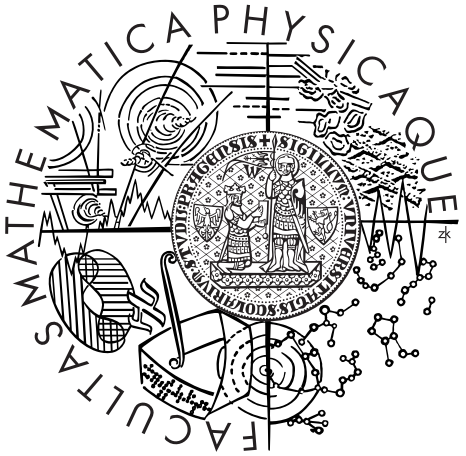
Charles University in Prague
Faculty of Mathematics and Physics

DOCTORAL THESIS



Martin Koutecký

Treewidth, Extended Formulations of CSP and MSO Polytopes, and their Algorithmic Applications

Department of Applied Mathematics

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

Study programme: Computer Science

Specialization: Discrete Models and Algorithms

Prague 2017

My greatest thanks go out to my advisor, Petr Kolman. For nine years he wooed and nudged me towards excellence, precision, and elegance, and has done so with supernatural patience. His genuine interest allowed me to rest in the certainty that he cares more about me than merely my performance. Hans Raj Tiwary got me interested in polytopes and optimization by the clarity of his intuition. Jiří Fiala introduced me to parameterized complexity and graph widths. Jiří Sgall was quick to point out proof flaws at seminars, but more importantly he made me feel I am always welcome to ask for advice, professional or otherwise.

Many thanks also to my colleagues from graduate school. Dušan Knop allowed me to flesh out my ideas aloud, pointed out problems and invented solutions. His attention to detail, clean notation and illuminating figures and examples inspire me. Tomáš Masařík and Tomáš Toufar have moved our research forward when I could not.

A large and pleasantly surprising experience for me have been research visits, and much gratitude belongs to my hosts. Petr Hliněný provided many logic-flavored insights and overall contributed so much more than I expected. Shmuel Onn greatly furthered my interest in optimization and integer programming and encouraged me at a time when I particularly needed it. Many thanks also to Josep Díaz, Iyad Kanj, Matthias Köppe, Jon Lee, Daniel Lokshtanov, Matthias Mnich, Saket Saurabh, and Maria Serna.

My work would not have been possible without the funding from the Czech Science Foundation (GA ČR), the Charles University Science Foundation (GA UK) and the Specific Academic Research Project grant (SVV). Many thanks also to the anonymous reviewers for pointing out existing work and improving ours with their feedback.

Finally, I am indebted to my family and friends. You know who you are: Mom & Dad, Tom, Jana & Pavel, Jordan, Jared, Keith & Rachel, Scott, Dave, Tom, Evča, Jan, Maru & Alan, Vláďa and many many others. Most of all, my love and respect goes to my wife Dana. She has chosen and continually chooses to sacrifice for me and submit to my lead, supports me and inspires me. She gives me perspective when I get sucked too close, and gives me focus when I get too distracted. I am a blessed man for having her.

I dedicate this thesis to Barunka. You taught me more about the meaning of family in a split second then I have ever known before. I cannot wait to meet you again.

Soli Deo Gloria

I declare that I carried out this doctoral 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 ..... date ..... signature of the author

Název práce: Stromová šířka, rozšířené formulace CSP a MSO polytopů a jejich algoritmické aplikace

Autor: Martin Koutecký

Katedra: Katedra aplikované matematiky

Vedoucí disertační práce: doc. Mgr. Petr Kolman, Ph.D., KAM MFF UK

Abstrakt: Tato práce podává důkaz existence kompaktních rozšířených formulací pro širokou škálu polytopů souvisejících s problémem omezujících podmínek (CSP), grafovou monadickou logikou druhého řádu (MSO) a rozšířeními MSO, mají-li dané instance omezenou stromovou šířku. Ukážeme, že naše rozšířené formulace mají další užitečné vlastnosti a odkrýváme souvislosti mezi MSO a CSP. Docházíme tak k závěru, že kombinace MSO logiky, CSP a geometrie poskytuje rozšiřitelný rámec pro konstrukci kompaktních rozšířených formulací a parametrizovaných algoritmů pro grafy s omezenou stromovou šířkou.

S použitím těchto nástrojů pak zcela zodpovíme otázku parametrizované složitosti různých rozšíření MSO na dvou třídách grafů, konkrétně grafech s omezenou stromovou šířkou a s omezenou různorodostí sousedství. Objevili jsme, že (ne)linearita těchto rozšíření určuje parametrizovanou složitost na grafech s omezenou různorodostí sousedství.

Na závěr studujeme tzv. posunutou kombinatorickou optimalizaci, která tvoří nelineární optimalizační rámec zobecňující standardní kombinatorickou optimalizaci. V této oblasti poskytneme prvotní zjištění z perspektivy parametrizované složitosti.

Klíčová slova: Rozšířená formulace, stromová šířka, MSO, CSP, kombinatorická optimalizace

Title: Treewidth, Extended Formulations of CSP and MSO Polytopes, and their Algorithmic Applications

Author: Martin Koutecký

Department: Department of Applied Mathematics

Supervisor: doc. Mgr. Petr Kolman, Ph.D., KAM MFF UK

Abstract: In the present thesis we provide compact extended formulations for a wide range of polytopes associated with the constraint satisfaction problem (CSP), monadic second order logic (MSO) on graphs, and extensions of MSO, when the given instances have bounded treewidth. We show that our extended formulations have additional useful properties, and we uncover connections between MSO and CSP. We conclude that a combination of the MSO logic, CSP and geometry provides an extensible framework for the design of compact extended formulations and parameterized algorithms for graphs of bounded treewidth.

Putting our framework to use, we settle the parameterized complexity landscape for various extensions of MSO when parameterized by two important graph width parameters, namely treewidth and neighborhood diversity. We discover that the (non)linearity of the MSO extension determines the difference between fixed-parameter tractability and intractability when parameterized by neighborhood diversity.

Finally, we study shifted combinatorial optimization, a new nonlinear optimization framework generalizing standard combinatorial optimization, and provide initial findings from the perspective of parameterized complexity.

Keywords: Extended formulation, treewidth, MSO, CSP, combinatorial optimization

Contents

Preface

I still remember how I have heard professor Martin Loebl describe Courcelle's theorem to our class, my first time hearing it. It was the summer semester of my first year at Charles University and I have somewhat naïvely signed up for a class on "Polyhedral Combinatorics and Mathematical Programming." Courcelle's theorem states that any problem expressible in a certain logical language (MSO) has an efficient algorithm on a certain graph class (graphs of bounded treewidth). Such a result seemed to me magical at the time, and even after eight years I am fascinated by its simplicity and power. That class was also my first encounter with my advisor, professor Petr Kolman. It was him who led us in our exploration of polytopes and their combinatorics.

The next year I was in a need of an advisor and a project. I turned to Petr, who was open and courageous enough to lead me in a project which ended up being my bachelor thesis. Its main topic, Courcelle's theorem.

In the following years I got even more interested in algorithms for specific graph classes and the whole area of parameterized complexity. Petr again agreed to advise my master thesis, even though it was not related to his main area of interest, which is network flows and cuts and approximation algorithms. Together with Dušan Knop, we have managed to design the first efficient algorithms for a few problems on so-called graphs of bounded neighborhood diversity.

When I approached Petr to once again become my advisor for my graduate studies, we have agreed that, this time, we will attempt to focus on Petr's main interest. We started experimenting with a linear programming relaxation for a certain cut problem. However, something strange happened: whatever graph we ran our relaxation on, the objective it returned was identical with the integer objective. After a while, we realized that all our graphs are so-called series-parallel graphs, and that our relaxation is always integral for such graphs. Eventuall, we were able to prove that a closely related linear program is an extended formulation of a CSP polytope, i.e. the polytope of feasible assignments of a constraint satisfaction problem, provided this CSP has bounded treewidth.

This way we have found ourselves, unintentionally, in the middle of the recently growing field of extension complexity. At the same time, we were (at least partially) back to my "home turf" of parameterized complexity, graph classes etc. I wondered: could we extend our CSP result to a "polytope analogue" of Courcelle's theorem? Petr, Hans Raj Tiwary and I have proved that this is so: there is a compact extended formulation of the MSO polytope, i.e. the polytope of satisfying assignments of an MSO formula on a graph of bounded treewidth. In the course of doing so, we have also rediscovered a "glued product" technique for constructing extended formulations.

Motivated by our experience with graphs of bounded neighborhood diversity, together with Dušan, Tomáš Masařík and Tomáš Toufar, we started wondering about Courcelle-like theorems on graphs of bounded treewidth or bounded neighborhood diversity, and for extensions of the MSO logic. About the same time, I have visited Shmuel Onn at the Technion and we explored an extension of the standard combinatorial optimization paradigm, called shifted combinatorial optimization (SCO). In particular, we were interested in optimizing over sets

definable by the MSO language.

To my surprise, both of these research directions lead me back to our results on MSO and CSP polytopes. I have discovered that the glued product operation behaves well with respect to certain useful polytope properties. Then, combining these results with CSP allowed us to show our Courcelle-like theorems for extensions of MSO, and some positive results for SCO.

While we explore here several additional topics and give much attention to complementing hardness results, the bulk of this work centers on the following thesis:

> *A combination of the* MSO *logic, CSP and geometry provides an extensible framework for the design of compact extended formulations and parameterized algorithms for graphs of bounded treewidth.*

The thesis is structured as follows. Chapter 1 first gives the reader a background in the areas covered by this thesis, then presents its main contributions, and finally reviews related work. This chapter is inteded for general audiences and attempts to stay away from technical details wherever possible. Then, Chapter 2 provides the necessary preliminaries such as definitions and notation for the following chapters. The first half of the results is presented in Chapters 3-5, where we develop connections between linear programming (LP), CSP and the MSO logic on graphs of bounded treewidth. This concludes with a new modeling tool useful for obtaining parameterized algorithms and compact extended formulations. Chapter 3 is based on a paper which appeared in the *Electronic Journal of Combinatorics* [70]. Chapters 4 and 5 are partially based on a paper which appeared at the *15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT'16)* [71].

Then, in Chapter 6 we unify prior work on extensions of the MSO logic and completely settle the complexity landscape on two graph classes by providing new positive and negative results. Moreover, the positive results carry over to extension complexity upper bounds. This chapter is mostly based on a paper which appeared at the *43rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'17)* [68]. Finally, in Chapter 7 we develop a new optimization framework generalizing standard combinatorial optimization and explore it from the perspective of parameterized complexity; this chapter is based on a paper which appeared at the *23rd Annual International Computing and Combinatorics Conference (COCOON'17)* [45]. In both chapters, we significantly use the tools developed in Chapter 5. We close with conclusions in Chapter 8.

1. Introduction

This chapter has three sections. First, we briefly introduce the notions which form the background and context of this thesis. Then, we give an overview of our contributions without going into too much formal detail; we make up for it in the subsequent chapters. Finally, we review other relevant work. Most formal definitions are postponed to Chapter 2.

1.1 MSO, CSP, Graph Widths and Extended Formulations

Parameterized Complexity

Our complexity viewpoint is that of *parameterized complexity*. Already in the 70's and 80's it was known that various computational problems belong to the class P of polynomially solvable problems when restricted to some special cases. For example, Lenstra [81] famously proved in 1983 that integer linear programming is solvable in polynomial time if the dimension is a fixed constant, and it was known that various NP-hard graph problems are solvable in polynomial time on trees. However, only in the 90's Downey and Fellows pioneered the field of parameterized complexity which is able to distinguish finer details.

Consider the example of the Vertex Cover and Independent Set problems, where we are to determine whether a vertex cover or an independent set of size k exists. On one hand, a $2^k + n$ algorithm exists for the Vertex Cover problem. On the other hand, it can be shown that under reasonable complexity assumptions the trivial n^k algorithm enumerating all independent sets is essentially optimal, ruling out the existence of *any* algorithm for Independent Set running in time[1] $f(k)n^{\mathcal{O}(1)}$. In the language of parameterized complexity, an algorithm solving problem Π in time $f(k)n^{\mathcal{O}(1)}$ is an FPT-algorithm (for *fixed-parameter tractable*) for Π with *parameter* k (or *parameterized by* k), and we say that Π is in FPT (or that Π is FPT) with respect to k. An analogue of NP-hardness in parameterized complexity is W[1]-hardness. W[1]-hard problems cannot have an algorithm running in time better than $n^{g(k)}$ for some computable function $g \in \omega(1)$ unless FPT = W[1], which is a commonly believed conjecture. An algorithm running in time $n^{g(k)}$ is an XP-algorithm (for *slice-wise polynomial*) and means that a problem belongs to XP (or is XP) with respect to k. Thus, while Vertex Cover is FPT parameterized by solution size, Independent Set is W[1]-hard and in XP. The key contribution here is in distinguishing problems which look the same from the perspective of classical complexity, i.e., NP-hard. For a thorough exposition, see the monograph of Downey and Fellows [30].

Graph Widths

Structural graph theory deals with establishing results that characterize various properties of graphs, and utilizes them in the design of efficient algorithms and

[1] We use the standard notation $\mathcal{O}(f), \omega(f)$: for two functions $g, f : \mathbb{N} \to \mathbb{N}$, $g \in \mathcal{O}(f)$ when $\exists c \in \mathbb{R}$ s.t. eventually $g \leq cf$, and $g \in \omega(f)$ when $\forall c \in \mathbb{R}$ eventually $g > cf$.

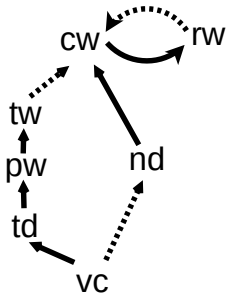


Figure 1.1: **Hierarchy of relevant parameters.** Included are vertex cover number, treedepth, pathwidth, treewidth, neighborhood diversity, cliquewidth and rankwidth. An arrow implies generalization, for example treewidth is a generalization of vertex cover number. A dashed arrow indicates that the generalization may increase the parameter exponentialy, for example treewidth k implies cliquewidth at most 2^k.

other applications. A subset of structural graph theory is the study of various *graph width* parameters. The fundamental example of a graph width is *treewidth* which constitutes a measure of "tree-likeness", and a graph of bounded treewidth can be thought of as a "fat tree"; we denote the treewidth of a graph G by $\mathrm{tw}(G)$. Generally, a graph width assigns a number to a graph, and we are typically interested in FPT or XP algorithms parameterized by a graph width. A graph width is typically associated with some kind of graph decomposition; for example, the fact that a graph G has treewidth τ is certified by a *tree decomposition* of width τ. Other parameters of graph structure which we consider here are the vertex cover number, treedepth, pathwidth, neighborhood diversity, cliquewidth and rankwidth, and for a graph G, we denote them by $\mathrm{vc}(G)$, $\mathrm{td}(G)$, $\mathrm{pw}(G)$, $\mathrm{nd}(G)$, $\mathrm{cw}(G)$ and $\mathrm{rw}(G)$, respectively. These parameters form a hierarchy, as depicted on Figure 1.1.

Our main focus in this thesis are graphs of bounded treewidth and graphs of bounded neighborhood diversity. These two classes are incomparable: for example, paths have unbounded neighborhood diversity but bounded treewidth, and vice versa for cliques. Treewidth has become a standard parameter with many practical applications (cf. a survey [12]); neighborhood diversity is of theoretical interest [1, 2, 13, 38, 47, 51, 90] because it can be viewed as representing the simplest of dense graphs.

Algorithmic Metatheorems

In the '70s and '80s, it was repeatedly observed that various NP-hard problems are solvable in polynomial time on graphs resembling trees. The graph property of *resembling a tree* was eventually formalized as having bounded treewidth, and in the beginning of the '90s, the class of problems efficiently solvable on graphs of bounded treewidth was shown by Courcelle [24] to contain the class of problems definable by the *Monadic Second Order Logic* (MSO); here, "efficiently solvable" means FPT with respect to $\mathrm{tw}(G)$ and the size of the MSO formula. This result is now known as Courcelle's theorem and it is a prototype of an algorithmic metatheorem: "logic L is decidable in time T on a graph class $\mathcal{G}$."

Courcelle's theorem has soon been extended to an optimization version (Arnborg et al. [3]) and MSO-evaluation (Courcelle and Mosbah [26]). Using similar techniques, analogous results for weaker logics were then proven for wider graph classes such as graphs of bounded cliquewidth and rankwidth [25]. Results of this kind are usually referred to as *Courcelle's theorem* for a specific class of

structures.

Monadic Second Order Logic. Let us shortly introduce MSO over graphs. In *first-order logic* (FO) we have variables for the elements $(x, y, \dots)$, equality for variables, quantifiers $\forall, \exists$ ranging over elements, and the standard Boolean connectives $\not, \wedge, \vee, \implies$. MSO is the extension of FO by quantification over sets $(X, Y, \dots)$. Graph MSO has the binary relational symbol $\mathrm{edge}(x, y)$ encoding edges, and traditionally comes in two flavours, MSO_1 and MSO_2, differing by the objects we are allowed to quantify over: in MSO_1 these are the vertices and vertex sets, while in MSO_2 we can additionally quantify over edges and edge sets. For example, the 3-colorability property can be expressed in MSO_1 as follows:

$$\exists X_1, X_2, X_3 \qquad [\forall x\,(x \in X_1 \vee x \in X_2 \vee x \in X_3) \wedge \bigwedge\nolimits_{i=1,2,3} \forall x, y\,(x \notin X_i \vee y \notin X_i \vee \neg\,\mathrm{edge}(x, y))\,]$$

For a formula φ, we denote by $|\varphi|$ the *size* (number of symbols) of φ. We briefly remark that MSO_2 can express properties which are not MSO_1 definable (e.g., Hamiltonicity). The relationship of MSO_1 and MSO_2 on various graph classes is somewhat complicated (cf. Subsection 6.1.2). Typically, on graphs of bounded treewidth, by MSO we mean MSO_2, and on graphs of bounded cliquewidth and bounded neighborhood diversity, by MSO we mean MSO_1.

Courcelle's theorem proliferated into many fields. Originating among automata theorists, it has since been reinterpreted in terms of finite model theory [83], database programming [53] and game theory [67]. Note the result of Gottlob et al. [53] which shows that Courcelle's theorem can be reinterpreted in the realm of database programming. We highlight it because it can be phrased as an *expressivity* result: "the database language Datalog can efficiently express MSO queries on bounded treewidth graphs."

Lower bounds. Another research direction was to improve the computational complexity of Courcelle's theorem. Its complexity is $f(|\varphi|, \tau) \cdot n$, which is FPT with respect to $|\varphi|$ and $\tau = \mathrm{tw}(G)$, but the function f grows as an exponential tower in the quantifier depth of the MSO formula. Unfortunatelly, Frick and Grohe [44] proved that this is unavoidable unless $\mathsf{P} = \mathsf{NP}$, which raises a question: is there a (nontrivial) graph class where MSO model checking can be done in single-exponential (i.e. $2^{k^{\mathcal{O}(1)}}$) time? This was answered in the affirmative by Lampis [75] for graphs of bounded neighborhood diversity: given a graph G, an MSO_1 formula φ can be decided in time $2^{\mathrm{nd}(G)|\varphi|} \cdot n$.

MSO partitioning. A related problem is MSO PARTITIONING, where we are given an MSO formula φ with one free set variable X, a graph G and an integer r, and the task is to partition $V(G)$ or $E(G)$ (depending on φ being an MSO_1 or MSO_2 formula) into r sets, all of which satisfy φ; we use the standard notation $V(G)$ for the vertices of G and $E(G)$ for its edges. Rao [103] showed that MSO_1 PARTITIONING is in XP with respect to $\mathrm{cw}(G)$ and the size of the formula.

Constraint Satisfaction Problem

The *constraint satisfaction problem (CSP)* is a powerful modeling tool. We consider a very general setting where an instance I of CSP consists of a set of *variables* V, a set of their *domains* $\mathcal{D}$, a set of *hard constraints* $\mathcal{H}$ and a set

of *soft constraints* $\mathcal{S}$. The hard constraints induce the set of feasible solutions $\text{Feas}(I)$ and the soft constraints then define an objective function over $\text{Feas}(I)$. The goal is to find a feasible solution with optimal objective value. The modeling power of CSP is large: it naturally captures many graph problems, SATISFIABILITY, INTEGER LINEAR PROGRAMMING (ILP), problems in scheduling, planning, databases, machine vision, belief maintenance, temporal reasoning, type reconstruction, and many other areas of artificial intelligence; cf. a complexity-focused review of Hell and Nešetřil [58].

As CSP captures many NP-hard problems, it is a natural problem to identify tractable special cases of CSP. Structural graph theory has been of use in this area as well. For example, Samer and Szeider [106] study the parameterized complexity of CSP with respect to the treewidth of various graphs related to a CSP instance. The most natural graph is the *primal graph* $G(I)$ (also known as *constraint graph*), which has a vertex for each variable and an edge between two variables if they appear together in a constraint. We let $\text{tw}(I) = \text{tw}(G(I))$ be the *treewidth of* I. Already in 1990, Freuder [43] showed that CSP instances whose constraint graph has treewidth bounded by τ and whose maximum domain size is D can be solved in time $O(D^\tau n)$. Later, Grohe et al. [55] proved that, assuming $FPT \neq W[1]$, this is essentially the only nontrivial class of graphs for which CSP is solvable in polynomial time (also cf. Marx [89]).

Extended Formulations

In recent years, a lot of attention has been given to study the *extension complexity* of problems [22]: given a problem Q, what is the minimum number of inequalities representing a polytope whose (suitably chosen) linear projection coincides with the convex hull H of all integral solutions of Q? Any polytope which projects to H is called an *extended formulation* (or an *extension*) of H. The question at hand is essentially an *expressivity* questions: is linear programming capable of expressing H compactly? Indeed, the pioneering result of Yannakakis [117] from 1991 was motivated by many claims that the Travelling Salesman Problem (TSP) polytope is compactly expressible by a certain kind of linear programs, which Yannakakis refuted. This notion of expressivity has also been extended to semidefinite programming (SDP). Note that membership of a problem in the class P does not necessarily imply the existence of an extended formulation of polynomial size: Rothvoss [105] shows that the polytope of matchings has exponential extension complexity, in spite of MAXIMUM MATCHING belonging to P. On the other hand, the existence of an extended formulation of a problem Π of polynomial size trivially implies that Π belongs to P.

1.2 Our Contribution

1.2.1 Chapter 3: The CSP Polytope

The main result of Chapter 3 is the following theorem:

Theorem 3.0.1. *There exists an extended formulation of the* CSP polytope $CSP(I)$ *that can be described by* $\mathcal{O}(D^\tau \cdot n)$ *inequalities, where* $n = |V|$, D *is the largest domain size, and* $\tau = \text{tw}(I)$.

