization. *J. Complexity 21*, 4 (2005), 543–556.

[46] Hildebrand, R., and Köppe, M. A new Lenstra-type algorithm for quasiconvex polynomial integer minimization with complexity 2O(nlogn). *Discrete Optimization 10*, 1 (2013), 69–84.

[47] il Oum, S. Approximating Rank-Width and Clique-Width Quickly. In *WG* (2005), D. Kratsch, Ed., vol. 3787 of *Lecture Notes in Computer Science*, Springer, p. 49–58.

[48] Kaminski, M., Lozin, V. V., and Milanic, M. Recent developments on graphs of bounded clique-width. *Discrete Applied Mathematics 157*, 12 (2009), 2747–2761.

[49] Kannan, R. Minkowski's convex body theorem and integer programming. *Math. Oper. Res. 12*, 3 (Aug. 1987), 415–440.

[50] Karmarkar, N. A New Polynomial-Time Algorithm for Linear Programming. In *STOC* (1984), R. A. DeMillo, Ed., ACM, p. 302–311.

[51] Karp, R. M. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations* (1972), R. E. Miller and J. W. Thatcher, Eds., The IBM Research Symposia Series, Plenum Press, New York, p. 85–103.

[52] Khachiyan, L., and Porkolab, L. Integer Optimization on Convex Semialgebraic Sets. *Discrete & Computational Geometry 23*, 2 (2000), 207–224.

[53] Khachiyan, L. G. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR 244* (1979), 1093–1096.

[54] Kneis, J., Langer, A., and Rossmanith, P. Courcelle's Theorem - A Game-Theoretic Approach. *CoRR abs/1104.3905* (2011).

[55] Kolman, P., Lidický, B., and Sereni, J.-S. On Fair Edge Deletion Problems, 2009.

[56] Komlós, J. Covering Odd Cycles. *Combinatorica 17*, 3 (1997), 393–400.

[57] LAMBERT, V. Srovnání Vertex cover, Twin-cover a Neighborhood diversity na grafech [online], 2012 [cit. 2013-04-02].

[58] Lampis, M. Algorithmic Meta-theorems for Restrictions of Treewidth. *Algorithmica 64*, 1 (2012), 19–37.

[59] Langer, A., Reidl, F., Rossmanith, P., and Sikdar, S. Evaluation of an MSO-Solver. In *ALENEX* (2012), D. A. Bader and P. Mutzel, Eds., SIAM / Omnipress, p. 55–63.

[60] Langer, A., Rossmanith, P., and Sikdar, S. Linear-Time Algorithms for Graphs of Bounded Rankwidth: A Fresh Look Using Game Theory. *CoRR abs/1102.0908* (2011).

[61] Lenstra, H. Integer programming with a fixed number of variables. *Mathmatics of Operation Research 8* (1983), 538–548.

[62] Levin, L. A. Universal sequential search problems. *Problems of Information Transmission 9*, 3 (1973), 265–266.

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

[64] Lipton, R. J., and Tarjan, R. E. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics 36*, 2 (1979), 177–189.

[65] McDiarmid, C., and Reed, B. A. Channel assignment on graphs of bounded treewidth. *Discrete Mathematics 273*, 1-3 (2003), 183–192.

[66] Nešetřil, J., and de Mendez, P. O. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb. 27*, 6 (2006), 1022–1041.

[67] Nešetřil, J., and de Mendez, P. O. *Sparsity - Graphs, Structures, and Algorithms.*, vol. 28 of *Algorithms and combinatorics*. Springer, 2012.

[68] Niedermeier, R. Ubiquitous Parameterization - Invitation to Fixed-Parameter Algorithms. In *MFCS* (2004), J. Fiala, V. Koubek, and J. Kratochvíl, Eds., vol. 3153 of *Lecture Notes in Computer Science*, Springer, p. 84–103.

[69] Nishizeki, T., Vygen, J., and Zhou, X. The edge-disjoint paths problem is NP-complete for series-parallel graphs. *Discrete Applied Mathematics 115*, 1-3 (2001), 177–186.

[70] Oertel, T., Wagner, C., and Weismantel, R. Convex integer minimization in fixed dimension, Mar. 2012.

[71] Oum, S.-i., and Seymour, P. D. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B 96*, 4 (2006), 514–528.

[72] Robertson, N., and Seymour, P. D. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B 36*, 1 (1984), 49–64.

[73] Robertson, N., and Seymour, P. D. Graph Minors. II. Algorithmic Aspects of Tree-Width. *J. Algorithms 7*, 3 (1986), 309–322.

[74] Robertson, N., and Seymour, P. D. Graph Minors .XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B 63*, 1 (1995), 65–110.

[75] Röhrig, H. Tree Decomposition: A Feasibility Study. Master's thesis, Universität des Saarlandes, 1998.

[76] Szeider, S. Monadic second order logic on graphs with local cardinality constraints. *ACM Trans. Comput. Log. 12*, 2 (2011), 12.

[77] Yeh, R. K. A survey on labeling graphs with a condition at distance two. *Discrete Mathematics 306*, 12 (2006), 1217–1231.