Notice that, in a sense, this is an expressivity result: a certain subclass of CSP instances can be compactly expressed with linear programming. Our proof technique in this chapter is the following. We directly provide an LP related to $CSP(I)$ and prove that every vertex of this LP is integral. Thus, this LP defines an extended formulation of the CSP polytope. This result was the first meta-theorem in the theory of extended formulations, providing an extension complexity upper bound for several important NP-hard problems on graphs of bounded treewidth. Prior to our result, there was only a failed attempt [110] at proving the same result and several *ad hoc* proofs of specializations of our result for various graph problems [18, 79]; cf. Section 1.3.

1.2.2 Chapter 4: The MSO Polytope

In Chapter 4, we prove an analogue of Courcelle's theorem for polytopes. We say that a set S is MSO-*definable* if $S = S_\varphi(G)$ is the set of satisfying assignments of an MSO formula φ on a graph G, represented in an appropriate space by their characteristic vectors. Then, let $P_\varphi(G) = \text{conv}(S_\varphi(G))$ be the polytope obtained as the convex hull of $S_\varphi(G)$. We call it the MSO *polytope*.

Theorem 4.3.1. *There exists an extended formulation of $P_\varphi(G)$ that can be described by $f(|\varphi|, \tau) \cdot n$ inequalities, where n is the number of vertices in G, τ is the treewidth of G and f is a computable function depending only on φ and τ.*

In other words, we prove that the extension complexity of $P_\varphi(G)$ is linear in the size of the graph G, with a constant depending on the treewidth of G and the formula φ.

This again provides a very general yet very simple meta-theorem about the extension complexity of polytopes related to a wide class of problems and graphs; we note that even though there is a certain equivalence between the expressivity of MSO and CSP (cf. the next subsection), there are problems which are naturally expressible in one way but not in the other.

In sharp contrast with the previous chapter, our proof technique here does not directly consider an LP representing the extension P of $P_\varphi(G)$. Rather, we develop a geometric tool which we term the *glued product of polytopes*, and use it to obtain P in a bottom-up way by gluing smaller polytopes.

Our proof essentially works by "merging the common wisdom" from the areas of extended formulations and parameterized complexity. It is known that dynamic programming can usually be turned into a compact extended formulation (Martin et al. [88] and Kaibel [61]), and that Courcelle's theorem can be seen as an instance of dynamic programming [74], and therefore it should be expected that the polytope of satisfying assignments of an MSO formula of a bounded treewidth graph be small.

However, there are a few roadblocks in trying to merge these two folklore wisdoms. For one, while Courcelle's theorem being an instance of dynamic programming in some sense may be obvious to a parameterized complexity theorist, it is far from clear to anyone else what that sentence may even mean. On the other hand, being able to turn a dynamic program into a compact polytope may be a theoretical possibility for an expert on extended formulations, but it is by no means an easy statement for an outsider to comprehend. What complicates

the matters even further is that the result of Martin et al. [88] is not a result that can be used in a black box fashion. That is, a certain condition must be satisfied to get a compact extended formulation out of a dynamic program. This is far from a trivial task, especially for a theorem like Courcelle's.

1.2.3 Chapter 5: Connecting MSO, CSP, and Treewidth

Chapter 5 ties together all of our previous research on polytopes, MSO, and CSP, and provides a new perspective on designing algorithms and extended formulations for bounded treewidth graphs.

We start by studying the glued product of polytopes in more depth. In particular, we study the decomposability of polyhedra: a polyhedron P is *decomposable* if any integer point in its r*-dilate* $rP = \{r\mathbf{x} \mid \mathbf{x} \in P\}$ can be written as a sum of r integer points from P; rP is essentially P blown up by a factor of r. We show that the glued product of two decomposable polyhedra is decomposable. Another property of the glued product is preserving the treewidth of matrices[2] of LPs defining the glued polytopes. Using these results, we are able to show:

Theorem 5.1.9. *There is an extended formulation of $P_\varphi(G)$ of size $f(|\varphi|, \tau) \cdot n$, where $\tau = \mathrm{tw}(G)$, which is decomposable and can be represented by an LP whose defining matrix has treewidth bounded by $f(|\varphi|, \tau)$ for some computable function f.*

The importance of decomposability stems from the fact that if rP is decomposable, then its integer points represent multisets of integer points of P of cardinality r (or r*-multisets* for short). Recall the MSO PARTITIONING problem mentioned above: using these notions we can essentially recast it as searching for a decomposable integer point $\mathbf{x} \in rP_\varphi(G)$ with an objective function enforcing that $\mathbf{x}$ corresponds to a partition (cf. Chapter 7).

Furthermore, we show two interesting connections between Chapters 3 and 4. On one hand, we show that an extension $P(I)$ of the CSP polytope $CSP(I)$ can be constructed using the glued product, which proves additional properties about it. Furthermore, we study the complexity of realizing an integer separable minimization oracle for its r-dilate $rP(I)$. We shall introduce this concept now. An *integer separable function* is a function $f : \sum_{i=1}^{n} f_i(x_i)$ with $f_i : \mathbb{N} \to \mathbb{R}$ for all i. An *integer separable minimization oracle* for a set of points P is an algorithm which, queried on an integer separable function f, returns an integer point $\mathbf{x} \in P$ minimizing $f(\mathbf{x})$, or reports that P is empty or unbounded.

Theorem 5.2.3. *The CSP polytope $CSP(I)$ has a decomposable extended formulation $P(I)$ of size $\mathcal{O}(D^\tau \cdot n)$ whose defining matrix has treewidth bounded by a function of $\tau = \mathrm{tw}(I)$ and D, and an integer separable minimization oracle for $rP(I)$ can be realized in time $\min(D^{\mathcal{O}(r\tau)}, r^{D^{\mathcal{O}(\tau)}}) \cdot \|I\|$, where $\|I\|$ is the encoding length of I.*

On the other hand, we show that Courcelle's theorem can be recast as an instance of CSP of bounded treewidth. To that end, we introduce the notion of

[2]The treewidth of a matrix $\mathbf{A}$ is the treewidth of its *Gaifman graph*, which has a vertex for each column of $\mathbf{A}$ and two vertices $i \neq j$ are connected if a row $\mathbf{a}$ of $\mathbf{A}$ exists s.t. both a_i and a_j are nonzero.

a CSP extended formulation. Similarly as with LPs, a CSP *extended formulation* (or *extension*) J of a CSP I is a CSP instance whose variables V_J are a superset of the variables V_I of I, and $\mathrm{Feas}(J)$ coincides with $\mathrm{Feas}(I)$ after discarding these extra variables. Then we have:

Lemma 5.2.8. *Let I be a CSP instance with $\mathrm{Feas}(I) = S_\varphi(G)$. Then I has a CSP extended formulation J of treewidth 2 and with domain sizes bounded by $f(|\varphi|, \mathrm{tw}(G))$ for some computable function f.*

Applying Theorem 3.0.1 to this instance then provides an alternative proof of Theorem 4.3.1.

In our applications, we typically wish to construct a CSP instance I with $\mathrm{Feas}(I) \subseteq S_\varphi(G)$, i.e., its solutions are satisfying assignments of an MSO formula φ complying with some additional constraints. We say that these additional constraints have the *local scope property* if they can be expressed by hard constraints which are restricted to variables corresponding to vertices of G contained in adjacent nodes of the tree decomposition of G. For $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$. Then, we are able to extend Lemma 5.2.8 as follows:

Lemma 5.2.10. *Let G be a graph with $\mathrm{tw}(G) = \tau$ and (T, B) be its optimal tree decomposition, φ be an MSO_2 formula with m free variables, and k an integer. Let I be a CSP instance with variables $\{y_v^i \mid v \in V(G), i \in [m]\}$ and $\{x_a^j \mid a \in V(T), j \in [k]\}$, with a hard constraint $\mathbf{y} \in S_\varphi(G)$ and additional hard constraints $\mathcal{H}'$ over $\mathbf{y}$ and $\mathbf{x}$ satisfying the local scope property. Then I has an extended formulation J of treewidth bounded by a function of $|\varphi|$, τ, and k, and with domain sizes bounded by a function of $|\varphi|$, τ, and the domain sizes of $\mathbf{x}$.*

Let us return to integer separable minimization oracles. Our motivation for studying them is the following. Let $S \subseteq \mathbb{N}^n$ be a set; then let $rS = \{\mathbf{x}^1 + \cdots + \mathbf{x}^r \mid \mathbf{x}^1, \ldots, \mathbf{x}^r \in S\}$, and let $\left(\!\binom{S}{r}\!\right)$ be the set of multisubsets of S of size r. It is clear that rS and $\left(\!\binom{S}{r}\!\right)$ are closely related, because $rS = \{\mathbf{x}^1 + \cdots + \mathbf{x}^r \mid \{\mathbf{x}^1, \ldots, \mathbf{x}^r\} \in \left(\!\binom{S}{r}\!\right)\}$, but they are not completely equivalent, because one element of rS may correspond to multiple elements of $\left(\!\binom{S}{r}\!\right)$. Abusing our terminology slightly, we say that there is an *integer separable minimization oracle over the r-multisubsets of S* if there is an integer separable minimization oracle for rS, and, together with a minimum $\mathbf{x}$, it returns its *decomposition* $\mathbf{x}^1 + \cdots + \mathbf{x}^r = \mathbf{x}$.

Then, combining Lemma 5.2.10 with Theorem 5.2.3 and utilizing this close connection between integer separable minimization over $rCSP(I)$ and the r-multisubsets of $\mathrm{Feas}(I)$, we show our "Master Theorem:"

Theorem 5.2.13 (Master MSO-CSP Theorem)**.** *Let I be a CSP instance as in Lemma 5.2.10. Then $\mathrm{xc}(CSP(I)) \leq (f(|\varphi|, \tau) + 2k) \cdot \|I\|$ and it is possible to realize an integer separable minimization oracle over the decomposable points of $rCSP(I)$ and over the r-multisubsets of $\mathrm{Feas}(I)$ in time $\min((f(|\varphi|, \tau) + k)^r, r^{f(|\varphi|,\tau)+k}) \cdot \|I\|$.*

In summary, we have brought together notions from optimization (CSP, integer separable minimization), geometry (decomposability), graph theory (treewidth) and logic (Courcelle's theorem). We believe that this combination

is unique and has its merit for the following reason. We have overviewed much work on extensions and reinterpretations of Courcelle's theorem. Our observation is that for these results, it seems unavoidable to either start from scratch, or to pull in many foreign notions (at least foreign to a researcher primarily familiar with classical graph theory and optimization) such as automata, finite model theory, etc. In this sense, the prior work resembles powerful black boxes. We believe that, in contrast, our work provides an approachable and extensible framework. Our main thesis can thus be stated as this:

> *A combination of* MSO*, CSP and geometry provides an extensible framework for the design of extended formulations and parameterized algorithms for graphs of bounded treewidth.*

The following chapters attempt to demonstrate this claim but also explore topics beyond problems on bounded treewidth graphs.

1.2.4 Chapter 6: Extensions of MSO Logic

As already mentioned, several extensions of Courcelle's theorem have been proven in the 90's. However, some important graph problems do not admit an MSO description and are W[1]-hard and thus unlikely to be solvable in FPT time on graphs of bounded treewidth. This led to examination of extensions of MSO which allow greater expressive power. The contribution of this chapter is twofold. First, we survey and enrich the so far studied extensions of MSO logic:

- CardMSO (Ganian and Obdržálek [49], which places cardinality constraints on the free set variables,
- fairMSO (Kolman et al. [72]), which asks to optimize a "fair" objective function, and,
- MSO-LCC (Szeider [111], which, for every vertex, places cardinality constraints on how many of its neighbors can belong to which free set variables.

While both MSO-LCC and CardMSO express certain *cardinality* constraints, the constraints of CardMSO are inherently *global* and *linear*, yet the constraints of MSO-LCC are *local* and *nonlinear*. This leads us to introduce two more fragments and rename the aforementioned ones: CardMSO becomes $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$, MSO-LCC becomes $\mathsf{MSO}^{\mathsf{L}}$ and we additionally have $\mathsf{MSO}^{\mathsf{G}}$ and $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$. Furthermore, we have $\mathsf{MSO}^{\mathsf{GL}}_{\mathsf{lin}}$, the combination of $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$ and $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$, and $\mathsf{MSO}^{\mathsf{GL}}$, the combination of $\mathsf{MSO}^{\mathsf{G}}$ and $\mathsf{MSO}^{\mathsf{L}}$. $\mathsf{MSO}^{\mathsf{GL}}$ thus represents the most expressive fragment under our consideration. By this we give a complete landscape for all possible combinations of global/local and linear/nonlinear. It is known that $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$ generalizes fairMSO, and that already fairMSO is W[1]-hard on graphs of bounded treewidth. Thus, we disregard fairMSO from now on.

Second, we study the parameterized complexity of the associated model checking problem for all of the newly introduced fragments. We completely settle the parameterized complexity landscape for the model checking problems with respect to the parameters treewidth and neighborhood diversity. Refer to Figure 1.2 for a joint overview of both treewidth and neighborhood diversity, and Figure 1.3,

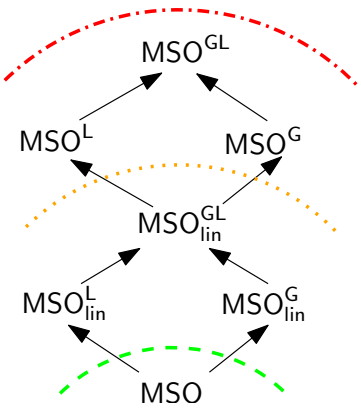


Figure 1.2: MSO **extensions.** A partial order of MSO extensions considered. An arrow denotes generalizations; e.g., $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$ generalizes MSO and is generalized by $\mathsf{MSO}^{\mathsf{GL}}_{\mathsf{lin}}$. Green (dashed) line separates logics whose model checking is FPT parameterized by $\mathrm{tw}(G)$ (Courcelle [24]) from those whose model checking is W[1]-hard (both $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$ and $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$ capture the W[1]-hard Equitable r-Coloring problem). Orange (dotted) line separates logics whose model checking is FPT parameterized by $\mathrm{nd}(G)$ (Theorem 6.3.5) from those whose model checking is W[1]-hard (Theorems 6.3.1 and 6.3.2). The model checking of all logics below the red (dashed-dotted) line is XP parameterized by both $\mathrm{tw}(G)$ (Theorem 6.2.1) and $\mathrm{nd}(G)$ (Theorem 6.3.15).

which gives a more detailed summary of our and prior work separately for the two parameters. We postpone formal definitions of logic extensions and corresponding model checking to Chapter 6.

Our results in Chapter 6 could be summarized by saying that,

- on graphs of bounded treewidth, even the weakest MSO extensions are already W[1]-hard, but even the strongest are still in XP, while
- on graphs of bounded neighborhood diversity, linear constraints make problems FPT, while nonlinear constraints make them W[1]-hard, and even the strongest MSO extension is still in XP.

One important contribution here is that we have identified the source of hardness for neighborhood diversity (or, indeed, already vertex cover number): the *non*linearity of the constraints. For graphs of bounded treewidth we prove using Theorem 5.2.13 the following.

Theorem 6.2.1. $\mathsf{MSO}^{\mathsf{GL}}$ Model Checking *is* XP *parameterized by* $\mathrm{tw}(G)$ *and* $|\varphi|$.

Theorem 5.2.13 allows us to formulate the proof of Theorem 6.2.1 as providing a certain CSP instance, surpassing the typical complexity of a dynamic programming formulation. We believe that the key advantage of our approach when compared with prior work is that it is fairly *declarative*: we state what a solution looks like and implicitely (by CSP constraints) describe how to obtain it along a tree decomposition, but we do not describe the algorithm computing it. This makes the proof cleaner and possible extensions easier. Moreover, since the proof uses Theorem 5.2.13 we immediatelly obtain extension complexity upper bounds as a corollary:

Corollary 6.2.2. *Let G be a graph of treewidth τ, I be an instance of* $\mathsf{MSO}^{\mathsf{GL}}$ Model Checking *and $P_\varphi(G)$ be the* $\mathsf{MSO}^{\mathsf{GL}}$ *polytope of satisfying assignments of instance I. Then* $\mathrm{xc}(P_\varphi(G)) \leq n^{f(|\varphi|,\tau)}$ *for some computable function f.*

Afterwards, we briefly discuss how these resuts can be applied:

nd, vc	$\emptyset$	fairMSO	$\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$	$\mathsf{MSO}^{\mathsf{L}}$
MSO	FPT [75]	FPT [90]		W[1]-h, Thm 6.3.2
$\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$	FPT [49]		FPT, Thm 6.3.5	
$\mathsf{MSO}^{\mathsf{G}}$	W[1]-h, Thm 6.3.1			XP, Thm 6.3.15

tw	$\emptyset$	fairMSO	$\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$	$\mathsf{MSO}^{\mathsf{L}}$
MSO	FPT [24]	W[1]-hard [90]		XP [111]
$\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$	W[1]-hard [49]			
$\mathsf{MSO}^{\mathsf{G}}$				XP, Thm 6.2.1

Figure 1.3: **MSO extensions tables.** Each cell of a table corresponds to a logic fragment formed by combining the fragments of the respective row and column. Moving right and down in the table gives a more general MSO extension. Thus, positive results (FPT, XP) spread to the left and up and W[1]-hardness spreads to the right and down. Green background (lighter gray) stands for FPT fragments, while orange (darker gray) stands for W[1]-hard and XP. The first table provides positive results when parameterized by $\mathrm{nd}(G)$ and negative results when parameterized by $\mathrm{vc}(G)$. The second table deals with parameterization by $\mathrm{tw}(G)$.

Theorem 6.2.3. *Let G be a graph of treewidth τ and with $n = |V(G)|$.*

(XP) *The following problems have algorithms with runtime $n^{f(\tau)}$ and extended formulations of the same size:* General Factor, Minimum Maximum Outdegree, Capacitated Dominating Set, Capacitated Vertex Cover, Vector Dominating Set, Generalized Domination.

(FPT) *The following problems have algorithms with runtime $f(\tau + k) \cdot n^{\mathcal{O}(1)}$ and extended formulations of the same size, with k specified further:*

- Equitable r-Coloring, Equitable connected r-Partition, r-Balanced Partitioning, *with $k = r$,*
- Graph Motif, *with $k = \chi$, where χ is the number of colors.*

For graphs of bounded neighborhood diversity we give two positive results.

Theorem 6.3.5. $\mathsf{MSO}^{\mathsf{GL}}_{\mathsf{lin}}$ Model Checking *is* FPT *parameterized by* $\mathrm{nd}(G)$ *and* $|\varphi|$.

Theorem 6.3.15. $\mathsf{MSO}^{\mathsf{GL}}$ Model Checking *is* XP *parameterized by* $\mathrm{nd}(G)$ *and* $|\varphi|$.

We complement the above results with a hardness results for $\mathsf{MSO}^{\mathsf{L}}$ and $\mathsf{MSO}^{\mathsf{G}}$ already when parameterizing by $\mathrm{vc}(G)$.

Theorem 6.3.2. $\mathsf{MSO}^{\mathsf{L}}$ Model Checking *is* W[1]-hard *parameterized by* $\mathrm{vc}\,(G)$ *and* $|\varphi|$.

Theorem 6.3.1. $\mathsf{MSO}^{\mathsf{G}}$ Model Checking *is* W[1]-hard *parameterized by* $\mathrm{vc}(G)$ *and* $|\varphi|$.

Interestingly, our finding about hardness being caused by nonlinearity in the case of neighborhood diversity carries over to a generalization of the Set Cover problem. In the Multidemand Set Multicover problem, we are given a *universe* $U = [k]$ of size k, *multidemands* $d_i \subseteq \mathbb{N}$ for all $i \in [k]$, a *covering system* represented by a multisubset $\mathcal{F} \subseteq 2^U$ with a *weight* w_F associated with each $F \in \mathcal{F}$. The task is to find a multisubset $\mathcal{F}' \subseteq \mathcal{F}$ of minimum weight which satisfies the demands, that is, for each $i \in [k]$, $|\{F \in \mathcal{F}' \mid i \in F\}| \in d_i$. We show that the (non)linearity of the multidemands is crucial:

Corollary 6.3.4. Multidemand Set Multicover *is* W[1]-hard *parameterized by* k.

Proposition 7.1.4. Multidemand Set Multicover *is* FPT *parameterized by* k *when each* d_i *is an interval.*

1.2.5 Chapter 7: Shifted Combinatorial Optimization

In the last chapter, we study a wide generalization of standard combinatorial optimization from the perspective of parameterized complexity. *Shifted combinatorial optimization* is a new nonlinear optimization framework which is a broad extension of standard combinatorial optimization, involving the choice of several feasible solutions at a time. This framework captures well studied and diverse problems ranging from so-called vulnerability problems to sharing and partitioning problems. In particular, every standard combinatorial optimization problem has its shifted counterpart, which is typically much harder. Already with explicitly given input set the shifted problem may be NP-hard. In this chapter we study the parameterized complexity of this framework.

Specifically, the following problem has been studied extensively in the literature.

(Standard) Combinatorial Optimization. Given $S \subseteq \{0,1\}^n$ and $\mathbf{w} \in \mathbb{Z}^n$, solve

$$\max\{\mathbf{ws} \mid \mathbf{s} \in S\} \ . \tag{1.1}$$

The complexity of the problem depends on $\mathbf{w}$ and the type and representation of S. Often, S is the set of indicating (characteristic) vectors of members of a family of subsets over a ground set $[n]$, such as the family of s–t dipaths in a digraph with n arcs, the set of perfect matchings in a bipartite or arbitrary graph with n edges, or the set of bases in a matroid over $[n]$ given by an independence oracle.

Partly motivated by *vulnerability* problems studied recently in the literature [4, 94] (as we shall discuss further on), we study a broad nonlinear extension of Combinatorial Optimization, in which the optimization is over r choices of elements of S and which is defined as follows. For a set $S \subseteq \mathbb{R}^n$, let S^r denote the set of $n \times r$ matrices having each column in S,

$$S^r \ = \ \{\mathbf{x} \in \mathbb{R}^{n\times r} \mid \mathbf{x}^1, \ldots, \mathbf{x}^r \in S\} \ .$$

Call $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n\times r}$ equivalent and write $\mathbf{x} \sim \mathbf{y}$ if each row of $\mathbf{x}$ is a permutation of the corresponding row of $\mathbf{y}$. The *shift* of $\mathbf{x} \in \mathbb{R}^{n\times r}$ is the unique matrix $\overline{\mathbf{x}} \in \mathbb{R}^{n\times r}$

satisfying $\overline{\mathbf{x}} \sim \mathbf{x}$ and $\overline{\mathbf{x}}^1 \geq \cdots \geq \overline{\mathbf{x}}^r$, that is, the unique matrix equivalent to $\mathbf{x}$ with each row nonincreasing. Our nonlinear optimization problem follows:

Shifted Combinatorial Optimization (SCO). Given $S \subseteq \{0,1\}^n$ and $\mathbf{c} \in \mathbb{Z}^{n \times r}$, solve

$$\max\{\mathbf{c}\overline{\mathbf{x}} \mid \mathbf{x} \in S^r\} \ . \tag{1.2}$$

(Here $\mathbf{c}\overline{\mathbf{x}}$ is used to denote the ordinary scalar product of the vectors $\mathbf{c}$ and $\overline{\mathbf{x}}$.)

Motivation

This problem easily captures many classical fundamental problems. For example, given a graph $G = (V, E)$ with n vertices, let $S = \{N[v] \mid v \in V\} \subseteq \{0,1\}^n$, where $N[v]$ is the characteristic vector of the closed neighborhood of v, i.e. $N[v]$ contains v. Choose an integer parameter r and let $c_i^1 = 1$ for all i and $c_i^j = 0$ for all i and all $j \geq 2$. Then the optimal objective function value of (1.2) is n if and only if we can select a set D of r vertices in G such that every vertex belongs to the closed neighborhood of at least one of the selected vertices, that is, when D is a dominating set of G. Likewise, one can formulate the vertex cover and independent set problems in a similar way.

Vulnerability. One specific motivation for the SCO problem is the following. Suppose S is the set of indicators of members of a family over $[n]$. A feasible solution $\mathbf{x} \in S^r$ then represents a choice of r members of the given family such that the k-th column $\mathbf{x}^k$ is the indicator of the k-th member. Call element i in the ground set *k-vulnerable* in $\overline{\mathbf{x}}$ if it is used by at least k of the members represented by $\mathbf{x}$, that is, if the i-th row $\mathbf{x}_i$ of $\mathbf{x}$ has at least k ones. It is easy to see that the k-th column $\overline{\mathbf{x}}^k$ of the shift of $\mathbf{x}$ is precisely the indicator of the set of k-vulnerable elements in $\mathbf{x}$. So the shifted optimization problem is to maximize

$$\mathbf{c}\overline{\mathbf{x}} \ = \ \sum\{c_i^k \mid i \text{ is } k\text{-vulnerable in } \mathbf{x},\ i \in [n]\,,\ k \in [r]\} \ .$$

Minimizing the numbers of k-vulnerable elements in $\mathbf{x}$ may be beneficial for survival of some family members under various attacks to vulnerable elements by an adversary, see e.g. [4, 94] for more details. For example, to minimize the number of k-vulnerable elements for some k, we set $c_i^k = -1$ for all i and $c_i^j = 0$ for all i and all $j \neq k$. To *lexicographically* minimize the numbers of r-vulnerable elements, then of $(r-1)$-vulnerable elements, and so on, till that of 1-vulnerable elements, we can set $c_i^k = -(n+1)^{k-1}$ for all i, k.

Partitioning and coloring. As another natural example, consider $\mathbf{c}$ with $c_i^1 = 1$ and $c_i^j = -1$ for $1 < j \leq r$. Then $\mathbf{c}\overline{\mathbf{x}} = n$ if and only if the columns of $\mathbf{x}$ represent a *partition* of S. This formulation hence allows us to optimize over partitions of the ground set. Or, consider $\mathbf{c}$ with $\mathbf{c}_i = (1, \ldots, 1, -1, \ldots, -1)$ of length $a > 0$ with $b \leq a$ ones, and let S be the family of independent sets of a graph G. Then $\max \mathbf{c}\overline{\mathbf{x}}$ relates to *fractional coloring of* G; it holds that $\max \mathbf{c}\overline{\mathbf{x}} = bn$ if and only if G has a coloring by a colors in total such that every vertex receives b distinct colors – this is the so-called $(a:b)$-COLORING problem.

Congestion game. Another interpretation of the problem is in terms of minimizing social cost in a *congestion game* [104]. We are given a set $S \subseteq \{0,1\}^n$ of indicators of members of a family over ground set $[n]$, representing a set of *strategies*. For $i \in [n]$ we are now given a function $f_i : [r] \to \mathbb{Z}$. Each $\mathbf{x} \in S^r$

represents a choice of r players with $\mathbf{x}^j \in S$ the choice of player j. The *congestion* of $\mathbf{x}$ at element $i \in [n]$ is the number $\sum_{j=1}^{r} x_{i,j}$ of players using i in $\mathbf{x}$. The cost of $\mathbf{x}$ at i is the value $f_i(\sum_{j=1}^{r} x_{i,j})$ of f_i on the congestion at i. The social cost of $\mathbf{x}$ is $\sum_{i=1}^{n} f_i(\sum_{j=1}^{r} x_{i,j})$ and we want to find $\mathbf{x} \in S^r$ minimizing social cost. For instance, S may be the set of s–t dipaths in a digraph; each player chooses a dipath; and the cost at edge i may be an increasing function f_i of the congestion at i. Now define $\mathbf{c} \in \mathbb{Z}^{n \times r}$ by $c_{i,j} := f_i(j-1) - f_i(j)$ for all i and j. Then for every $\mathbf{x} \in S^r$ we have $\mathbf{c}\overline{\mathbf{x}} = \sum_{i=1}^{n} f_i(0) - \sum_{i=1}^{n} f_i(\sum_{j=1}^{r} x_{i,j})$ and so $\mathbf{x}$ maximizes $\mathbf{c}\overline{\mathbf{x}}$ if and only if it minimizes the social cost. So the congestion and shifted problems are equivalent. A case of special interest is when the f_i are *convex*, implying the rows of $\mathbf{c}$ are nonincreasing, that is, $\mathbf{c} = \overline{\mathbf{c}}$ is *shifted*.

Clearly, the set rS defined previously is related to the set S^r. Roughly, rS corresponds to multisubsets of S, while S^r corresponds to r-tuples of elements of S. Without going into details, we remark that this difference is significant. Because the shift operator is indifferent to the order of the tuple $\mathbf{x}$, SCO is more closely related to optimizing over rS than S^r. We make this distinction clearer in Chapter 7.

Prior results

The complexity of the SCO problem depends on $\mathbf{c}$ and on the representation of S, and is typically harder than the corresponding standard combinatorial optimization problem. Say, when S is the set of perfect matchings in a graph, the standard problem is polynomial time solvable, but the shifted problem is NP-hard even for $r = 2$ and cubic graphs, as the optimal value of the above 2-vulnerability problem is 0 if and only if the graph is 3-edge-colorable [82]. The minimization of 2-vulnerable arcs with S the set of s–t dipaths in a digraph, also called the Minimum shared edges problem, was recently first shown to be NP-hard for r variable in [94], then XP with respect to r [4], and finally FPT with respect to r [41].

In the rest of this chapter we always assume that the number r of choices is variable. Call a matrix $\mathbf{c} \in \mathbb{Z}^{n \times r}$ *shifted* if $\mathbf{c} = \overline{\mathbf{c}}$, that is, if its rows are nonincreasing. In [63] it was shown that when $S = \{\mathbf{s} \in \{0,1\}^n \mid \mathbf{A}\mathbf{s} = \mathbf{b}\}$ where $\mathbf{A}$ is a totally unimodular matrix and $\mathbf{b}$ is an integer vector, the shifted problem with shifted $\mathbf{c}$, and hence in particular the above lexicographic vulnerability problem, can be solved in polynomial time. In particular this applies to the cases of S the set of s–t dipaths in a digraph and S the set of perfect matchings in a bipartite graph. In [82] it was shown that the shifted problem with shifted $\mathbf{c}$ is also solvable in polynomial time for S the set of bases of a matroid presented by an independence oracle (in particular, spanning trees in a graph), and even for the intersection of matroids of certain type.

Our results

Our results can be summarized as follows. First we show that whether SCO over an explicitly given set S parameterized by $|S|$ belongs to the class XP, FPT, or P depends on the objective function. Particularly, in the FPT case we use the fact that convex integer minimization is FPT parameterized by the dimension, which so far has few applications in the literature. Second, we study the shifted

problem over MSO-definable sets (which includes, e.g., the MSO PARTITIONING problem). Our main results here are that shifted combinatorial optimization over MSO-definable sets is in XP with respect to the MSO formula and the treewidth (or more generally cliquewidth) of the input graph, and is W[1]-hard even when the formula is first-order and the graph has bounded treedepth.

More specifically, we first consider the case when the set S is given explicitely. While the standard problem is always trivial in such case, the SCO problem can be NP-hard for an explicit set S (Proposition 7.1.1). Our main results in this case can be briefly summarized as follows:

Theorem 7.1.2. *The shifted combinatorial optimization problem parameterized by $|S| = m$ is;*

- *for general* $\mathbf{c}$ *in* XP *and* W[1]-hard,
- *for shifted* $\mathbf{c}$ *in* FPT, *and,*
- *for shifted* $-\mathbf{c}$ *in* P.

Then, we study a more general framework of SCO for MSO-definable sets, i.e. $S = S_\varphi(G)$. We give a result generalizing known results about MSO PARTITIONING:

Theorem 7.2.2. *The shifted combinatorial optimization problem, for*

- *graphs of bounded treewidth and S defined by $\varphi \in \mathsf{MSO}_2$, or*
- *graphs of bounded cliquewidth and S defined by $\varphi \in \mathsf{MSO}_1$,*

is XP *parameterized by the width and $|\varphi|$.*

The proof of this statement connects SCO to integer separable minimization and then uses Theorem 5.2.13. To complement the previous tractability result, we then prove the following negative result under much more restrictive parametrization.

Theorem 7.3.1. *There exists a fixed First Order formula ϕ such that the associated* MSO_1 PARTITIONING *problem, and hence also the SCO problem with $S = S_\phi(G)$, are* W[1]-hard *on graphs of bounded treedepth.*

Thus the result of Theorem 7.2.2 is in a sense best possible.

1.3 Related Work

1.3.1 CSP

Many important combinatorial optimization problems belong to the class of constraint satisfaction problems (CSP) [58]. Naturally, a lot of effort has been given to design efficient approximation algorithms for CSP, to prove complexity lower bounds for CSP, and to identify "islands of tractability" of CSP, recently especially from the point of view of parameterized complexity.

A prominent way of defining islands of tractability for CSP is to restrict the relations that may occur in the constraints to a fixed set Γ, called a *constraint*

language. The main standing[3] open problem in this direction is the dichotomy conjecture of Feder and Vardi [34].

A very active area of research takes a different approach than restricting the constraint language and instead considers the constraint graph of a CSP instance. Besides the aforementioned results of Freuder [43], Grohe et al. [55], Marx [89] or Samer and Szeider [106], we highlight for example the recent work of Ganian et al. [50] on combining treewidth with backdoors. A *backdoor* is essentially a set of variables whose valuation turns the instance into a tractable one.

1.3.2 Extended Formulations

A lot of recent work on extended formulations has focused on establishing lower bounds in various settings: exact, approximate, linear vs. semidefinite, etc. (see for example [40, 6, 14, 80]). A wide variety of tools have been developed and used for these results including connections to nonnegative matrix factorizations [117], communication complexity [33], information theory [15], and quantum communication [40], among others.

For proving upper bounds on extended formulations, several authors have proposed various tools as well. Kaibel and Loos [62] describe a setting of branched polyhedral systems which was later used by Kaibel and Pashkovich [64] to provide a way to construct polytopes using reflection relations.

A particularly specific composition rule which we term glued product (cf. Section 4.2) was studied by Margot in his PhD thesis [87]. Margot showed that a property called the projected face property suffices to glue two polytopes efficiently. Conforti and Pashkovich [23] describe and strengthen Margot's result to make the projected face property to be a necessary and sufficient condition to describe the glued product in a particularly efficient way.

Martin et al. [88] have shown that under certain conditions, an efficient dynamic programming based algorithm can be turned into a compact extended formulation. Kaibel [61] summarizes this and various other methods.

CSP and graphs of bounded treewidth. Describing the polytope of CSP solutions by the means of linear programming for instances of bounded treewidth is not a new idea. In 2007, Sellmann et al. published a paper [110] in which they described a linear program that was supposed to define the convex hull of all feasible solutions of a binary CSP when the constraint graph is a tree. They also provided a procedure to convert a given CSP instance with bounded treewidth into one whose constraint graph is a tree, at the cost of blowing up the number of variables and constraints by a function of the treewidth. Unfortunately, there was a substantial bug in their proof and one of the main theorems in the paper does not even hold [109].

The paper [110] also implicitly includes this folklore result: if the constraint graph has treewidth at most τ, then CSP can be solved by τ levels of the Sherali-Adams hierarchy (we are not aware of an explicit formulation of this result in its generality though partial results of this type are known, e.g., for independent set [9]). The resulting formulation is of size $\mathcal{O}(n^\tau)$ while our approach yields size $\mathcal{O}(D^\tau n)$. This formulation seems to have recently been rediscovered by Braun et

[3] Although probably not for long; cf. recent claims for proof [19, 99, 119]

al. [16]. They give a uniform extended formulation for the MATCHING, INDEPENDENT SET and VERTEX COVER polytopes of size $\mathcal{O}(n^\tau)$, where *uniform* means that the same extended formulation can be used for any graph, and a given graph is only encoded in the objective function; this makes our extended formulations nonuniform.

Laurent [79] provides extended formulations for the INDEPENDENT SET and the CUT polytopes, both a special case of CSP, that have size $O(2^\tau n)$ where τ denotes the treewidth of the given graph. These results are given in the context of moment matrices as an application of a sparsity structure present in instances of bounded treewidth. Independently, Buchanan and Butenko [18] later gave the same result for the INDEPENDENT SET polytope. Our results can be viewed as a generalization: the size of our formulation for a general CSP, when applied to the INDEPENDENT SET and the CUT polytopes, is also $O(2^\tau n)$.

In a recent work, independently of our result, Bienstock and Munoz [8] define a class of so called *general binary* optimization problems which are essentially weighted boolean CSP problems, and, among other results, for instances of treewidth τ provide an LP formulation of size $O(2^\tau n)$. Again, this is a special case of our Theorem 3.0.1. It is worth mentioning at this point that every CSP instance can be transformed into a boolean CSP instance by encoding every variable with domain size D by $\lceil \log D \rceil$ boolean variables. This increases the treewidth of the constraint graph by a factor of $\lceil \log D \rceil$ and thus leads to a formulation of size $\mathcal{O}(2^{\tau \lceil \log D \rceil} n)$. This bound corresponds to $\mathcal{O}(D^\tau n)$ in the special case that D is a power of two, and to $\mathcal{O}(D^\tau 2^\tau n)$ in the general case. Compared to their approach, we describe directly the polytope of the original problem and our proof is self-contained, without relying on other techniques; also, our results apply not only for binary CSP but for any CSP.

CSP and general graphs. Chan et al. [20] study the extent to which linear programming relaxations can be used in dealing with approximating CSP. They show that polynomial-sized LPs are exactly as powerful as LPs obtained from a constant number of rounds of the Sherali-Adams hierarchy. They also prove integrality gaps for polynomial-sized LPs for some CSP.

Raghavendra [100] shows that under the Unique Games Conjecture, a certain simple SDP relaxation achieves the best approximation ratio for every CSP. In a follow up paper, Raghavendra and Steurer [101] describe an efficient rounding scheme that achieves the integrality gap of the simple SDP relaxation, and, in another paper [102], they show unconditionally that the integrality gap of this SDP relaxation cannot be reduced by Sherali-Adams hierarchies.

1.3.3 Algorithmic Metatheorems

Connecting different fields. Because of the wide relevance of the treewidth parameter in many areas (cf. survey by Bodlaender [12]) and the large expressivity of MSO and its extensions (cf. the surveys of Langer et al. [78] and Grohe and Kreutzer [54]), considerable attention was given to Courcelle's theorem by theorists from various fields, reinterpreting it into their own setting. These reinterpretations helped uncover several interesting connections.

The classical way of proving Courcelle's theorem is constructing a tree automaton A in time only dependent on φ and the treewidth τ, such that A accepts

a tree decomposition of a graph of treewidth τ if and only if the corresponding graph satisfies φ; this is the automata theory perspective [24]. Another perspective comes from finite model theory [83] where one can prove that a certain equivalence on the set of graphs of treewidth at most τ has only finitely many (depending on φ and τ) equivalence classes and that it *behaves well* [53]. (This is the approach we chose to build on.) Another approach proves that a quite different equivalence on so-called extended model checking games has finitely many equivalence classes [67] as well; this is the game-theoretic perspective. It can be observed that the finiteness in either perspective stems from the same roots.

The result of Gottlob et al. [53] can also be viewed as an expressivity result: they prove that on bounded treewidth graphs, a certain subset of the database query language Datalog has the same expressive power as MSO. This provides an interesting connection between automata theory and database theory. Our results can be seen as an analogue connecting automata theory to LP and CSP.

Practicality of Courcelle's theorem. Because of its unavoidably restrictive time complexity bounds, Courcelle's theorem has long been considered a classification-only tool. It was argued (e.g. by Niedermeier [92] or more recently by Cygan et al. [28]) that proving a problem to be FPT on bounded treewidth graphs using Courcelle's theorem is only the first step, and one should afterwards turn to designing a problem-specific algorithm attaining reasonable runtimes. However, based on the recent work of Kneis et al. [67] and Langer et al. [78], an MSO solver was implemented and evaluated [77], and it was found that it beats commercial ILP solvers on several real-world instances. The work of the research group of Woltran (e.g. [10]) can be seen as a continuation of these efforts to bring the theory behind Courcelle's theorem to practical settings. For these reasons, we think it is no longer fair to claim that Courcelle's theorem is merely of theoretical interest.

Extensions of Courcelle's Theorem

Objective functions. A linear optimization version of Courcelle's theorem was given by Arnborg, Lagergren and Seese [3]. An extension to further objectives was given by Courcelle and Mosbah [26]. Kolman, Lidický and Sereni [72] introduce MSO with a fair objective function (fairMSO) which, for a given MSO formula $\varphi(F)$ with a free edge set variable F, minimizes the maximum degree in the subgraph given by F, and present an XP algorithm. This is justified by the problem being W[1]-hard, as was later shown by Masařík and Toufar [90], who additionally give an FPT algorithm on graphs of bounded neighborhood diversity for MSO_1 and an FPT algorithm on graph of bounded vertex cover for MSO_2.

Extended logics. Along with MSO, Courcelle also considered *counting* MSO (cMSO) where predicates of the form "$|X| \equiv p \mod q$" are allowed, with the largest modulus q constant. Szeider [111] introduced MSO with *local cardinality constraints* (MSO-LCC) and gave an XP algorithm deciding it on graphs of bounded treewidth. MSO-LCC can express various problems, such as General Factor, Equitable r-Coloring or Minimum Maximum Outdegree, which are known to be W[1]-hard on graphs of bounded treewidth. Ganian and Obdržálek [49] study CardMSO, which is incomparable with MSO-LCC in its ex-

pressive power; they give an FPT algorithm on graphs of bounded neighborhood diversity.

2. Preliminaries

In this chapter, we introduce the central notions and associated notation. Let n be a non-negative integer; by $[n]$ we denote the set $\{1, \ldots, n\}$. For two integers a, b we define the set $[a, b] = \{x \in \mathbb{Z} \mid a \leq x \leq b\}$.

We write vectors of numbers in boldface, e.g., $\mathbf{x}, \mathbf{y}$ etc., and thus distinguish them from their entries, written in normal font. The i-th entry of $\mathbf{x}$ is denoted by x_i, $x(i)$ or $x[i]$. For a vector $\mathbf{x}$ and a subset I of coordinates, we denote by $\mathbf{x}|_I$ the *projection of* $\mathbf{x}$ *to* I, which is the subvector of $\mathbf{x}$ specified by the coordinates I. We write tuples of other objects (e.g. vertices, vertex sets etc.) as $\vec{p} = (p_1, \ldots, p_k)$. By $\mathrm{supp}(\mathbf{x}) = \{i \mid x_i \neq 0\}$ we denote the *support of* $\mathbf{x}$, that is, the set of its nonzero coordinates. By $\mathbf{A}_j$ we denote the j-th row of a matrix $\mathbf{A}$. Let $x \in \mathbb{Z}$; by $\langle x \rangle = \log_2 x + 1$ we denote the *encoding length* of x, and extend this notion naturally to vectors (for $\mathbf{x} \in \mathbb{Z}^n$, $\langle \mathbf{x} \rangle = \sum_{i=1}^{n} \langle x_i \rangle$), matrices (for $\mathbf{A} \in \mathbb{Z}^{n \times m}$, $\langle \mathbf{A} \rangle = \sum_{j=1}^{m} \langle \mathbf{A}_j \rangle$) and functions with finite domains (for $f : D \to \mathbb{R}$, $\langle f \rangle = \sum_{i \in D} \langle f(i) \rangle$). For simplification, we sometimes write the dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$ incorrectly not as $\mathbf{x}^\intercal \mathbf{y}$, but as $\mathbf{xy}$; the expression $\mathbf{xy}$ is always a dot product.

2.1 Parameterized Complexity

A *parameterized problem* Q is a subset of $\Sigma^* \times \mathbb{N}_0$, where Σ is a finite alphabet. A parameterized problem Q is said to be *fixed-parameter tractable* if there is an algorithm that given $(x, k) \in \Sigma \times \mathbb{N}_0$ decides whether (x, k) is a YES-instance of Q in time $f(k) \cdot p(|x|)$ where f is some computable function of k alone, p is a polynomial and $|x|$ is the size measure of the input. The class of such problems is denoted by FPT. The class XP is the class of parameterized problems that admit algorithms with a run-time of $|x|^{f(k)}$ for some computable function f, i.e. polynomial-time for every fixed value of k.

The theory of parameterized complexity also defines complexity classes W[t] for $t \geq 1$, where W[t] $\subseteq$ XP for all integers $t \geq 1$. For instance, the k-INDEPENDENT SET problem (with parameter k) is complete for W[1]. Problems that are W[1]-hard do not admit an FPT algorithm unless the *Exponential Time Hypothesis* (ETH) fails, which is considered unlikely.

2.2 Graphs and General Relational Structures

We use standard notation for graphs, see e.g. Diestel's book [29]. For a vertex $v \in V$ of a graph $G = (V, E)$, we denote by $N_G(v)$ the *neighborhood of* v *in* G, that is, $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$; the subscript G is omitted when clear from the context. For a rooted tree T, $N_T(v)$ denotes the *down-neighborhood of* v *in* G, i.e., the set of descendants of v. For general directed graphs, $N_G^+(v)$ and $N_G^-(v)$ denotes the *in-neighborhood and out-neighborhood of* v *in* G, respectively.

Regarding more general relational structures, in most cases we stick to standard notation as given by Libkin [83]. A *vocabulary* σ is a finite collection of *constant* symbols $c_1, c_2, \ldots$ and *relation* symbols $P_1, P_2, \ldots$. Each relation sym-

bol P_i has an associated arity r_i. A *σ-structure* is a tuple $\mathcal{A} = (A, \{c_i^{\mathcal{A}}\}, \{P_i^{\mathcal{A}}\})$ that consists of a universe A together with an interpretation of the constant and relation symbols: each constant symbol c_i from σ is associated with an element $c_i^{\mathcal{A}} \in A$ and each relation symbol P_i from σ is associated with an r_i-ary relation $P_i^{\mathcal{A}} \subseteq A^{r_i}$.

To give an example, a graph $G = (V, E)$ can be viewed as a σ_1-structure $(V, \emptyset, \{E\})$ where E is a symmetric binary relation on $V \times V$ and the vocabulary σ_1 contains a single relation symbol. Alternatively, for another vocabulary σ_2 containing three relation symbols, one of arity two and two of arity one, we can view a graph $G = (V, E)$ also as a σ_2-structure $I(G) = (V_I, \emptyset, \{E_I, L_V, L_E\})$, with $V_I = V \cup E$, $E_I = \{\{v, e\} \mid v \in e, e \in E\}$, $L_V = V$ and $L_E = E$; we will call $I(G)$ the *incidence graph* of G.

2.3 Graph Widths

2.3.1 Treewidth and Pathwidth

For notions related to the treewidth of a graph, we mostly stick to the standard terminology as given in the book of Kloks [66]; the only deviation is in the leaf nodes of the nice tree decomposition where we assume that the bags are empty.

Definition 2.3.1 (Tree decomposition, Treewidth)**.** *A* tree decomposition *of a graph $G = (V, E)$ is a pair (T, B), where T is a tree and B is a mapping $B : V(T) \to 2^V$ satisfying*

Edge condition: *for any $uv \in E$, there exists $a \in V(T)$ such that $u, v \in B(a)$,*

Connectedness condition: *if $v \in B(a)$ and $v \in B(b)$, then $v \in B(c)$ for all c on the path from a to b in T.*

We use the convention that the vertices of the tree are called nodes *and the sets $B(a)$ are called* bags*. Occasionally, we will view the mapping B as the set $B = \{B(u) \mid u \in V\}$.*

The treewidth $\mathrm{tw}((T, B))$ of a tree decomposition (T, B) *is the size of the largest bag of (T, B) minus one. The* treewidth $\mathrm{tw}(G)$ of a graph G *is the minimum treewidth over all possible tree decompositions of G.*

Before moving on to more refined notions, we also mention the closely related parameter pathwidth.

Definition 2.3.2 (Pathwidth)**.** *A* path decomposition *of a graph G is its tree decomposition (T, B) such that T is a path. The* pathwidth $\mathrm{pw}(G)$ of a graph G *is the minimum treewidth over all possible path decompositions of G.*

Definition 2.3.3 (Nice tree decomposition)**.** *A* nice tree decomposition *is a tree decomposition in which T is a rooted tree and each node is one of the following types:*

Leaf node: *a leaf a of T with $B(a) = \emptyset$.*

Introduce node: *an internal node a of T with one child b for which $B(a) = B(b) \cup \{v\}$ for some $v \in B(a)$; for short we write $a = b * (v)$.*

Forget node: *an internal node a of T with one child b for which $B(a) = B(b) \setminus \{v\}$ for some $v \in B(b)$; for short $a = b \dagger (v)$.*

Join node: *an internal node a with two children b and c with $B(a) = B(b) = B(c)$; for short $a = \Lambda(b, c)$.*

For a vertex $v \in V$, we denote by $\mathrm{top}(v)$ the topmost node of the nice tree decomposition (T, B) that contains v in its bag. For any graph G of treewith τ on n vertices, a nice tree decomposition of G of width τ with at most $8n$ nodes can be computed in time $f(\tau) \cdot n$, for some computable function f: Bodlaender's algorithm [11] can find an optimal tree decompositon, and [66, Lemma 13.1.3] then converts it into a nice tree decomposition.

Given a graph $G = (V, E)$ and a subset of vertices $\{v_1, \ldots, v_d\} \subseteq V$, we denote by $G[\{v_1, \ldots, v_d\}]$ the subgraph of G induced by the set $\{v_1, \ldots, v_d\}$. Given a tree decomposition (T, B) and a node $a \in V(T)$, we denote by T_a the subtree of T rooted in a, and by G_a the subgraph of G induced by all vertices in bags of T_a, that is, $G_a = G[\bigcup_{b \in V(T_a)} B(b)]$. Throughout this thesis we assume that for every graph, its vertex set is a subset of $\mathbb{N}$. We define the following operator η: for any set $U = \{v_1, v_2, \ldots, v_k\} \subseteq \mathbb{N}$, $\eta(U) = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ such that $v_{i_1} < v_{i_2} \cdots < v_{i_k}$.

Definition 2.3.4 (Gaifman graph)**.** *Given a relational structure $\mathcal{A} = (A, \mathcal{S})$ with $\mathcal{S} \subseteq 2^A$, its* Gaifman graph *is the graph $G(\mathcal{A}) = (A, E)$ where $E = \{\{u, v\} \mid \exists S \in \mathcal{S} : u, v \in S\}$.*

The *Gaifman graph* $G(\mathbf{A})$ associated with a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the Gaifman graph of the structure $([n], \{\mathrm{supp}(\mathbf{A}_i) \mid i \in [m]\})$ where $\mathbf{A}_i$ is the i-th row of $\mathbf{A}$. In other words, the graph $G(\mathbf{A})$ has a vertex for each column of $\mathbf{A}$ and two vertices are connected by an edge if the supports of the corresponding columns have non-empty intersection.

Definition 2.3.5 (Treewidth of a matrix)**.** *The* treewidth of a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, *denoted* $\mathrm{tw}(A)$*, is the treewidth of its Gaifman graph. The* treewidth of a system of inequalities $\mathbf{Ax} \leq \mathbf{b}$ *is defined as* $\mathrm{tw}(\mathbf{A})$.

2.3.2 Treedepth

Definition 2.3.6 (Treedepth [91])**.** *Let the* height *of a rooted tree or forest be the maximum root-to-leaf distance in it. The* closure $\mathrm{cl}(F)$ *of a rooted forest F is the graph obtained from F by making every vertex adjacent to all of its ancestors. The* treedepth $\mathrm{td}(G)$ *of a graph G is one more than the minimum height of a forest F such that $G \subseteq \mathrm{cl}(F)$.*

Note that always $\mathrm{td}(G) \geq \mathrm{tw}(G) + 1$ since we can use the vertex sets of the root-to-leaf paths of the forest F in a proper order as the bags of a tree-decomposition of G.

2.3.3 Cliquewidth

Definition 2.3.7 (Cliquewidth [27])**.** *Let G be a graph. The* cliquewidth of G *is the smallest number of labels $\gamma = \mathrm{cw}(G)$ such that some labeling of G can*

be constructed by an algebraic γ-expression *using the following operations (where* $1 \leq i, j \leq \gamma$*):*

1. *create a new vertex with label* i*;*
2. *take the disjoint union of two labeled graphs;*
3. *add all edges between vertices of label* i *and label* j*; and*
4. *relabel all vertices with label* i *to have label* j*.*

The parameter *rankwidth* [96] has an involved definition which we skip here, because we never directly work with graphs of bounded rankwidth. It is significant to us only since it is currently an open problem whether an optimal γ-expression can be computed for a graph with $\mathrm{cw}(G) = \gamma$. However, an optimal rank-decomposition of G *can* be computed in FPT time using an algorithm of Hliněný and Oum [59], and can be used as an approximation of a γ-expression with up to an exponential jump, which does not matter for a fixed parameter γ in theory.

2.3.4 Neighborhood Diversity

We say that two (distinct) vertices u, v are of the same *neighborhood type* if they share their respective neighborhoods, that is, when $N(u) \setminus \{v\} = N(v) \setminus \{u\}$.

Definition 2.3.8 (Neighborhood decomposition)**.** *Let* $G = (V, E)$ *be a graph. We call a partition of its vertices* $\mathcal{T} = \{T_1, \ldots, T_\nu\}$ *a* neighborhood decomposition *if, for every* $i \in [\nu]$*, all vertices of* T_i *are of one neighborhood type. We call the sets* $T_1, \ldots, T_\nu$ types*.*

Note that every type induces either a clique or an independent set in G and two types are either joined by a complete bipartite graph or no edge between vertices of the two types is present in G. Thus, we introduce the notion of a *type graph* $T_{\mathcal{T}}(G)$. The vertices of $T_{\mathcal{T}}(G)$ are the types $T_1, \ldots, T_\nu$ and two types T_i, T_j are joined by an edge if T_i and T_j are joined by a complete bipartite graph in G. If the decomposition $\mathcal{T}$ is clear from the context, we omit the subscript $\mathcal{T}$.

Definition 2.3.9 (Neighborhood diversity [75])**.** *A graph* $G = (V, E)$ *has* neighborhood diversity ν *(*$\mathrm{nd}(G) = \nu$*) if its minimal neighborhood decomposition is of size* ν*.*

The definition above is sound, as it is known [75] that a minimal neighborhood decomposition is unique. Moreover, it can be computed in linear time.

For completeness, we introduce one more graph parameter. For a graph $G = (V, E)$, a set $U \subseteq V$ is a *vertex cover* of G if for every edge $e \in E$ it holds that $e \cap U \neq \emptyset$.

Definition 2.3.10 (Vertex cover number)**.** *A graph* G *has* vertex cover number k *(*$\mathrm{vc}(G) = k$*) if its minimum vertex cover is of size* k*.*

Recall that these graph parameter form a hierarchy as presented in Figure 1.1.

2.4 Constraint Satisfaction Problem (CSP)

Definition 2.4.1 (CSP instance)**.** A CSP instance $I = (V, \mathcal{D}, \mathcal{H}, \mathcal{S})$ *consists of*

- *a set of* variables z_v*, one for each* $v \in V$*; without loss of generality we assume that* $V = [n]$*, but variables can have arbitrary names in formulations,*
- *a set* $\mathcal{D}$ *of finite* domains D_v *(also denoted* $D(v)$*), one for each* $v \in V$*,*
- *a set of* hard constraints $\mathcal{H} \subseteq \{C_U \mid U \subseteq V\}$ *where each hard constraint* $C_U \in \mathcal{H}$ *with a* scope $U = \{i_1, i_2, \ldots, i_k\} \subseteq V$ *and* $i_1 < i_2 < \cdots < i_k$*, is a* $|U|$*-ary relation* $C_U \subseteq D_{i_1} \times D_{i_2} \times \cdots \times D_{i_k}$*,*
- *a set of* soft constraints $\mathcal{S} \subseteq \{C_U \mid U \subseteq V\}$ *where each soft constraint* $C_U \in \mathcal{S}$ *with a* scope $U = \{i_1, i_2, \ldots, i_k\} \subseteq V$ *and* $i_1 < i_2 < \cdots < i_k$*, is a* $|U|$*-ary relation* $C_U \subseteq D_{i_1} \times D_{i_2} \times \cdots \times D_{i_k}$*.*

A vector $\mathbf{z} \in \mathbb{Z}^n$ *satisfies the hard constraint* $C_U \in \mathcal{H}$ if and only if $\mathbf{z}|_U \in C_U$. We say that a vector $\mathbf{z}^\star = (z_1^\star, \ldots, z_n^\star)$ is *a feasible assignment* for I if $\mathbf{z}^\star \in D_1 \times \ldots \times D_n$ and $\mathbf{z}^\star$ satisfies every hard constraint $C \in \mathcal{H}$; let $\mathrm{Feas}(I)$ be the set of all feasible assignments of I. We denote by D_I the largest size of all domains, that is, $D_I = \max_{u \in V} |D_u|$, and we omit the subscript I if the instance is clear from the context. Finally, we denote by $\|\mathcal{D}\|$, $\|\mathcal{H}\|$ and $\|\mathcal{S}\|$ the *length of* $\mathcal{D}$*,* $\mathcal{H}$ *and* $\mathcal{S}$, respectively, and define it as $\|\mathcal{D}\| = \sum_{v \in V} |D_v|$, $\|\mathcal{H}\| = \sum_{C_U \in \mathcal{H}} |C_U|$ and $\|\mathcal{S}\| = \sum_{C_U \in \mathcal{S}} |C_U|$. Analogously, we let $\|I\| = \|\mathcal{D}\| + \|\mathcal{H}\| + \|\mathcal{S}\|$ be the *length of* I*.*

Decision / Max / Min CSP. In the *decision* version of CSP, the set $\mathcal{S}$ of soft constraints is empty and the task is to decide whether there exists a feasible assignment. In the *maximization* (*minimization*, resp.) version of the problem, the task is to find a feasible assignment that maximizes (minimizes, resp.) the number of *satisfied* (unsatisfied, resp.) soft constraints. Note that there is no difference between maximization and minimization versions of the problem with respect to optimal solutions but the two versions differ significantly from an approximation perspective.

Weighted CSP. In the *weighted* version of CSP we are also given a weight function $w : \mathcal{S} \to \mathbb{R}$ that specifies for each soft constraint $C \in \mathcal{S}$ its weight $w(C)$. The goal is to find a feasible assignment that maximizes (minimizes, resp.) the total weight of satisfied (unsatisfied, resp.) constraints. The unweighted version of CSP is equivalent to the weighted version with $w(C) = 1$ for all $C \in \mathcal{S}$.

Even more generally, the relations in the soft constraints can be replaced by bounded real valued payoff functions: a soft constraint $w_U \in \mathcal{C}$ with $U = \{i_1, i_2, \ldots, i_k\}$ is not a $|U|$-ary relation but a function $w_U : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \to \mathbb{R}$ and the payoff of the soft constraint w_U for a feasible assignment $\mathbf{z}^\star$ is $w_U(z^\star|_U)$; the objective is to maximize (minimize, resp.) the total payoff. Then, $\|\mathcal{S}\| = \sum_{w_U} |w_U|$, where $|w_U|$ denotes the size of the subset of $D_{i_1} \times \cdots \times D_{i_k}$ for which the function w_U is nonzero.

Definition 2.4.2 (Constraint (primal) graph)**.** *The* constraint (or primal) graph of I *denoted* $G(I)$ *is defined as* $G(I) = (V, E)$ *where* $E = \{\{u, v\} \mid \exists C_U \in \mathcal{S} \cup \mathcal{H}$ *s.t.* $\{u, v\} \subseteq U\}$*.*

Definition 2.4.3 (Treewidth of CSP)**.** *The* treewidth of a CSP instance I $\mathrm{tw}(I)$ *is defined as the treewidth of its constraint graph* $\mathrm{tw}(G(I))$.

In *binary CSP*, every hard and soft constraint is a unary or binary relation, and in *boolean CSP*, the domain D_v of every variable $v \in V$ is $D_v = \{0, 1\}$.

In Chapter 5 we will use Freuder's algorithm for solving CSPs of small treewidth:

Theorem 2.4.4 (Freuder [43])**.** *For a CSP instance I of treewidth τ and maximum domain size D, a minimum weight solution can be found in time $\mathcal{O}(D^\tau n + \|\mathcal{H}\| + \|\mathcal{S}\|)$.*

2.5 Monadic Second Order Logic

The monadic second order logic MSO extends first order logic using so called monadic variables, which are variables for sets of vertices in MSO_1 and in addition variables for sets of edges in MSO_2. To simplify the presentation, without loss of generality (cf. [98, Subsection 3.1.3]) we can assume that the input formulae are given in a variant of MSO_1 that uses only set variables (and no element variables).

Thus, atomic formulas are c_i for every constant symbol c_i, $P_i(X_1, \ldots, X_{r_i})$ for every r_i-ary relational symbol P_i and variables $X_1, \ldots, X_{r_i}$, and $X \subseteq Y$ for two variables X, Y. MSO formulas are built from atomic formulas using usual Boolean connectives $(\neg, \wedge, \vee, \rightarrow, \leftrightarrow)$ and quantification over variables $(\forall X, \exists X)$. We also use the usual shortcuts such as $X = Y$ for $X \subseteq Y \wedge Y \subseteq X$ etc. A (set) variable X is *free* in φ if it does not appear in any quantification in φ. If $\vec{X}$ is the tuple of all free variables in φ, we write $\varphi(\vec{X})$. A variable X is *bound* in φ if it is not free. By $\mathrm{qr}(\varphi)$ we denote the *quantifier rank* of φ which is the number of quantifiers of φ when transformed into the prenex form (i.e., all quantifiers are at the beginning of the formula). Let G be a graph, $S \subseteq V(G)$, and $\varphi(X)$ be an MSO formula with one free variable. We write

$$G \models \varphi(S)$$

to indicate that φ for G if X is interpreted as S; similarly for multiple free variables.

In our proofs we will restrict our attention to MSO_1 for two reasons which we only sketch now; see a detailed argument in Subsection 6.1.2. First, on graphs of bounded treewidth, we can equivalently study MSO_1 over the vocabulary σ_2 instead of MSO_2 over the standard graph vocabulary σ_1. Second, on graphs of bounded neighborhood diversity (and thus also cliquewidth), deciding MSO_2 is not even in XP.

MSO-definable sets. The following definition allows us to phrase our results as results about the expressivity and optimization over MSO-definable sets.

Definition 2.5.1 (MSO-definable sets)**.** *For a graph G on $|V(G)| = n$ vertices, we interpret a 0/1 vector $\mathbf{x} \in \{0,1\}^{nm}$ as m sets $X_1, \ldots, X_m \subseteq V$ where $v \in X_i$ iff $x_v^i = 1$. We then say that* $\mathbf{x}$ satisfies a formula φ *with m free variables if $G \models \varphi(X_1, \ldots, X_m)$. Let*

$$S_\varphi(G) = \{\mathbf{x} \mid \mathbf{x} \text{ satisfies } \varphi \text{ in } G\}.$$

When $m = 1$, we omit the superscript i and call the coordinates of $\mathbf{x}$ simply x_v for $v \in V(G)$.

2.6 Polytopes, Extended Formulations and Extension Complexity

For background on polytopes we refer the reader to Grünbaum [56] and Ziegler [120].

We write vectors of numbers in boldface, e.g., $\mathbf{x}, \mathbf{y}$ etc., and thus distinguish them from their entries, written in normal font. The i-th entry of $\mathbf{x}$ is denoted by x_i, $x(i)$ or $x[i]$. For a vector $\mathbf{x}$ and a subset I of coordinates, we denote by $\mathbf{x}|_I$ the *projection of* $\mathbf{x}$ *to* I, which is the subvector of $\mathbf{x}$ specified by the coordinates I. We write tuples of other objects (e.g. vertices, vertex sets etc.) as $\vec{p} = (p_1, \ldots, p_k)$. By $\operatorname{supp}(\mathbf{x}) = \{i \mid x_i \neq 0\}$ we denote the *support of* $\mathbf{x}$, that is, the set of its nonzero coordinates. By $\mathbf{A}_j$ we denote the j-th row of a matrix $\mathbf{A}$. Let $x \in \mathbb{Z}$; by $\langle x \rangle = \log_2 x + 1$ we denote the *encoding length* of x, and extend this notion naturally to vectors (for $\mathbf{x} \in \mathbb{Z}^n$, $\langle \mathbf{x} \rangle = \sum_{i=1}^n \langle x_i \rangle$), matrices (for $\mathbf{A} \in \mathbb{Z}^{n \times m}$, $\langle \mathbf{A} \rangle = \sum_{j=1}^m \langle \mathbf{A}_j \rangle$) and functions with finite domains (for $f : D \to \mathbb{R}$, $\langle f \rangle = \sum_{i \in D} \langle f(i) \rangle$). For simplification, we sometimes write the dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$ incorrectly not as $\mathbf{x}\mathbf{y}^\intercal$, but as $\mathbf{x}\mathbf{y}$; the expression $\mathbf{x}\mathbf{y}$ is always a dot product.

Definition 2.6.1 (Hyperplane)**.** *A* hyperplane *in $\mathbb{R}^n$ is a closed convex set of the form $\{\mathbf{x} \mid \mathbf{a}^\intercal \mathbf{x} = b\}$ where $\mathbf{a} \in \mathbb{R}^n, b \in \mathbb{R}$.*

Definition 2.6.2 (Halfspace)**.** *A* halfspace *in $\mathbb{R}^n$ is a closed convex set of the form $\{\mathbf{x} \mid \mathbf{a}^\intercal \mathbf{x} \leq b\}$ where $\mathbf{a} \in \mathbb{R}^n, b \in \mathbb{R}$. The inequality $\mathbf{a}^\intercal \mathbf{x} \leq b$ is said to define the corresponding halfspace.*

Definition 2.6.3 (Convex hull)**.** *A* convex hull *of a set $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subseteq \mathbb{R}^d$ is the set* $\operatorname{conv}(V) = \{\lambda_1 \mathbf{v}_1 + \cdots + \lambda_n \mathbf{v}_n \mid \sum_{i=1}^n \lambda_i = 1;\ \lambda_i \geq 0 \text{ for } i \in [n]\}$.

Definition 2.6.4 (Polytope)**.** *A* polytope *$P \subseteq \mathbb{R}^n$ is a bounded subset defined by intersection of finite number of halfspaces. A result of Minkowsky-Weyl states that equivalently, every polytope is the convex hull of a finite number of points.*

Definition 2.6.5 (Valid inequality)**.** *Let h be a halfspace defined by an inequality $\mathbf{a}^\intercal \mathbf{x} \leq b$; the inequality is said to be* valid *for a polytope P if $P = P \cap h$.*

Definition 2.6.6 (Face)**.** *Let $\mathbf{a}^\intercal \mathbf{x} \leq b$ be a valid inequality for polytope P; then, $P \cap \{\mathbf{x} \mid \mathbf{a}^\intercal \mathbf{x} = b\}$ is said to be a* face *of P.*

Note that, taking $\mathbf{a}$ to be the zero vector and $b = 0$ results in the face being P itself. Also, taking $\mathbf{a}$ to be the zero vector and $b = 1$ results in the empty set. These two faces are often called the trivial faces and they are polytopes "living in" dimensions n and -1, respectively. Every face – that is not trivial – is itself a polytope of dimension d where $0 \leq d \leq n - 1$.

It is not uncommon to refer to three separate (but related) objects as a face: the actual face as defined above, the valid inequality defining it, and the equation corresponding to the valid inequality. While this is clearly a misuse of notation, the context usually makes it clear as to exactly which object is being referred to.

Definition 2.6.7 (Vertices and Facets)**.** *The zero dimensional faces of a polytope are called its* vertices*, and the* $(n-1)$*-dimensional faces are called its* facets*.*

We denote by $\text{vert}(P)$ the set of vertices of a polytope P. By Definition 2.6.4, we have that $P = \text{conv}(\text{vert}(P))$.

Definition 2.6.8 (Extended formulation)**.** *Let P be a polytope in $\mathbb{R}^d$. A polytope Q in $\mathbb{R}^{d+r}$ is called an* extended formulation *or an* extension *of P if P is a projection of Q onto the first d coordinates. Note that for any linear map $\pi : \mathbb{R}^{d+r} \to \mathbb{R}^d$ such that $P = \pi(Q)$, a polytope Q' exists such that P is obtained by dropping all but the first d coordinates on Q', and, moreover, Q and Q' have the same number of facets.*

Definition 2.6.9 (Size of a polytope)**.** *The* size *of a polytope is defined to be the number of its facet-defining inequalities.*

Definition 2.6.10 (Extension complexity)**.** *Finally, the* extension complexity *of a polytope P, denoted by $\text{xc}(P)$, is the size of its smallest extended formulation.*

We refer the readers to the surveys [22, 61, 115, 116] for details and background of the subject and we only state two basic propositions about extended formulations here.

Proposition 2.6.11. *Let P be a polytope with a vertex set $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. Then $\text{xc}(P) \leq n$.*

Proof. Let $P = \text{conv}(\{\mathbf{v}_1, \ldots, \mathbf{v}_n\})$ be a polytope. Then, P is the projection of

$$Q = \left\{ (\mathbf{x}, \boldsymbol{\lambda}) \,\middle|\, \mathbf{x} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i;\ \sum_{i=1}^{n} \lambda_i = 1; \lambda_i \geq 0 \text{ for } i \in [n] \right\}.$$

It is clear that Q has at most n facets and therefore $\text{xc}(P) \leq n$. □

Proposition 2.6.12. *Let P be a polytope obtained by intersecting a set H of hyperplanes with a polytope Q. Then $\text{xc}(P) \leq \text{xc}(Q)$.*

Proof. Note that any extended formulation of Q, when intersected with H, gives an extended formulation of P. Intersecting a polytope with hyperplanes does not increase the number of facet-defining inequalities (and only possibly reduces it). □

3. Extension Complexity of the CSP Polytope

Before stating the main result of this chapter, we additionally introduce the notion of *extended feasible assignments*.

Extended feasible assignments. Most natural graph problems are modeled as maximization or minimization CSP. In order to express the objective function as a linear function, it is useful to additionally consider binary variables for soft constraints which indicate whether a given constraint is satisfied or not. Formally, for a given feasible assignment $\mathbf{z}^\star$ we define an *extended feasible assignment* $\mathrm{ex}(\mathbf{z}^\star) = (\mathbf{z}^\star, \mathbf{h}^\star) \in \mathbb{R}^{n+|\mathcal{S}|}$ as follows: the coordinates of $\mathbf{h}^\star$ are indexed by the soft constraints from $\mathcal{S}$ (or, more precisely, their scopes) and for each $C_U \in \mathcal{S}$, we have $h^\star_U = 1$ if and only if $\mathbf{z}^\star|_U \in C_U$, and $h^\star_U = 0$ otherwise. We denote by $\mathrm{Feas}^{ex}(I) = \{\mathrm{ex}(\mathbf{z}^\star) \mid \mathbf{z}^\star \in \mathrm{Feas}(I)\}$ the set of all extended feasible assignments for I.

CSP polytopes. For every instance I we define two polytopes: $CSP^{ex}(I)$ is the convex hull of $\mathrm{Feas}^{ex}(I)$ and $CSP(I)$ is the convex hull of $\mathrm{Feas}(I)$. We also define three trivial linear projections:

- $\mathrm{proj}_V(\mathbf{z}, \mathbf{h}) = \mathbf{z}, \qquad \mathrm{proj}_E(\mathbf{z}, \mathbf{h}) = \mathbf{h}, \qquad \mathrm{proj}_{id}(\mathbf{z}, \mathbf{h}) = (\mathbf{z}, \mathbf{h})$

where $\mathbf{z} \in \mathbb{R}^n$ and $\mathbf{h} \in \mathbb{R}^{|\mathcal{S}|}$, and observe that $\mathrm{proj}_V(CSP^{ex}(I)) = CSP(I)$.

Our main result is then summarized as the following theorem.

Theorem 3.0.1. *For every CSP instance $I = (V, \mathcal{D}, \mathcal{H}, \mathcal{S})$, there exists an extended formulation $P(I)$ of $CSP(I)$ and $CSP^{ex}(I)$ of size $\mathcal{O}(D^\tau n)$ where τ is the treewidth of I; moreover, given a tree decomposition of the constraint graph of I of width τ, the corresponding LP can be constructed in time $\mathcal{O}(D^\tau n)$.*

As a corollary, in Section 3.2 we obtain upper bounds on the extension complexity for several NP-hard problems on the class of graphs with bounded treewidth; as far as we know, these results have not been known.

3.1 Integer Linear Programming Formulation

We start by introducing the terms and notation that we use throughout this section. We assume that $I = (V, \mathcal{D}, \mathcal{H}, \mathcal{S})$ is a given instance of CSP. For the sake of simplicity of the presentation we do not consider the problem in the general weighted case here, although our techniques apply in the general setting as well. Let λ be a symbol not appearing in any of the domains D_u, $u \in V$. For every subset $W \subseteq V$ we define the set of all *configurations of* W as

$$\mathcal{K}(W) = \Big\{(\alpha_1, \ldots, \alpha_n) \mid \forall C_U \in \mathcal{H} : (U \subseteq W \implies \boldsymbol{\alpha}|_U \in C_U), \\ \text{and } \forall i \notin W : \alpha_i = \lambda\Big\}$$

Let $\mathbf{k} \in \mathcal{K}(U)$ be a configuration and $v \in V$. Recall that since $\mathbf{k}$ is a vector, $k(v)$ refers to the v-th element of $\mathbf{k}$. For $v \in V \setminus U$ and $\alpha \in D_v$, we use the notation

$\mathbf{k}[v \leftarrow \alpha]$ to denote the vector $\mathbf{k}'$ such that $k'(v) = \alpha$ and $k'(u) = k(u)$ for every $u \neq v$. Note that $\mathbf{k}[v \leftarrow \alpha]$ does not necessarily have to be a configuration.

For an n-dimensional vector $\mathbf{k} = (\alpha_1, \ldots, \alpha_n)$ and a subset of variables $U \subseteq V$ we denote by $\mathbf{k}\restriction_U$ the *restriction of* $\mathbf{k}$ *to* U that is defined as an n-dimensional vector with $\mathbf{k}\restriction_U(i) = k(i)$ for $i \in U$ and $\mathbf{k}\restriction_U(i) = \lambda$ for $i \notin U$ (i.e., we set to λ all coordinates of $\mathbf{k}$ outside of U).

In our linear program, for every index $v \in V$ and every $i \in D_v$, we introduce a binary variable y_v^i. The task of the variable y_v^i is to encode the value of the CSP-variable z_v: the variable y_v^i is set to one if and only if $z_v = i$. Since in every solution each variable assumes a unique value, we enforce the constraint $\sum_{i \in D(v)} y_v^i = 1$ for each $v \in V$.

For every configuration $\mathbf{k} \in \bigcup_{U : C_U \in \mathcal{S} \cup \mathcal{H}} \mathcal{K}(U)$ we introduce a binary variable $g(\mathbf{k})$. The intended meaning of the variable $g(\mathbf{k})$, for $\mathbf{k} \in \mathcal{K}(U)$ and $U \subseteq V$, is to provide information about the values of the CSP-variables z_u for $u \in U$ in the following way: $g(\mathbf{k}) = 1$ if and only if for every $u \in U$, $z_u = k(u)$. To ensure consistency between the y and g variables, for every $C_U \in \mathcal{S} \cup \mathcal{H}$ and for every $v \in U$, we enforce the constraint $\sum_{\mathbf{k} \in \mathcal{K}(U) : k(v) = i} g(\mathbf{k}) = y_v^i$. Note that for binary CSP, the $\mathbf{g}$ variables capture the values of CSP-variables $\mathbf{z}$ for pairs of elements from V that correspond to edges of the constraint graph.

Relaxing the integrality constraints we obtain the following initial LP relaxation of the CSP problem $I = (V, \mathcal{D}, \mathcal{H}, \mathcal{S})$:

$$\sum_{i \in D(v)} y_v^i = 1 \qquad \forall v \in V \tag{3.1}$$

$$\sum_{\mathbf{k} \in \mathcal{K}(U) : k(v) = i} g(\mathbf{k}) = y_v^i \qquad \forall C_U \in \mathcal{S} \cup \mathcal{H}\ \forall v \in U\ \forall i \in D(v) \tag{3.2}$$

$$\mathbf{0} \leq \mathbf{y}, \mathbf{g} \leq \mathbf{1}\ . \tag{3.3}$$

Note that there is a one to one correspondence between the (extended) feasible assignments of I and integral solutions of (3.1)–(3.3); from now on we denote by proj_1 the linear projection of the convex hull of integral solutions of (3.1)–(3.3) to $CSP^{ex}(I)$. Also observe that the total weight of CSP-constraints satisfied by an integral vector $(\mathbf{y}, \mathbf{g})$ satisfying (3.1)–(3.3) is[1]

$$\sum_{C_U \in \mathcal{S}} w(C_U) \sum_{\mathbf{k} \in \mathcal{K}(U) : \mathbf{k}|_U \in C_U} g(\mathbf{k})\ .$$

Unfortunately, even for CSP problems whose constraint graph is series-parallel, the polytope given by the LP (3.1)–(3.3) is not integral (consider, e.g., the instance of CSP corresponding to the independent set problem on K_3). The weakness of the formulation is that no *global* consistency among the $\mathbf{y}$ variables is guaranteed. To strengthen the relaxation, we introduce new variables and constraints derived from a tree decomposition of the constraint graph of I.

3.1.1 Extended Formulation

Here we describe, for every CSP instance $I = (V, \mathcal{D}, \mathcal{H}, \mathcal{S})$, a polytope $P(I)$, and in the next subsection we prove that $P(I)$ is an extended formulation of

[1] In the case of general payoff functions, the total weight is given by $\sum_{w_U \in \mathcal{S}} \sum_{\mathbf{k} \in \mathcal{K}(U) : w_U(\mathbf{k}|_U) \neq 0} w_U(\mathbf{k}|_U) g(\mathbf{k})$

$CSP^{ex}(I)$ and $CSP(I)$. The set of variables in the given LP description of $P(I)$ is substantially different from the set of variables used in the LP (3.1)–(3.3), and the set of new constraints is completely different from the the set of constraints in the LP (3.1)–(3.3). Whereas in the previous subsection, there is (roughly) a variable $g(\mathbf{k})$ for every feasible assignment of every subset of CSP variables corresponding to a *soft or hard constraint*, here we have a variable for every feasible assignment of every subset of CSP variables corresponding to a *bag in a given tree decomposition of the constraint graph*. Nevertheless, as we show after defining $P(I)$, there exists a simple linear projection of $P(I)$ to the convex hull of all integral points in the polytope given by the LP (3.1)–(3.3).

Let (T, B) be a fixed nice tree decomposition. Let $\mathcal{K}_B = \bigcup_{a \in V(T)} \mathcal{K}(B(a))$ be the set of all configurations of all bags in T. We use $V_I \subseteq V(T)$ to denote the subset of all introduce nodes in T and $V_F \subseteq V(T)$ to denote the subset of all forget nodes in T.

For every configuration $\mathbf{k} \in \mathcal{K}_B$ we introduce a binary variable $f(\mathbf{k})$. As in the previous subsection, the intended meaning of the variable $f(\mathbf{k})$ for $a \in V(T)$ and $\mathbf{k} \in \mathcal{K}(B(a))$, is to provide information about the values of the CSP-variables z_u for $u \in B(a)$ in the following way: $f(\mathbf{k}) = 1$ if and only if for every $u \in B(a)$, $z_u = k(u)$. To ensure consistency among variables indexed by the configurations of the same bag, namely to ensure that for every $a \in V(T)$ there exists exactly one configuration $\mathbf{k} \in \mathcal{K}(B(a))$ with $f(\mathbf{k}) = 1$, we introduce for every $a \in V(T)$ the LP constraint $\sum_{\mathbf{k} \in \mathcal{K}(B(a))} f(\mathbf{k}) = 1$.

For every introduce node $c \in V(T)$ with a child $b \in V(T)$ and for every configuration $\mathbf{k} \in \mathcal{K}(B(b))$ we have the constraint $\sum_{\mathbf{k}' \in \mathcal{K}(B(c)) : \mathbf{k}' \restriction_{B(b)} = \mathbf{k}} f(\mathbf{k}') = f(\mathbf{k})$, and symmetrically, for every forget node $c \in V(T)$ with a child $b \in V(T)$ and for every configuration $\mathbf{k} \in \mathcal{K}(B(c))$ we have the constraint $\sum_{\mathbf{k}' \in \mathcal{K}(B(b)) : \mathbf{k}' \restriction_{B(c)} = \mathbf{k}} f(\mathbf{k}') = f(\mathbf{k})$.

We remark that the idea of ensuring *local consistency* over the bags of a tree decomposition is indeed natural, but also profound. It is one of the crucial tools in heuristics for general CSPs, and also the key idea in Freuder's algorithm (Theorem 2.4.4).

Relaxing the integrality constraints and putting all these additional constraints together, we obtain:

$$\sum_{\mathbf{k} \in \mathcal{K}(B(a))} f(\mathbf{k}) = 1 \qquad \forall a \in V(T) \tag{3.4}$$

$$\sum_{\mathbf{k}' \in \mathcal{K}(B(c)) : \mathbf{k}' \restriction_{B(b)} = \mathbf{k}} f(\mathbf{k}') = f(\mathbf{k}) \qquad \forall c \in V_I, \forall \mathbf{k} \in \mathcal{K}(B(b)) \text{ where } b \text{ is the only child of } c \tag{3.5}$$

$$\sum_{\mathbf{k}' \in \mathcal{K}(B(b)) : \mathbf{k}' \restriction_{B(c)} = \mathbf{k}} f(\mathbf{k}') = f(\mathbf{k}) \qquad \forall c \in V_F, \forall \mathbf{k} \in \mathcal{K}(B(c)) \text{ where } b \text{ is the only child of } c \tag{3.6}$$

$$\mathbf{0} \leq \mathbf{f} \leq \mathbf{1} \ . \tag{3.7}$$

For the given binary CSP instance I, we denote the polytope associated with the LP (3.4)–(3.7), as $P(I)$. Note that it is not necessary to add any constraints for a join node $c = \Lambda(a, b)$: since $B(a) = B(b) = B(c)$, the variables $f(\mathbf{k})$ for all $\mathbf{k} \in \mathcal{K}(B(c))$ are "shared" by a and b.

Consider now a vector $\mathbf{f} \in P(I)$ and the following set of linear equations:

$$y_v^i = \sum_{\mathbf{k} \in \mathcal{K}(B(a)) : k(v) = i} f(\mathbf{k}) \qquad \forall a \in V(T), \forall v \in B(a), \forall i \in D_v \tag{3.8}$$

$$g(\mathbf{k}) = \sum_{\mathbf{k}' \in \mathcal{K}(B(a)) : \mathbf{k}' \restriction_U = \mathbf{k}} f(\mathbf{k}') \quad \forall a \in V(T), \forall C_U \in \mathcal{S} \cup \mathcal{H} \text{ s.t. } U \subseteq B(a), \forall \mathbf{k} \in \mathcal{K}(U) \ . \tag{3.9}$$

It is just a technical exercise to check that for a given $\mathbf{f} \in P(I)$, there always exists a unique solution $(\mathbf{y}, \mathbf{g})$ of this LP and that the unique $(\mathbf{y}, \mathbf{g})$ is a linear projection of $\mathbf{f}$. Moreover, such a vector $(\mathbf{y}, \mathbf{g})$ also satisfies the LP constraints (3.1)–(3.3). The point is that there exists a linear projection, obtained from (3.8)–(3.9), of $P(I)$ into the polytope defined by the LP (3.1)–(3.3); moreover, an integral point from $P(I)$ is mapped on an integral point. From now on we denote this projection proj_2.

3.1.2 Proof of Theorem 3.0.1

As in the previous subsections, we assume that $I = (V, \mathcal{D}, \mathcal{H}, \mathcal{S})$ is a given instance of CSP, $G = (V, E)$ is the constraint graph of I and (T, B) is a fixed nice tree decomposition of G. We start by introducing several notions that will help us dealing with tree decompositions and our linear program.

Recall that for a node $a \in V(T)$, T_a is the subtree of T rooted in a. The *configurations relevant to* T_a are those in the set $\mathcal{R}(a) = \bigcup_{b \in V(T_a)} \mathcal{K}(B(b))$, and the *variables relevant to* T_a are those $f(\mathbf{k})$ for which $\mathbf{k} \in \mathcal{R}(a)$. For succinctness of notation, we denote the projection $\mathbf{f}|_{\mathcal{R}(a)}$ of the vector $\mathbf{f}$ on the set of variables relevant to T_a also by $\mathbf{f}|_a$. The *constraints relevant to* T_a are those containing only the variables relevant to T_a. We say that a vector $\mathbf{p} \in \{0,1\}^{\mathcal{R}(a)}$ *agrees with the configuration* $\mathbf{k} \in \mathcal{R}(a)$ if $p(\mathbf{k}) = 1$.

Let $\mathbf{f}$ be a fixed solution of the LP (3.4)–(3.7) that corresponds to a vertex of the polytope $P(I)$. Our main tool is the following lemma; our approach was partially inspired by the proof of the matching polytope theorem as given by Schrijver [107].

Lemma 3.1.1. *For every node $b \in V(T)$, there exist a positive integer M and binary vectors $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M \in \{0,1\}^{\mathcal{R}(b)}$, some possibly identical, such that*

- ♠ *every $\mathbf{p}_i$ satisfies the constraints relevant to T_b,*
- ♣ $\mathbf{f}|_b = \frac{1}{M} \sum_{i=1}^{M} \mathbf{p}_i$.

Proof. By induction. We start in the leaves of T and proceed in a bottom-up fashion.

Base case. Assume that $b \in V(T)$ is a leaf of the nice decomposition tree T. By the definition of a nice tree decomposition, the bag $B(b)$ is empty and thus $\mathcal{R}(b)$ is empty. This makes the lemma trivially satisfied by the empty vector.

Inductive step. Consider an internal node $c \in V(T)$ of the nice decomposition tree T. We distinguish three cases: c is a join node, c is an introduce node and c is a forget node.

Join node. Assume $c = \Lambda(a, b)$. Recall that $B(a) = B(b) = B(c)$. By the inductive assumption, there exist integers M and M' and integral vectors $\mathbf{p}_1, \ldots, \mathbf{p}_M \in \{0,1\}^{\mathcal{R}(a)}$, each of them satisfying the relevant constraints for T_a and such that $\mathbf{f}|_a = \frac{1}{M}\sum_{i=1}^{M} \mathbf{p}_i$, and integral vectors $\mathbf{q}_1, \ldots, \mathbf{q}_{M'} \in \{0,1\}^{\mathcal{R}(b)}$, each of them satisfying the relevant constraints for T_b and such that $\mathbf{f}|_b = \frac{1}{M'}\sum_{i=1}^{M'} \mathbf{q}_i$.

Two vectors $\mathbf{p}_i$ and $\mathbf{q}_j$ that agree with a given configuration $\mathbf{k} \in \mathcal{K}(B(c))$ can be easily merged into an integral vector $\mathbf{r} \in \{0,1\}^{\mathcal{R}(c)}$ that satisfies $\mathbf{r}|_a = \mathbf{p}_i$ and $\mathbf{r}|_b = \mathbf{q}_j$; as the set of all constraints relevant to T_c is the union of the constraints relevant to T_a and the constraints relevant to T_b, the vector $\mathbf{r}$ satisfies also all the constraints relevant to T_c.

For simplicity we assume, without loss of generality, that $M = M'$. Then, by the property ♣ and since $B(a) = B(b) = B(c)$, for every configuration $\mathbf{k} \in \mathcal{K}(B(c))$, the number of vectors $\mathbf{p}_i$ that agree with $\mathbf{k}$ is equal to the number of vectors $\mathbf{q}_j$ that agree with $\mathbf{k}$, namely $M \cdot f(\mathbf{k})$. Thus, it is possible to match the vectors $\mathbf{p}_i$ and $\mathbf{q}_j$ one to one in such a way that both vectors in each pair agree with the same configuration; let $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_M$ denote the result of their merging as described above. Then the vectors $\mathbf{r}_i$ satisfy the property ♠ as explained in the previous paragraph, and by construction they also satisfy the property ♣.

Introduce node. Assume that $c = b * (v)$ and thus $B(c) = B(b) \cup \{v\}$. By the inductive assumption, there exists integer M and integral vectors $\mathbf{p}_1, \ldots, \mathbf{p}_M \in \{0,1\}^{\mathcal{R}(b)}$, each of them satisfying the relevant constraints for T_b and such that $\mathbf{f}|_b = \frac{1}{M}\sum_{i=1}^{M} \mathbf{p}_i$. Without loss of generality we assume that for every variable relevant to T_c, its M-multiple is integral. We partition the vectors $\mathbf{p}_1, \ldots, \mathbf{p}_M$ into several groups indexed by the configurations from $\mathcal{K}(B(b))$: the group $Z_{\mathbf{k}}$, for $\mathbf{k} \in \mathcal{K}(B(b))$, consists exactly of those vectors $\mathbf{p}_i$ that agree with $\mathbf{k}$.

Consider a fixed configuration $\mathbf{k} \in \mathcal{K}(B(b))$ and the corresponding group $Z_{\mathbf{k}}$. Note that the size of this group is $M \cdot f(\mathbf{k})$. We further partition the group $Z_{\mathbf{k}}$ into at most $|D_v|$ subgroups $Z_{\mathbf{k}'}$, where $\mathbf{k}' = \mathbf{k}[v \leftarrow j]$, for every $j \in D_v$ satisfying $\mathbf{k}[v \leftarrow j] \in \mathcal{K}(B(c))$, in such a way that $Z_{\mathbf{k}'}$ contains exactly $M \cdot f(\mathbf{k}')$ vectors (it does not matter which ones); the LP constraint (3.5) makes this possible. Then, for every $j \in D_v$, we create from every vector $\mathbf{p} \in Z_{\mathbf{k}[v\leftarrow j]}$ a new integral vector $\mathbf{q}_{\mathbf{p}}$ in the following way:

- for every $\bar{\mathbf{k}} \in \mathcal{R}(b)$, $q_{\mathbf{p}}(\bar{\mathbf{k}}) = p(\bar{\mathbf{k}})$; this guarantees $\mathbf{q}_{\mathbf{p}}|_b = \mathbf{p}$,
- $q_{\mathbf{p}}(\mathbf{k}[v \leftarrow j]) = 1$,
- for every $i \in D_v$, $i \neq j$, $q_{\mathbf{p}}(\mathbf{k}[v \leftarrow i]) = 0$.

Obviously, the new vectors $\mathbf{q}_{\mathbf{p}}$ satisfy all constraints relevant to T_b, and it is easy to check that they satisfy all constraints relevant to T_c as well, given the definitions above. Moreover, the definitions above imply that the vectors $\mathbf{q}_{\mathbf{p}}$ satisfy the property ♣.

Forget node. Assume that $c = b \dagger (v)$ and thus $B(c) = B(b) \setminus \{v\}$. This case is symmetric to the previous one in that instead of splitting the groups $Z_{\mathbf{k}}$ into smaller groups $Z_{\mathbf{k}'}$, we merge them into bigger $Z_{\mathbf{k}'}$.

By the inductive assumption, there exists an integer M and integral vectors $\mathbf{p}_1, \ldots, \mathbf{p}_M \in \{0,1\}^{\mathcal{R}(b)}$, each of them satisfying the relevant constraints for T_b and such that $\mathbf{f}|_b = \frac{1}{M}\sum_{i=1}^{M} \mathbf{p}_i$. Without loss of generality we assume that for every variable relevant to T_c, its M-multiple is integral. We partition the vectors

$\mathbf{p}_1, \ldots, \mathbf{p}_M$ into several groups indexed by the configurations from $\mathcal{K}(B(b))$: the group $Z_{\mathbf{k}}$, for $\mathbf{k} \in \mathcal{K}(B(b))$, consists exactly of those vectors $\mathbf{p}_i$ that agree with $\mathbf{k}$. Note that the size of $Z_{\mathbf{k}}$ is $M \cdot f(\mathbf{k})$.

For every $\mathbf{k}' \in \mathcal{K}(B(c))$ we create a bigger group $Z_{\mathbf{k}'}$ by merging $|D_v|$ of the groups $Z_{\mathbf{k}}$, namely those satisfying $\mathbf{k}|_{B(c)} = \mathbf{k}'$. By the LP constraint (3.6), the new group $Z_{\mathbf{k}'}$ contains exactly $M \cdot f(\mathbf{k}')$ vectors. For every $\mathbf{k}' \in \mathcal{K}(B(c))$, we create from every vector $\mathbf{p} \in Z_{\mathbf{k}'}$ a new integral vector $\mathbf{q_p}$ in the following way:

- for every $\bar{\mathbf{k}} \in \mathcal{R}(b)$, $q_{\mathbf{p}}(\bar{\mathbf{k}}) = p(\bar{\mathbf{k}})$.

If $\mathcal{K}(B(c)) \subseteq \mathcal{R}(b)$, there is nothing more to do. Otherwise we further define

- $q_{\mathbf{p}}(\mathbf{k}') = 1$, and for every $\hat{\mathbf{k}} \in \mathcal{K}(B(c))$, $\hat{\mathbf{k}} \neq \mathbf{k}'$, $q_{\mathbf{p}}(\hat{\mathbf{k}}) = 0$.

We have to check that the vectors $\mathbf{q_p}$ satisfy all constraints relevant to T_c. The only possibly new constraints are those using variables $f(\mathbf{k}')$ for $\mathbf{k}' \in \mathcal{K}(B(c))$ and it is easily seen that they are satisfied, given the definitions above. Also, the definitions above imply that the vectors $\mathbf{q_p}$ satisfy the property ♣. □

By applying Lemma 3.1.1 to the whole tree T, that is, to the subtree rooted in the root of T, we immediately obtain that $\mathbf{f}$ is an integral vector, and, thus, also the corresponding vertex of $P(I)$ is integral. As this holds for every vertex of $P(I)$, we conclude that $P(I)$ is an integral polytope.

Considering the notes at the ends of the previous two subsections, we also conclude that $CSP^{ex}(I) = \mathrm{proj}_1(\mathrm{proj}_2(P(I))$ and $CSP(I) = \mathrm{proj}_V(CSP^{ex}(I))$.

To complete the proof of Theorem 3.0.1, we observe that the number of variables and constraints in the LP (3.4)–(3.7) is $\mathcal{O}(D^\tau n)$.

3.2 Applications

The purpose of this section is to make explicit the extension complexity upper bounds given in Theorem 3.0.1 for several well known graph problems. Note that in all considered cases the constraint graph obtained from our CSP formulation is identical with the input graph; specifically, this means the treewidth of the CSP instance is the treewidth of the input graph. This is not obvious: for example, a natural CSP formulation of the Minimum Dominating Set problem involves constraints over a neighborhood of a vertex and thus has a possibly unbounded treewidth. (On the other hand, it *is* possible to provide a bounded treewidth CSP formulation for MDS. However, its constraint graph is not G but the incidence graph $I(G)$ of G, and the constraints are a little complicated, so we do not describe it here. Moreover, an existence of a compact extension for the Dominating Set polytope follows from the main result of the next chapter, Theorem 4.3.1.)

We find it interesting that the attained extension complexity upper bounds almost meet the best possible, assuming Strong ETH, *time complexity* lower bounds, given by Lokshtanov et al. [85]. They show for several problems whose time complexity is upper bounded by $\mathcal{O}(c^\tau n^{\mathcal{O}(1)})$, that they cannot be solved in time $\mathcal{O}((c-\epsilon)^\tau n^{\mathcal{O}(1)})$, for any $\epsilon > 0$, unless SETH fails. The only exception in our list is the Multiway Cut problem where the corresponding lower bounds are not known. To state our results, we use for each problem the following template:

PROBLEM NAME Projection Extension complexity Time complexity
INSTANCE: ...
SOLUTION: ...
CSP FORMULATION: V, $\mathcal{D}$, $\mathcal{H}$, $\mathcal{S}$. CSP version: Decision / MAX / MIN
where *Projection* is the name of the linear projection that yields the natural polytope of the problem I from the $CSP^{ex}(I)$ polytope (or from the $P(I)$ polytope, in case of the OCT problem).

COLORING / CHROMATIC NUMBER [5] proj_V $\mathcal{O}(q^\tau n)$ $\mathcal{O}(q^\tau n)$
INSTANCE: Graph $G = (V, E)$, set of colors $[q]$.
SOLUTION: A coloring of G with q colors with no monochromatic edges.
CSP FORMULATION: $V = [n]$, $D_v = [q]$ for every $v \in V$, $H_{uv} = \{(i, j) \mid i \in D_u, j \in D_v, i \neq j\}$ for every $uv \in E$, $\mathcal{S} = \emptyset$. Decision
Comment: Note that CHROMATIC NUMBER $\chi(G)$ of G is always upper bounded by $\tau + 1$ since graphs treewidth τ are τ-degenerate [66] and τ-degenerate graphs are $(\tau + 1)$-colorable [112]. Thus, if the goal is to determine $\chi(G)$, it suffice to find the smallest q such that $CSP^{ex}(I)$ is non-empty.

LIST-H-COLORING / LIST HOMOMORPHISM proj_V $\mathcal{O}(L^\tau n)$ $\mathcal{O}(L^\tau n)$ [35]
INSTANCE: Graph $G = (V, E)$, graph $H = (V_H, E_H)$ possibly containing loops, and for every vertex $v \in V$ a set $L(v) \subseteq V_H$.
SOLUTION: A mapping $f : V \to V_H$ such that $\forall uv \in E$ it holds that $f(u)f(v) \in E_H$ and $f(v) \in L(v)$ for every $v \in V$.
CSP FORMULATION: $V = [n]$, $D_v = L(v)$ for every $v \in V$, $H_{uv} = \{(i, j) \mid i \in D_u, j \in D_v, ij \in E_H\}$ for every $uv \in E$, $\mathcal{S} = \emptyset$. Decision
Comment: Note that the problems LIST COLORING, PRECOLORING EXTENSION and H-COLORING (or GRAPH HOMOMORPHISM) are all special cases of this problem. The lower bound given by Lokshtanov et al. [85] applies to all of them since COLORING is a special case of each of them.

UNIQUE GAMES [65] proj_{id} $\mathcal{O}(t^\tau n)$ —
INSTANCE: Graph $G = (V, E)$, an integer $t \in \mathbb{N}$, a permutation π_e of order t for every edge $e \in E$.
SOLUTION: A mapping $\ell : V \to [t]$ such that the number of edges $uv \in E$ with $\pi_{uv}(\ell(u)) = \ell(v)$ is maximized.
CSP FORMULATION: $V = [n]$, $D_v = [t]$ for every $v \in V$, $\mathcal{H} = \emptyset$, $C_{uv} = \{(i, \pi_{uv}(i)) \mid i \in D_u\}$ for every edge $uv \in E$. MAX
Comment: The decision variant of this problem is not interesting as it is trivially solvable in polynomial time.

MIN MULTIWAY CUT [5] proj_E $\mathcal{O}(t^\tau n)$ $\mathcal{O}(t^\tau n)$
INSTANCE: Graph $G = (V, E)$, an integer $t \in \mathbb{N}$ and t vertices $s_1, \ldots, s_t \in V$.
SOLUTION: A partition of V into sets $V_1, \ldots, V_t$ such that for every i we have $s_i \in V_i$ and the total number of edges between V_i and V_j for $i \neq j$ is minimized.
CSP FORMULATION: $V = [n]$, $D_v = [t]$ for every $v \in V$, $\mathcal{H} = \emptyset$, $C_{uv} = \{(i, i) \mid i \in [t]\}$ for every edge $uv \in E$. MAX

Comment: Setting $z_v = i$ models vertex v belonging to the set V_i. Not satisfying the constraint C_{uv} means that the edge uv belongs to the multiway cut.

MAX CUT [5] $\quad \text{proj}_E \quad \mathcal{O}(2^\tau n) \quad \mathcal{O}(2^\tau n)$

INSTANCE: Graph $G = (V, E)$.
SOLUTION: A partition of vertices into two sets V_1, V_2 such that the number of edges between V_1 and V_2 is maximized.
CSP FORMULATION: $V = [n]$, $D_v = \{0, 1\}$ for every $v \in V$, $\mathcal{H} = \emptyset$, $C_{uv} = \{(1, 0), (0, 1)\}$ for every edge $uv \in E$. MAX
Comment: The values $0, 1$ model the vertex belonging to the set V_1 or V_2. If we replace maximization by minimization, the problem becomes EDGE BIPARTIZATION (aka EDGE OCT) problem which is a parametric dual of MAX CUT.

MIN VERTEX COVER [5] $\quad \text{proj}_V \quad \mathcal{O}(2^\tau n) \quad \mathcal{O}(2^\tau n)$

INSTANCE: Graph $G = (V, E)$.
SOLUTION: A set of vertices $C \subseteq V$ of minimal size such that every edge contains a vertex $v \in C$ as at least one of its endpoints.
CSP FORMULATION: $V = [n]$, $D_v = \{0, 1\}$ for every $v \in V$, $H_{uv} = \{(1, 1), (0, 1), (1, 0)\}$ for every edge $uv \in E$, $C_v = \{0\}$ for every vertex $v \in V$. MIN
Comment: The values $1, 0$ model the vertex belonging to C or $V \setminus C$.

MAX INDEPENDENT SET [5] $\quad \text{proj}_V \quad \mathcal{O}(2^\tau n) \quad \mathcal{O}(2^\tau n)$

INSTANCE: Graph $G = (V, E)$.
SOLUTION: A set of vertices $C \subseteq V$ of maximal size such that no edge contains both its endpoints in C.
CSP FORMULATION: $V = [n]$, $D_v = \{0, 1\}$ for every $v \in V$, $H_{uv} = \{(0, 0), (0, 1), (1, 0)\}$ for every edge $uv \in E$, $C_v = \{1\}$ for every vertex $v \in V$. MAX
Comment: The values $1, 0$ model the vertex belonging to C or $V \setminus C$.

ODD CYCLE TRANSVERSAL [85] $\quad \text{proj}_{OCT} \circ \text{proj}_2 \quad \mathcal{O}(3^\tau n) \quad \mathcal{O}(3^\tau n)$

INSTANCE: Graph $G = (V, E)$.
SOLUTION: A subset of vertices $W \subseteq V$ of minimal size such that $G[V \setminus W]$ is a bipartite graph.
CSP FORMULATION: $V = [n]$, $D_v = \{0, 1, 2\}$ for every $v \in V$, $H_{uv} = \{0, 1, 2\}^2 \setminus \{(0, 0), (1, 1)\}$ for every edge $uv \in E$, $C_v = \{0, 1\}$ for every $v \in V$. MIN
Comment: The values $0, 1, 2$ model the vertex belonging to either the first or the second partite of a bipartite graph, or the deletion set W. Satisfying the constraint C_v corresponds to not putting v in the deletion set W. Also known as VERTEX BIPARTIZATION. The projection $\text{proj}_{OCT} : P(I) \to \{0, 1\}^V$ is defined as

$$\text{proj}_{OCT}(y_1^0, y_1^1, y_1^2, y_2^0, y_2^1, y_2^2, \ldots, y_n^0, y_n^1, y_n^2, \mathbf{g}) = (y_1^2, y_2^2, \ldots, y_n^2) \quad .$$

4. Extension Complexity of the MSO Polytope

Let us give an outline of the following chapter. In Section 4.1, we describe the foundations of Courcelle's theorem by introducing the notion of $[m]$-colored τ-boundaried graphs and their logical equivalences. In Section 4.2, we describe our geometric tool, the glued product of polytopes. In Section 4.3 we prove the existence of compact extended formulations for MSO polytopes parameterized by the length of the given MSO formula and the treewidth of the given graph. In Section 4.4 we describe how to efficiently construct such a polytope given a tree decomposition of a graph. Finally, in Section 4.5, we show applicability of our proof to graphs of bounded cliquewidth, and obtain an optimization version of Courcelle's theorem in a particularly simple way.

4.1 Preliminaries

4.1.1 $[m]$-colored τ-boundaried Graphs

For an integer $m \geq 0$, an *$[m]$-colored graph* is a pair $(G, \vec{V})$ where $G = (V, E)$ is a graph and $\vec{V} = (V_1, \ldots, V_m)$ is an m-tuple of subsets of vertices of G called an *m-coloring of G*. For integers $m \geq 0$ and $\tau \geq 0$, an *$[m]$-colored τ-boundaried graph* is a triple $(G, \vec{V}, \vec{p})$ where $(G, \vec{V})$ is an $[m]$-colored graph and $\vec{p} = (p_1, \ldots, p_\tau)$ is a τ-tuple of vertices of G called a *boundary* of G. If the tuples $\vec{V}$ and $\vec{p}$ are clear from the context or if their content is not important, we simply denote an $[m]$-colored τ-boundaried graph by $G^{[m],\tau}$. For a tuple $\vec{p} = (p_1, \ldots, p_\tau)$, we denote by p the corresponding set, that is, $p = \{p_1, \ldots, p_\tau\}$.

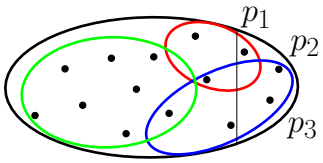


Figure 4.1: A [3]-colored 3-boundaried graph with $\vec{p} = (p_1, p_2, p_3)$.

Two $[m]$-colored τ-boundaried graphs $(G_1, \vec{V}, \vec{p})$ and $(G_2, \vec{U}, \vec{q})$ are *compatible* if the function $h : \vec{p} \to \vec{q}$, defined by $h(p_i) = q_i$ for each i, is an isomorphism of the induced subgraphs $G_1[\{p_1, \ldots, p_\tau\}]$ and $G_2[\{q_1, \ldots, q_\tau\}]$, and if for each i and j, $p_i \in V_j \Leftrightarrow q_i \in U_j$.

Given two compatible $[m]$-colored τ-boundaried graphs $G_1^{[m],\tau} = (G_1, \vec{U}, \vec{p})$ and $G_2^{[m],\tau} = (G_2, \vec{W}, \vec{q})$, the *join* of $G_1^{[m],\tau}$ and $G_2^{[m],\tau}$, denoted by $G_1^{[m],\tau} \oplus G_2^{[m],\tau}$, is the $[m]$-colored τ-boundaried graph $G^{[m],\tau} = (G, \vec{V}, \vec{p})$ where

- G is the graph obtained by taking the disjoint union of G_1 and G_2, and for each i, identifying the vertex p_i with the vertex q_i and keeping the label p_i for it;

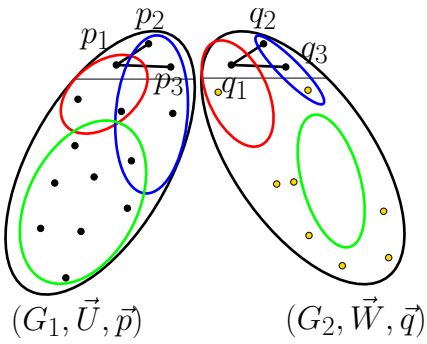


Figure 4.2: Compatibility of two [3]-colored 3-boundaried graphs.

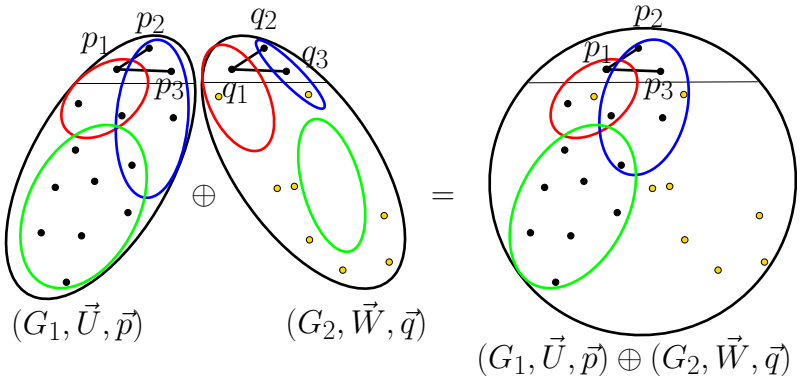


Figure 4.3: The *join* of two $[m]$-colored τ-boundaried graphs.

- $\vec{V} = (V_1, \ldots, V_m)$ with $V_j = U_j \cup W_j$ and every q_i replaced by p_i, for each j and i;
- $\vec{p} = (p_1, \ldots, p_\tau)$ with p_i being the node in $V(G)$ obtained by the identification of $p_i \in V(G_1)$ and $q_i \in V(G_2)$, for each i.

Because of the choice of referring to the boundary vertices by their names in $G_1^{[m],\tau}$, it does not always hold that $G_1^{[m],\tau} \oplus G_2^{[m],\tau} = G_2^{[m],\tau} \oplus G_1^{[m],\tau}$; however, the two structures are isomorphic and equivalent for our purposes (see below).

4.1.2 Monadic Second Order Logic and Types of Graphs

An $[m]$-colored τ-boundaried graph $G = (V, E)$ with boundary $p_1, \ldots, p_\tau$ colored with $V_1, \ldots, V_m$ is viewed as a structure $(V_I, \{p_1, \ldots, p_\tau\}, \{E_I, L_V, L_E, V_1, \ldots, V_m\})$; for notational simplicity, we stick to the notation $G^{[m],\tau}$ or $(G, \vec{V}, \vec{p})$. The corresponding vocabulary is denoted by $\sigma_{m,\tau}$. We denote by $\mathsf{MSO}[k, \tau, m]$ the set of all MSO_1 formulae φ over the vocabulary $\sigma_{\tau,m}$ with $\mathrm{qr}(\varphi) \leq k$.

Two $[m]$-colored τ-boundaried graphs $G_1^{[m],\tau}$ and $G_2^{[m],\tau}$ are MSO$[k]$-*elementarily equivalent* if they satisfy the same $\mathsf{MSO}[k, \tau, m]$ formulae; this is denoted by $G_1^{[m],\tau} \equiv_k^{MSO} G_2^{[m],\tau}$. The main tool in the model theoretic approach to Courcelle's theorem, that will also play a crucial role in our approach, can be stated as the following theorem which follows from [83, Proposition 7.5 and Theorem 7.7].

Theorem 4.1.1 ([83]). *For any fixed $\tau, k, m \in \mathbb{N}$, the equivalence relation $\equiv_k^{MSO}$ has a finite number of equivalence classes.*

Let us denote the equivalence classes of the relation $\equiv_k^{MSO}$ by $\mathcal{C} = \{\alpha_1 \ldots, \alpha_w\}$, fixing an ordering such that α_1 is the class containing the empty graph. Note that the size of $\mathcal{C}$ depends only on k, m and τ, that is, $|\mathcal{C}| = f(k, m, \tau)$ for some computable function f. For a given MSO formula φ with m free variables, we define an *indicator function* $\rho_\varphi : \{1, \ldots, |\mathcal{C}|\} \to \{0, 1\}$ as follows: for every i, if there exists a graph $G^{[m],\tau} \in \alpha_i$ such that $G^{[m],\tau} \models \varphi$, we set $\rho_\varphi(i) = 1$, and we set $\rho_\varphi(i) = 0$ otherwise; note that if there exists a graph $G^{[m],\tau} \in \alpha_i$ such that $G^{[m],\tau} \models \varphi$, then $G'^{[m],\tau} \models \varphi$ for every $G'^{[m],\tau} \in \alpha_i$.

For every $[m]$-colored τ-boundaried graph $G^{[m],\tau}$, its *type*, with respect to the relation $\equiv_k^{MSO}$, is the class to which $G^{[m],\tau}$ belongs. We say that *types* α_i and α_j are *compatible* if there exist two $[m]$-colored τ-boundaried graphs of types α_i and α_j that are compatible; note that this is well defined as all $[m]$-colored τ-boundaried graphs of a given type are compatible. For every $i \geq 1$, we will encode the type α_i naturally as a binary vector $\{0, 1\}^{|\mathcal{C}|}$ with exactly one 1, namely with 1 on the position i.

An important property of the types and the join operation is that the type of a join of two $[m]$-colored τ-boundaried graphs depends on their types only. The following lemma really only says this simple fact; we encourage the reader not be frightened by notation. We will see that the join operation corresponds to join nodes of a tree decomposition. Thus, the following lemma will be key in dealing with join nodes.

Lemma 4.1.2 (Join node lemma [83, Lemma 7.11] and [53, Lemma 3.5])**.** *Let $G_a^{[m],\tau}$, $G_{a'}^{[m],\tau}$, $G_b^{[m],\tau}$ and $G_{b'}^{[m],\tau}$ be $[m]$-colored τ-boundaried graphs such that $G_a^{[m],\tau} \equiv_k^{MSO} G_{a'}^{[m],\tau}$ and $G_b^{[m],\tau} \equiv_k^{MSO} G_{b'}^{[m],\tau}$. Then $(G_a^{[m],\tau} \oplus G_b^{[m],\tau}) \equiv_k^{MSO} (G_{a'}^{[m],\tau} \oplus G_{b'}^{[m],\tau})$.*

The importance of the lemma rests in the fact that for determination of the type of a join of two $[m]$-colored τ-boundaried graphs, it suffices to know only a *small* amount of information about the two graphs, namely their types. The following two lemmas deal in a similar way with the type of a graph in the remaining situations of an introduce and a forget node. Essentially, the lemmas say that the type of an $[m]$-colored τ-boundaried graph obtained by attaching a new vertex v to the boundary and coloring it is entirely determined by the type of the original graph, the way v is attached to the boundary, and the coloring of v. The lemma which follows after deals analogously with the situation when we omit a vertex from the boundary.

Lemma 4.1.3 (Introduce node lemma [53, implicit])**.** *Let $(G_a, \vec{X}, \vec{p})$, $(G_b, \vec{Y}, \vec{q})$ be $[m]$-colored τ-boundaried graphs and let $(G_{a'}, \vec{X'}, \vec{p'})$, $(G_{b'}, \vec{Y'}, \vec{q'})$ be $[m]$-colored $(\tau + 1)$-boundaried graphs with $G_a = (V, E)$, $G_{a'} = (V', E')$, $G_b = (W, F)$, $G_{b'} = (W', F')$ such that*

1. *$(G_a, \vec{X}, \vec{p}) \equiv_k^{MSO} (G_b, \vec{Y}, \vec{q})$;*
2. *$V' = V \cup \{v\}$ for some $v \notin V$, $N(v) \subseteq p$, $\vec{p}$ is a subtuple of $\vec{p'}$ and $(G_{a'}[V], \vec{X'}[V], \vec{p'}[V]) = (G_a, \vec{X}, \vec{p})$;*
3. *$W' = W \cup \{w\}$ for some $w \notin W$, $N(w) \subseteq q$, $\vec{q}$ is a subtuple of $\vec{q'}$ and $(G_{b'}[W], \vec{Y'}[W], \vec{q'}[W]) = (G_b, \vec{Y}, \vec{q})$;*

4. $(G_{a'}, \vec{X'}, \vec{p'})$ *and* $(G_{b'}, \vec{Y'}, \vec{q'})$ *are compatible.*

Then $(G_{a'}, \vec{X'}, \vec{p'}) \equiv_k^{MSO} (G_{b'}, \vec{Y'}, \vec{q'})$.

Lemma 4.1.4 (Forget node lemma [53, implicit])**.** *Let* $(G_a, \vec{X}, \vec{p})$, $(G_b, \vec{Y}, \vec{q})$ *be* $[m]$*-colored* τ*-boundaried graphs and let* $(G_{a'}, \vec{X'}, \vec{p'})$, $(G_{b'}, \vec{Y'}, \vec{q'})$ *be* $[m]$*-colored* $(\tau+1)$*-boundaried graphs with* $G_a = (V, E)$, $G_{a'} = (V', E')$, $G_b = (W, F)$, $G_{b'} = (W', F')$ *such that*

1. $(G_{a'}, \vec{X'}, \vec{p'}) \equiv_k^{MSO} (G_{b'}, \vec{Y'}, \vec{q'})$;
2. $V \subseteq V'$, $|V'| = |V| + 1$, $\vec{p}$ *is a subtuple of* $\vec{p'}$ *and* $(G_{a'}[V], \vec{X'}[V], \vec{p'}[V]) = (G_a, \vec{X}, \vec{p})$;
3. $W \subseteq W'$, $|W'| = |W|+1$, $\vec{q}$ *is a subtuple of* $\vec{q'}$ *and* $(G_{b'}[W], \vec{Y'}[W], \vec{q'}[W]) = (G_b, \vec{Y}, \vec{q})$.

Then $(G_a, \vec{X}, \vec{p}) \equiv_k^{MSO} (G_b, \vec{Y}, \vec{q})$.

These three lemmas are examples of Feferman-Vaught-style theorems, in the context of algorithmic metatheorems also known as the "composition method". Its applicability is large; cf. a survey of Makowsky [86].

4.1.3 Feasible Types

Suppose that we are given an MSO_1 formula φ over σ_2 with m free variables and of quantifier rank at most k, a graph G of treewidth at most τ represented as the σ_2 structure $I(G)$, and a nice tree decomposition (T, B) of the graph G.

For every node of T we are going to define certain types and tuples of types as *feasible*. Recall that the operator $\eta(X)$ for $X \subseteq V$ orders X according to a fixed ordering of V (Subsection 2.3.1). For a node $b \in V(T)$ of any kind (leaf, introduce, forget, join) and for $\alpha \in \mathcal{C}$, we say that α is a *feasible type of the node* b if there exist $X_1, \ldots, X_m \subseteq V(G_b)$ such that $(G_b, \vec{X}, \eta(B(b)))$ is of type α where $\vec{X} = (X_1, \ldots, X_m)$; we say that $\vec{X}$ *realizes type* α *on the node* b. We denote the set of feasible types of the node b by $\mathcal{F}(b)$.

For an *introduce* node $b \in V(T)$ with a child $a \in V(T)$ (assuming that v is the new vertex), for $\alpha \in \mathcal{F}(a)$ and $\beta \in \mathcal{F}(b)$, we say that (α, β) is a *feasible pair of types for* b if there exist $\vec{X} = (X_1, \ldots, X_m)$ and $\vec{X'} = (X'_1, \ldots, X'_m)$ realizing types α and β on the nodes a and b, respectively, such that for each i, either $X'_i = X_i$ or $X'_i = X_i \cup \{v\}$. We denote the set of feasible pairs of types of the introduce node b by $\mathcal{F}_p(b)$.

For a *forget* node $b \in V(T)$ with a child $a \in V(T)$ and for $\beta \in \mathcal{F}(b)$ and $\alpha \in \mathcal{F}(a)$, we say (α, β) is a *feasible pair of types for* b if there exists $\vec{X}$ realizing β on b and α on a. We denote the set of feasible pairs of types of the forget node b by $\mathcal{F}_p(b)$.

For a *join* node $c \in V(T)$ with children $a, b \in V(T)$ and for $\alpha \in \mathcal{F}(c)$, $\gamma_1 \in \mathcal{F}(a)$ and $\gamma_2 \in \mathcal{F}(b)$, we say that $(\gamma_1, \gamma_2, \alpha)$ is a *feasible triple of types for* c if γ_1, γ_2 and α are mutually compatible and there exist $\vec{X^1}, \vec{X^2}$ realizing γ_1 and γ_2 on a and b, respectively, such that $\vec{X} = (X_1^1 \cup X_1^2, \ldots, X_m^1 \cup X_m^2)$ realizes α on c. We denote the set of feasible triples of types of the join node c by $\mathcal{F}_t(c)$.

We define an indicator function $\nu : \mathcal{C} \times V(T) \times V(G) \times [m] \to \{0,1\}$ such that $\nu(\beta, b, v, i) = 1$ if and only if there exists $\vec{X} = (X_1, \ldots, X_m)$ realizing the type β on the node $b \in V(T)$ with $v \in B(b)$ and $v \in X_i$. Additionally, we define $\mu : \mathcal{C} \times V(G) \times [m] \to \{0,1\}$ to be $\mu(\beta, v, i) = \nu(\beta, \text{top}(v), v, i)$.

4.2 Glued Product of Polytopes over Common Coordinates

The (cartesian) *product* of two polytopes P_1 and P_2 is defined as

$$P_1 \times P_2 = \text{conv}\left(\{(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in P_1, \mathbf{y} \in P_2\}\right).$$

The following is a well known fact.

Proposition 4.2.1. *Let P_1, P_2 be two polytopes. Then*

$$\text{xc}(P_1 \times P_2) \leq \text{xc}(P_1) + \text{xc}(P_2) \ .$$

Proof. Let Q_1 and Q_2 be extended formulations of P_1 and P_2, respectively. Then, $Q_1 \times Q_2$ is an extended formulation of $P_1 \times P_2$. Now assume that $Q_1 = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ and $Q_2 = \{\mathbf{y} \mid \mathbf{C}\mathbf{y} \leq \mathbf{d}\}$ and that these are the smallest extended formulations of P_1 and P_2, resp. Then,

$$Q_1 \times Q_2 = \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{C}\mathbf{y} \leq \mathbf{d}\} \ .$$

That is, we have an extended formulation of $P_1 \times P_2$ of size at most $\text{xc}(P_1) + \text{xc}(P_2)$. □

We are going to define the glued product of polytopes, a slight generalization of the usual product of polytopes. We study a case where the extension complexity of the glued product of two polytopes is upper bounded by the sum of the extension complexities of the two polytopes and which exhibits several other nice properties. Then we use it in Section 4.3 to describe a small extended formulation for the MSO polytope $P_\varphi(G)$ on graphs with bounded treewidth.

Let $P \subseteq \mathbb{R}^{d_1+k}$ and $Q \subseteq \mathbb{R}^{d_2+k}$ be 0/1-polytopes defined by m_1 and m_2 inequalities and with vertex sets $\text{vert}(P)$ and $\text{vert}(Q)$, respectively. Let $I_P \subseteq [d_1+k]$ be a subset of coordinates of size k, $I_Q \subseteq [d_2+k]$ be a subset of coordinates of size k, and let $I'_P = [d_1 + k] \setminus I_P$. Recall that for a vector $\mathbf{x}$ and a subset of its coordinates I, we denote by $\mathbf{x}|_I$ the projection of $\mathbf{x}$ to the coordinates I. The *glued product* of P and Q, (glued) with respect to the k coordinates I_P and I_Q, denoted by $P \times_k Q$, is defined as

$$P \times_k Q = \text{conv}\left(\left\{(\mathbf{x}|_{I'_P}, \mathbf{y}) \in \mathbb{R}^{d_1+d_2+k} \mid \mathbf{x} \in \text{vert}(P), \mathbf{y} \in \text{vert}(Q), \mathbf{x}|_{I_P} = \mathbf{y}|_{I_Q}\right\}\right).$$

We adopt the following convention while discussing glued products in the rest of this chapter. In the above scenario, we say that $P \times_k Q$ is obtained by gluing P and Q along the k coordinates I_P of P with the k coordinates I_Q of Q. If, for example, these coordinates are named $\mathbf{z}$ in P and $\mathbf{w}$ in Q, then we also say that P and Q have been glued along the $\mathbf{z}$ and $\mathbf{w}$ coordinates and we refer to the

coordinates $\mathbf{z}$ and $\mathbf{w}$ as the *glued coordinates*. In the special case that we glue along the last k coordinates, the definition of the glued product simplifies to

$$P \times_k Q = \operatorname{conv}\left(\left\{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbb{R}^{d_1+d_2+k} \mid (\mathbf{x}, \mathbf{z}) \in \operatorname{vert}(P), (\mathbf{y}, \mathbf{z}) \in \operatorname{vert}(Q)\right\}\right).$$

This notion was studied by Margot [87] who provided a sufficient condition for being able to write the glued product in a specific (and efficient) way from the descriptions of P and Q. We will use this particular way in Lemma 4.2.2. The existing work [23, 87], however, is more focused on characterizing exactly when this particular method works. We do not need the result in its full generality and therefore we only state a very specific version of it that is relevant for our purposes; for the sake of completeness, we also provide a proof of it.

Lemma 4.2.2 (Gluing lemma)**.** *Let P and Q be 0/1-polytopes and let the k (glued) coordinates in P be labeled $z_1, \ldots, z_k$, and the k (glued) coordinates in Q be labeled $w_1, \ldots, w_k$. Suppose that $\mathbf{1}^\intercal \mathbf{z} \leq 1$ is valid for P and $\mathbf{1}^\intercal \mathbf{w} \leq 1$ is valid for Q. Then $\operatorname{xc}(P \times_k Q) \leq \operatorname{xc}(P) + \operatorname{xc}(Q)$.*

Proof. Let $(\mathbf{x}', \mathbf{z}', \mathbf{y}', \mathbf{w}')$ be a point from $P \times Q \cap \{(\mathbf{x}, \mathbf{z}, \mathbf{y}, \mathbf{w}) | \mathbf{z} = \mathbf{w}\}$. Observe that the point $(\mathbf{x}', \mathbf{z}')$ is a convex combination of points $(\mathbf{x}', \mathbf{0}), (\mathbf{x}', \mathbf{e}_1), \ldots, (\mathbf{x}', \mathbf{e}_k)$ from P with coefficients $(1 - \sum_{i=1}^k z'_i), z'_1, z'_2, \ldots, z'_k$ where $\mathbf{e}_i$ is the i-th unit vector. Similarly, the point $(\mathbf{y}', \mathbf{w}')$ is a convex combination of points $(\mathbf{y}', \mathbf{0}), (\mathbf{y}', \mathbf{e}_1), \ldots, (\mathbf{y}', \mathbf{e}_k)$ from Q with coefficients $(1 - \sum_{i=1}^k w'_i), w'_1, w'_2, \ldots, w'_k$. Notice that for every $j \in [k]$, $(\mathbf{x}'_j, \mathbf{e}_j, \mathbf{y}'_j)$ is a point from the glued product. As $w_i = z_i$ for every $i \in [k]$, we conclude that $(\mathbf{x}', \mathbf{w}', \mathbf{z}') \in P \times_k Q$. Thus, by Proposition 2.6.12 the extension complexity of $P \times_k Q$ is at most that of $P \times Q$ which is at most $\operatorname{xc}(P) + \operatorname{xc}(Q)$ by Proposition 4.2.1. □

We remark that in some sense the Gluing lemma is optimal. The glued product has an additive extension complexity only when the gluing is done over coordinates containing at most one 1. If some vertices of the multiplicand polytopes contain more than one 1 along the glued coordinates, then the extension complexity of the glued product cannot in general be additive, and must require a multiplicative factor strictly larger than one. We omit a proof of this claim here since the polytopes we consider do satisfy the requirement of the above lemma.

4.3 Extension Complexity of the MSO Polytope

For a given MSO_1 formula $\varphi(\vec{X})$ over σ_2 with m free set variables $X_1, \ldots, X_m$, we define a polytope of satisfying assignments on a given graph G, represented as a σ_2-structure $I(G)$ with $|V(I(G))| = n$, in a natural way as the convex hull of $S_\varphi(G)$, that is,

$$P_\varphi(G) = \operatorname{conv}\left(\{\mathbf{y} \in \{0,1\}^{nm} \mid \mathbf{y} \text{ satisfies } \varphi\}\right).$$

For the sake of simplicity, we state the following theorem and carry out the exposition for *graphs*; however, identical arguments can be carried out analogously for any σ_2-structure (or any fixed signature σ) whose Gaifman graph has treewidth bounded by τ.

Theorem 4.3.1 (Extension complexity of the MSO polytope)**.** *For every graph G represented as $I(G)$ and for every MSO_1 formula φ over σ_2, $\mathrm{xc}(P_\varphi(G)) \leq f(|\varphi|, \tau) \cdot n$ where f is some computable function, $\tau = \mathrm{tw}(G)$ and $n = |V_I|$.*

Proof. Let (T, B) be a fixed nice tree decomposition of treewidth τ of $I(G)$ and let k denote the quantifier rank of φ and m the number of free variables of φ. Let $\mathcal{C}$ be the set of equivalence classes of the relation $\equiv_k^{MSO}$. For each node b of T we introduce $|\mathcal{C}|$ binary variables that will represent a feasible type of the node b; we denote the vector of them by $\mathbf{t}_b$ (i.e., $\mathbf{t}_b \in \{0,1\}^{|\mathcal{C}|}$). For each introduce and each forget node b of T, we introduce additional $|\mathcal{C}|$ binary variables that will represent a feasible type of the child (descendant) of b; we denote the vector of them by $\mathbf{d}_b$ (i.e., $\mathbf{d}_b \in \{0,1\}^{|\mathcal{C}|}$). Similarly, for each join node b we introduce additional $|\mathcal{C}|$ binary variables, denoted by $\mathbf{l}_b$, that will represent a feasible type of the left child of b, and other $|\mathcal{C}|$ binary variables, denoted by $\mathbf{r}_b$, that will represent a feasible type of the right child of b (i.e., $\mathbf{l}_b, \mathbf{r}_b \in \{0,1\}^{|\mathcal{C}|}$).

We are going to describe inductively a polytope in the dimension given (roughly) by all the binary variables of all nodes of the given nice tree decomposition. Then we show that its extension complexity is small and that a properly chosen face of it is an extension of $P_\varphi(G)$.

First, for each node b of T, depending on its type, we define a polytope P_b as follows:

- b is a *leaf*. P_b consists of a single point $P_b = \{\overbrace{100\ldots0}^{|\mathcal{C}|}\}$.
- b is an *introduce* or *forget* node. For each feasible pair of types $(\alpha_i, \alpha_j) \in \mathcal{F}_p(b)$ of the node b, we create a vector $(\mathbf{d}_b, \mathbf{t}_b) \in \{0,1\}^{2|\mathcal{C}|}$ with $d_b(i) = t_b(j) = 1$ and all other coordinates zero. P_b is defined as the convex hull of all such vectors.
- b is a *join* node. For each feasible triple of types $(\alpha_h, \alpha_i, \alpha_j) \in \mathcal{F}_t(b)$ of the node b, we create a vector $(\mathbf{l}_b, \mathbf{r}_b, \mathbf{t}_b) \in \{0,1\}^{3|\mathcal{C}|}$ with $l_b(h) = r_b(i) = t_b(j) = 1$ and all other coordinates zero. P_b is defined as the convex hull of all such vectors.

It is clear that for every node b in T, the polytope P_b contains at most $|\mathcal{C}|^3$ vertices, and, thus, by Proposition 2.6.11 it has extension complexity at most $\mathrm{xc}(P_b) \leq |\mathcal{C}|^3$. Recalling our discussion in Section 4.1 about the size of $\mathcal{C}$, we conclude that there exists a function f such that for every $b \in V(T)$, it holds that $\mathrm{xc}(P_b) \leq f(|\varphi|, \tau)$.

We create an extended formulation for $P_\varphi(G)$ by gluing these polytopes together, starting in the leaves of T and processing T in a bottom up fashion. We create polytopes Q_b for each node b in T recursively as follows:

- If b is a leaf then $Q_b = P_b$.
- If b is an introduce or forget node, then $Q_b = Q_a \times_{|\mathcal{C}|} P_b$ where a is the child of b and the gluing is done along the coordinates $\mathbf{t}_a$ in Q_a and $\mathbf{d}_b$ in P_b.
- If b is a join node, then we first define $R_b = Q_a \times_{|\mathcal{C}|} P_b$ where a is the left child of b and the gluing is done along the coordinates $\mathbf{t}_a$ in Q_a and $\mathbf{l}_b$ in P_b. Then Q_b is obtained by gluing R_b with Q_c along the coordinates $\mathbf{t}_c$ in Q_c and r_b in R_b where c is the right child of b.

The following lemma states the key property of the polytopes Q_b.

Lemma 4.3.2. *For every vertex* $\mathbf{y}$ *of the polytope* Q_b *there exist* $X_1, \ldots, X_m \subseteq V(G_b)$ *such that* $(G_b, (X_1, \ldots, X_m), \eta(B(b)))$ *is of type* α *where* α *is the unique type such that the coordinate of* y *corresponding to the binary variable* $t_b(\alpha)$ *is equal to one.*

Proof. The proof is by induction, starting in the leaves of T and going up towards the root. For leaves, the lemma easily follows from the definition of the polytopes P_b.

For the inductive step, we consider an inner node b of T and we distinguish two cases:

- If b is a join node, then the claim for b follows from the inductive assumptions for the children of b, definition of a feasible triple, definition of the polytope P_b, Lemma 4.1.2 and the construction of the polytope Q_b.

- If b is an introduce node or a forget node, respectively, then, analogously, the claim for b follows from the inductive assumption for the child of b, definition of a feasible pair, definition of the polytope P_b, Lemma 4.1.3 or Lemma 4.1.4, respectively, and the construction of the polytope Q_b.

□

Let c be the root node of the tree decomposition T. Consider the polytope Q_c. From the construction of Q_c, our previous discussion and the Gluing lemma, it follows that $\mathrm{xc}(Q_c) \leq \sum_{b \in V(T)} \mathrm{xc}(P_b) \leq f(|\varphi|, \tau) \cdot n$. It remains to show that a properly chosen face of Q_c is an extension of $P_\varphi(G)$. We start by observing that $\sum_{i=1}^{|\mathcal{C}|} t_c(i) \leq 1$ and $\sum_{i=1}^{|\mathcal{C}|} \rho_\varphi(i) \cdot t_c(i) \leq 1$, where ρ_φ is the indicator function defined in Subsection 4.1.2, are valid inequalities for Q_c.

Let Q_φ be the face of Q_c corresponding to the valid inequality $\sum_{i=1}^{|\mathcal{C}|} \rho_\varphi(i) \cdot t_c(i) \leq 1$. Then, by Lemma 4.3.2, the polytope Q_φ represents those $[m]$-colorings of G for which φ holds. The corresponding feasible assignments of φ on G are obtained as follows: for every vertex $v \in V(G)$ and every $i \in [m]$ we set $y_v^i = \sum_{j=1}^{|\mathcal{C}|} \mu(\alpha_j, v, i) \cdot t_{\mathrm{top}(v)}(j)$. The sum is 1 if and only if there exists a type j such that $t_{\mathrm{top}(v)}(j) = 1$ and at the same time $\mu(\alpha_j, v, i) = 1$; by the definition of the indicator function μ in Subsection 4.1.3, this implies that $v \in X_i$. Thus, by applying the above projection to Q_φ we obtain $P_\varphi(G)$, as desired.

It is worth mentioning at this point that the polytope Q_c depends only on the treewidth τ, the quantifier rank k of φ and the number of free variables of φ. The dependence on the formula φ itself only manifests in the choice of the face Q_φ of Q_c and its projection to $P_\varphi(G)$. □

Corollary 4.3.3. *The extension complexity of the convex hull of all satisfying assignments of a given* MSO_2 *formula* φ *on a given graph* G *of bounded treewidth is linear in the size of the graph* G*.*

4.4 Efficient Construction of the MSO Polytope

In the previous section we have proven that $P_\varphi(G)$ *has* a compact extended formulation but our definition of feasible tuples and the indicator functions μ and ρ_φ did not explicitly provide a way how to actually *obtain* it efficiently. That is what we do in this section.

As in the previous section we assume that we are given a graph G of treewidth τ and an MSO formula φ with m free variables and quantifier rank k. We start by constructing a nice tree decomposition (T, B) of G of treewidth τ in time $f(\tau) \cdot n$ [11, 66].

Let $\mathcal{C}$ denote the set of equivalence classes of $\equiv_k^{MSO}$. Because $\mathcal{C}$ is finite and its size is independent of the size of G (Theorem 4.1.1), for each class $\alpha \in \mathcal{C}$, there exists an $[m]$-colored τ-boundaried graph $(G^\alpha, \vec{X}^\alpha, \vec{p^\alpha})$ of type α whose size is upper-bounded by a function of k, m and τ. For each $\alpha \in \mathcal{C}$, we fix one such graph, denote it by $W(\alpha)$ and call it the *witness of* α. Let $\mathcal{W} = \{W(\alpha) \mid \alpha \in \mathcal{C}\}$. The witnesses make it possible to easily compute the indicator function ρ_φ: for every $\alpha \in \mathcal{C}$, we set $\rho_\varphi(\alpha) = 1$ if and only if $W(\alpha) \models \varphi$ (which can be evaluated straightforwardly), and we set $\rho_\varphi(\alpha) = 0$ otherwise.

The following Lemma is implicit in [53] in the proof of Theorem 4.6 and Corollary 4.7.

Lemma 4.4.1 ([53])**.** *The set $\mathcal{W}$ and the indicator function ρ_φ can be computed in time $f(k, m, \tau)$, for some computable function f.*

It will be important to have an efficient algorithmic test for MSO$[k, \tau]$-elementary equivalence. This can be done using the Ehrenfeucht-FraÃŕssÃŀ games:

Lemma 4.4.2 ([83, Theorem 7.7])**.** *Given two $[m]$-colored τ-boundaried graphs $G_1^{[m],\tau}$ and $G_2^{[m],\tau}$, it can be decided in time $f(m, k, \tau, |G_1|, |G_2|)$ whether $G_1^{[m],\tau} \equiv_k^{MSO} G_2^{[m],\tau}$, for some computable function f.*

Corollary 4.4.3. *Recognizing the type of an $[m]$-colored τ-boundaried graph $G^{[m],\tau}$ can be done in time $f(m, k, \tau, |G|)$, for some computable function f.*

Now we describe a linear time construction of the sets of feasible types, pairs and triples of types $\mathcal{F}(b)$, $\mathcal{F}_p(b)$ and $\mathcal{F}_t(b)$ for all relevant nodes b in T. In the initialization phase we construct the set $\mathcal{W}$, using the algorithm from Lemma 4.4.1. The rest of the construction is inductive, starting in the leaves of T and advancing in a bottom up fashion towards the root of T. The idea is to always replace a possibly *large* graph $G_a^{[m],\tau}$ of type α by the *small* witness $W(\alpha)$ when computing the set of feasible types for the father of a node a.

Leaf node. For every leaf node $a \in V(T)$ we set $\mathcal{F}(a) = \{\alpha_1\}$. Obviously, this corresponds to the definition in Section 4.1.

Introduce node. Assume that $b = a * (v)$ is an introduce node for which $\mathcal{F}(a)$ has already been computed. For every $\alpha \in \mathcal{F}(a)$, we first produce a τ'-boundaried graph $H^{\tau'} = (H^\alpha, \vec{q})$ from $W(\alpha) = (G^\alpha, \vec{X^\alpha}, \vec{p^\alpha})$ as follows: let $\tau' = |\vec{p^\alpha}| + 1$ and H^α be obtained from G^α by attaching to it a new vertex in the same way as v is attached to G_a. The boundary $\vec{q}$ is obtained from the boundary $\vec{p^\alpha}$ by inserting in it the new vertex at the same position that v has in the boundary of

$(G_a, \eta(B(a)))$. For every subset $I \subseteq [m]$ we construct an $[m]$-coloring $\vec{Y}^{\alpha,I}$ from $\vec{X}^{\alpha}$ by setting $Y_i^{\alpha,I} = X_i^{\alpha} \cup \{v\}$, for every $i \in I$, and $Y_i^{\alpha,I} = X_i^{\alpha}$, for every $i \notin I$. Each of these $[m]$-colorings $\vec{Y}^{\alpha,I}$ is used to produce an $[m]$-colored τ'-boundaried graph $(H^{\alpha}, \vec{Y}^{\alpha,I}, \vec{q})$ and the types of all these $[m]$-colored τ'-boundaried graphs are added to the set $\mathcal{F}(b)$ of feasible types of b, and, similarly, the pairs (α, β) where β is a feasible type of some of the $[m]$-colored τ'-boundaried graph $(H^{\alpha}, \vec{Y}^{\alpha,I}, \vec{q})$, are added to the set $\mathcal{F}_p(b)$ of all feasible pairs of types of b. The correctness of the construction of the sets $\mathcal{F}(b)$ and $\mathcal{F}_p(b)$ for the node b of T follows from Lemma 4.1.3.

Forget node. Assume that $b = a \dagger (v)$ is a forget node for which $\mathcal{F}(a)$ has already been computed and where v is the d-th vertex of the boundary $\eta(B(a))$. We proceed in a similar way as in the case of the introduce node. For every $\alpha \in \mathcal{F}(a)$ we produce an $[m]$-colored τ'-boundaried graph $(H^{\alpha}, \vec{Y}^{\alpha}, \vec{q})$ from $W(\alpha) = (G^{\alpha}, \vec{X}^{\alpha}, \vec{p^{\alpha}})$ as follows: let $\tau' = |\vec{p^{\alpha}}| - 1$, $H^{\alpha} = G^{\alpha}$, $\vec{Y}^{\alpha} = \vec{X}^{\alpha}$ and $\vec{q} = (p_1, \ldots, p_{d-1}, p_{d+1}, \ldots, p_{\tau'+1})$. For every $\alpha \in \mathcal{F}(a)$, the type β of the constructed graph is added to $\mathcal{F}(b)$, and, similarly, the pairs (α, β) are added to $\mathcal{F}_p(b)$. The correctness of the construction of the sets $\mathcal{F}(b)$ and $\mathcal{F}_p(b)$ for the node b of T follows from Lemma 4.1.4.

Join node. Assume that $c = \Lambda(a, b)$ is a join node for which $\mathcal{F}(a)$ and $\mathcal{F}(b)$ have already been computed. For every pair of compatible types $\alpha \in \mathcal{F}(a)$ and $\beta \in \mathcal{F}(b)$, we add the type γ of $W(\alpha) \oplus W(\beta)$ to $\mathcal{F}(c)$, and the triple (α, β, γ) to $\mathcal{F}_t(c)$. The correctness of the construction of the sets $\mathcal{F}(c)$ and $\mathcal{F}_t(c)$ for the node b of T follows from Lemma 4.1.2.

It remains to construct the indicator functions ν and μ. We do it during the construction of the sets of feasible types as follows. We initialize ν to zero. Then, every time we process a node b in T and we find a new feasible type β of b, for every $v \in B(b)$ and for every i for which the d-th vertex in the boundary of $W(\beta) = (G^{\beta}, \vec{X}, \vec{p})$ belongs to X_i, we set $\mu(\beta, b, v, i) = 1$ where d is the order of v in the boundary of $(G_b, \eta(B(b))$. The correctness follows from the definition of ν and the definition of feasible types. The function μ is then straightforwardly defined using ν.

Concerning the time complexity of the inductive construction, we observe, exploiting Corollary 4.4.3, that for every node b in T, the number of steps, the sizes of graphs that we worked with when dealing with the node b, and the time needed for each of the steps, depend on k, m and τ only. We summarize the main result of this section in the following theorem.

Theorem 4.4.4. *Under the assumptions of Theorem 4.3.1, the polytope $P_{\varphi}(G)$ can be constructed in time $f'(|\varphi|, \tau) \cdot n$, for some computable function f'.*

4.5 Extensions

4.5.1 Cliquewidth

The results of this paper can be extended also to graphs of bounded *cliquewidth*, a more general class of graphs, at the cost of restricting our logic from MSO_2 to MSO_1 (or, equivalently, restricting it from MSO_1 over σ_2 to MSO_1 over σ_1). Recall that a γ-expression is a concept analogous to a tree decomposition.

Theorem 4.5.1. *Let G be a graph of cliquewidth $cw(G) = \gamma$ represented as a σ_1-structure, let ψ be an MSO_1 formula with m free variables, and let $P_\psi(G) = \mathrm{conv}(S_\psi(G))$. Then $\mathrm{xc}(P_\psi(G)) \leq f(|\psi|, \gamma) \cdot n$ for some computable function f.*

Moreover, if G is given along with its γ-expression Γ, $P_\psi(G)$ can be constructed in time $f(|\psi|, \gamma) \cdot n$.

While we could prove Theorem 4.5.1 directly using notions analogous to feasible types and tuples, it is easier to use a tool demonstrating a close relationship between cliquewidth and treewidth. This tool simply extends a folklore fact that a class of graphs is of bounded cliquewidth if and only if it has an MSO_1 interpretation in the class of rooted trees.

Lemma 4.5.2 (Treewidth–cliquewidth correspondence)**.** *Let G be a graph of cliquewidth $cw(G) = \gamma$ given along with its γ-expression Γ, and let ψ be an MSO_1 formula. One can, in time $\mathcal{O}(|V(G)| + |\Gamma| + |\psi|)$, compute a tree T and an MSO_1 formula φ such that $V(G) \subseteq V(T)$ and*

$$\text{for every } X,\ T \models \varphi(X),\ \text{if and only if } X \subseteq V(G) \text{ and } G \models \psi(X).$$

Addition to Lemma 4.5.2. Before giving a (short) proof of this lemma, we need to formally introduce a simplified concept of MSO interpretability. Let σ and ϱ be two relational vocabularies. A *one-dimensional* MSO *interpretation* of ϱ in σ is a tuple $I = \Big(\nu(x),\ \{\eta_R(\bar{x})\}_{R \in \varrho}\Big)$ of $\mathsf{MSO}[\sigma]$-formulas where ν has one free element variable and the number of free element variables in η_R is equal to the arity of R in ϱ.

- To every σ-structure A the interpretation I assigns a ϱ-structure A^I with the domain $A^I = \{a \mid A \models \nu(a)\}$ and the relations $R^I = \{\bar{a} \mid A \models \eta_R(\bar{a})\}$ for each $R \in \varrho$. We say that a class $\mathcal{C}$ of ϱ-structures has an interpretation in a class $\mathcal{D}$ of σ-structures if there exists an interpretation I such that for each $C \in \mathcal{C}$ there exists $D \in \mathcal{D}$ such that $C \simeq D^I$, and for every $D \in \mathcal{D}$ the structure D^I is isomorphic to a member of $\mathcal{C}$.
- The interpretation I of ϱ in σ defines a translation of every $\mathsf{MSO}[\varrho]$-formula ψ to an $\mathsf{MSO}[\sigma]$-formula ψ^I as follows:
  - every $\exists x.\phi$ is replaced by $\exists x.(\nu(x) \wedge \phi^I)$,
  - every $\exists X.\phi$ is replaced by $\exists X.(\forall y(y \in X \to \nu(y)) \wedge \phi^I)$, and
  - every occurrence of a σ-atom $R(\bar{x})$ is replaced by the corresponding formula $\eta_R(\bar{x})$.

It is a folklore fact that for all $\mathsf{MSO}[\varrho]$-formulas ψ and all σ-structures A

$$A \models \psi^I \iff A^I \models \psi.$$

Proof of Lemma 4.5.2. Let T_0 be the parse tree of the given γ-expression Γ constructing G. Hence T_0 is a rooted tree such that the set of its leaves is $V(G)$. It is well known that there is a one-dimensional MSO_1 interpretation $I_1 = \Big(\nu_1(x), \eta_1(x, y)\Big)$ of the graphs of cliquewidth $\leq \gamma$ in the class of colored rooted trees (such that the finite set of colors depends only on γ). The used

colors are in fact the vertex labels (in the leaves) and the operator symbols (in the internal nodes) from Definition 2.3.7, and the interpretation I_1 is easy to construct for given γ. See, e.g., [74, Section 4.3] for close details. Consequently, $G \simeq T_0^{I_1}$.

To finish the proof, we just need to "remove" the colors from T_0 and "forget" the root (to make an ordinary uncolored tree T). We first give the root of T_0 a new distinguished color c_r (as a copy of its original color). Then the parent-child relation in T_0 can be easily interpreted based on the unique path to a vertex colored c_r. Let all the colors used in I_1 be $C = \{c_1, \ldots, c_m\}$ where m depends on γ (including our distinguished root colors). We construct a tree T from T_0 by attaching i new leaves to every vertex of T_0 of color $c_i \in C$.

We now straightforwardly define an MSO_1 interpretation $I_2 = \big(\nu_2(x), \eta_2(x,y), \lambda_2^i \text{ for } i \in [m]\big)$ of rooted C-colored trees in the class of ordinary trees:

- $\nu_2(x)$ simply asserts that x is not a leaf (x has more than one neighbor),
- $\eta_2(x,y) \equiv \mathrm{edge}(x,y)$ (note that T_0 is an induced subgraph of T), and
- $\lambda_2^i(x)$ asserts that x has precisely i neighbors which are leaves—this has a routine brute-force expression by existential quantification of the i leaf neighbors, coupled by non-existence of $i+1$ leaf neighbors.

Clearly, $T^{I_2} \simeq T_0$ (including the colors of T_0). We finally set $I = I_1 \circ I_2$ (interpretability is a transitive concept) and $\varphi = \psi^I$ and we are done. □

Proof of Theorem 4.5.1. Let Γ be some γ-expression of G. Consider the tree T of Lemma 4.5.2 derived from G and γ and the MSO_1 formula φ derived from ψ. By the relationship between G and T and ψ and φ of Lemma 4.5.2, the polytope $P_\varphi(T)$ is an extended formulation of $P_\psi(G)$. Thus, by applying Theorem 4.3.1 to T and φ suffices to show that $\mathrm{xc}(P_\psi(G)) \leq \mathrm{xc}(P_\varphi(T)) \leq f(|\psi|, \gamma) \cdot n$ for some computable function f. If a γ-expression of G is provided, clearly the proof becomes constructive. □

For simplicity, we have required that G comes along with its γ-expression to obtain a constructivity in Theorem 4.5.1. See our remark in Subsection 2.3.3 about using rank-decompositions as approximations of γ-expressions.

4.5.2 Courcelle's Theorem and Optimization.

It is worth noting that even though linear time *optimization* versions of Courcelle's theorem are known, our result provides a linear size LP for these problems out of the box. Together with a polynomial algorithm for solving linear programming we immediately get the following:

Theorem 4.5.3. *Given a graph G on n vertices with treewidth τ, a formula $\varphi \in \mathsf{MSO}$ with m free variables and real weights w_v^i, for every $v \in V(G)$ and $i \in [m]$ the problem*

$$\mathit{opt}\left\{ \sum_{v \in V(G)} \sum_{i=1}^{m} w_v^i \cdot y_v^i \;\middle|\; \mathbf{y} \text{ satisfies } \varphi \right\}$$

where opt is min *or* max*, is solvable in time $f(|\varphi|, \tau) \cdot n^{\mathcal{O}(1)} \cdot \langle \mathbf{w} \rangle$, for some function f, where $\langle \mathbf{w} \rangle$ is the encoding length of $\mathbf{w}$.*

4.5.3 Total Unimodularity

We believe there is also an alternative way to prove Theorem 4.3.1 by exploiting the well known fact that polyhedra defined by totally unimodular matrices are integral. However, at the moment we are only able to demonstrate this approach for graphs of bounded pathwidth. This will still provide us with some insight, as we discuss in Subsection 7.3.1.

It is known that the polytope of s–t dipaths in a directed graph $D = (W, A)$ is integral by the fact that its defining matrix is totally unimodular [107]. We shall define a directed graph D with vertices s and t whose s–t dipaths will be closely related to the vertices of the previously constructed polytope P. A suitable integer projection then provides the polytope $P_\varphi(G)$.

Theorem 4.5.4. *Let G be a graph with $\mathrm{pw}(G) = \tau$ and φ an MSO_1 formula of quantifier depth k with m free variables. Then $\mathrm{xc}(P_\varphi(G) \leq f(\tau, k, m) \cdot |V(G)|$ for some computable function f.*

Proof. Let (T, B) be an optimal path decomposition of G with N nodes labeled $b_1, \ldots, b_N$. We abuse the notation slightly and use $\mathrm{top}(v)$ not only as the last node b_i containing v, but also as its index i. We shall construct a directed graph $D = (W, A)$ with two distinguished vertices s and t. This graph is layered and acyclic, with arcs going only between consecutive layers. We denote the set of vertices of layer i by $\mathcal{L}_i$. The layers and arcs are as follows.

- $\mathcal{L}_0 = \{s\}$ and $\mathcal{L}_{N+1} = \{t\}$.
- For $i \in [N]$, $\mathcal{L}_i = \{(i, \alpha) \mid \alpha \in \mathcal{F}(b_i)\}$, i.e., layer i contains one vertex for each feasible type from $\mathcal{F}(b_i)$.
- There is an arc from s to vertex $(1, \alpha_1)$; note that since b_1 has to be a leaf, its only feasible type is α_1, i.e., $\mathcal{L}_1 = \{(1, \alpha_1)\}$.
- There is an arc from (N, α) to t for every $\alpha \in \mathcal{F}(b_N)$ with $\rho_\varphi(\alpha) = 1$, i.e., for every type α which makes the formula φ satisfied.
- Finally, let $i \in [N-1]$. There is an arc between a vertex $(i, \alpha) \in \mathcal{L}_i$ and a vertex $(i+1, \beta) \in \mathcal{L}_{i+1}$ if $(\alpha, \beta) \in \mathcal{F}_p(b_{i+1})$; note that since a path decomposition only contains introduce and forget nodes, this definition is sound.

Fix an s–t dipath in D and consider its vertices. Omitting s and t, they can be viewed as an N-tuple $(\alpha_1, \ldots, \alpha_N)$ of types satisfying that every consecutive pair is a feasible pair for the respective node, and that $\rho_\varphi(\alpha_N) = 1$. By an argument similar to that of Lemma 4.3.2, we see that taking $y_v^i = \mu(\alpha_{\mathrm{top}(v)}, v, i)$ produces a vector $\mathbf{y}$ encoding a satisfying assignment to φ.

Now, consider the polytope of s–t dipaths in D:

$$P_{s-t}(D) = \{\mathbf{x} \mid \mathbf{x} \text{ is an indicator of an } s\text{–}t \text{ dipath in } D\} \subseteq \{0,1\}^A,$$

and a projection $\pi : \mathbb{R}^A \to \mathbb{R}^{nm}$:

$$y_v^i = \sum_{\alpha \in \mathcal{F}(\mathrm{top}(v))} \mu(\alpha, v, i) \cdot \sum_{a \in N_D^+((\mathrm{top}(v), \alpha))} x_a \ .$$

By the fact that π is an integer projection, and by our argument above, $P_\varphi(G) = \pi(P_{s-t}(D))$. Finally, since $P_{s-t}(D) = \{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b}, \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}$ where $\mathbf{A}$ is a totally unimodular matrix of flow-preservation equalities for every vertex, we have that $\mathrm{xc}(P_{s-t}(D)) \leq |W| + 2|A| = f(\tau, k, m) \cdot n$, concluding the proof. □

Observe that we immediatelly obtain that $P_\varphi(G)$ has a constructively decomposable extension by the fact that $P_{s-t}(D)$ is constructively decomposable (since it is defined by a TU system) and $P_\varphi(G)$ is obtaiend from it by an integer projection (Lemma 5.1.3).

We note the results of Tiwary [113] on the extension complexity of formal languages defined by certain a certain automaton. It is not difficult to see that we can construct a digraph whose paths correspond to computations of said automaton and thus obtain their results using the ideas sketched in the proof above.

5. Connecting MSO, CSP, and Treewidth

In Section 5.1 we show further properties of the glued product, and then use it to prove that there is an extension of the MSO polytope which has these desirable properties. Then, in Section 5.2, we show connections between MSO and CSP for graphs and instances of bounded treewidth, and between CSP and integer separable minimization.

5.1 MSO Polytope: Decomposability and Treewidth

5.1.1 Decomposability of Polyhedra

Now we will define *decomposable* polyhedra and show that decomposability is preserved by taking the glued product. Decomposability is also known as *integer decomposition property* or being *integrally closed* in the literature (cf. Schrijver [107]). The best known example are polyhedra given by totally unimodular matrices [7].

Definition 5.1.1 (Decomposable polyhedron, Decomposition oracle)**.** *A polyhedron $P \subseteq \mathbb{R}^n$ is* decomposable *if for every $r \in \mathbb{N}$ and every $\mathbf{x} \in rP \cap \mathbb{Z}^n$, there exist $\mathbf{x}^1, \dots, \mathbf{x}^r \in P \cap \mathbb{Z}^n$ with $\mathbf{x} = \mathbf{x}^1 + \cdots + \mathbf{x}^r$, where $rP = \{r\mathbf{y} \mid \mathbf{y} \in P\}$ is called the* r-dilate of P. *A* decomposition oracle *for a decomposable P is one that, queried on $r \in \mathbb{N}$ and on $\mathbf{x} \in rP \cap \mathbb{Z}^n$, returns its* decomposition $\mathbf{x}^1, \dots, \mathbf{x}^r \in P \cap \mathbb{Z}^n$ *with $\mathbf{x} = \mathbf{x}^1 + \cdots + \mathbf{x}^r$. If a decomposition oracle for P is realizable by an algorithm running in time polynomial in the length of the unary encoding of r and $\mathbf{x}$, we say that P is* constructively decomposable.

Lemma 5.1.2 (Decomposability and glued product)**.** *Let $P \subseteq \mathbb{R}^{d_1+k}$ and $Q \subseteq \mathbb{R}^{d_2+k}$ be 0/1-polytopes and let the k glued coordinates in P be labeled $z_1, \dots, z_k$, and the k glued coordinates in Q be labeled $w_1, \dots, w_k$. Suppose that $\mathbf{1}^\intercal \mathbf{z} \leq 1$ is valid for P and $\mathbf{1}^\intercal \mathbf{w} \leq 1$ is valid for Q. Then if P and Q are constructively decomposable, so is $P \times_k Q$.*

Proof. For the sake of simplicity, we assume without loss of generality that glueing is done along the last k coordinates. Then $P \times_k Q = \operatorname{conv}\{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbb{R}^{d_1+d_2+k} \mid (\mathbf{x}, \mathbf{z}) \in \operatorname{vert}(P), (\mathbf{y}, \mathbf{w}) \in \operatorname{vert}(Q), \mathbf{z} = \mathbf{w}\}$. Let $R = P \times_k Q$. To prove that R is constructively decomposable, it suffices to find, for every integer $r \in \mathbb{N}$ and every integer vector $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in rR$, r integer vectors $(\mathbf{x}^i, \mathbf{y}^i, \mathbf{z}^i) \in R$ such that $(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{i=1}^r (\mathbf{x}^i, \mathbf{y}^i, \mathbf{z}^i)$. Using the assumption that P and Q are constructively decomposable, we find in polynomial time r integer vectors $(\mathbf{x}^i, \mathbf{z}^i) \in P$ such that $(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^r (\mathbf{x}^i, \mathbf{z}^i)$ and r integer vectors $(\mathbf{y}^j, \bar{\mathbf{z}}^j) \in Q$ such that $(\mathbf{y}, \mathbf{z}) = \sum_{j=1}^r (\mathbf{y}^j, \bar{\mathbf{z}}^j)$.

Observe that $\mathbf{z} = \sum_{i=1}^r \mathbf{z}^i = \sum_{j=1}^r \bar{\mathbf{z}}^j$. Moreover, because $\mathbf{z}^i$ and $\bar{\mathbf{z}}^j$ satisfy $\mathbf{1}^\intercal \mathbf{z}^i \leq 1$ and $\mathbf{1}^\intercal \bar{\mathbf{z}}^j \leq 1$ for all i and j, respectively, each vector $\mathbf{z}^i$ and each vector

$\bar{\mathbf{z}}^j$ contains at most one 1. Clearly, the number of vectors $\mathbf{z}^i$ with $\mathbf{z}^i_l = 1$ is equal to the number of vectors $\bar{\mathbf{z}}^j$ with $\bar{\mathbf{z}}^j_l = 1$, namely z_l.

Thus, it is possible to greedily pair the vectors $(\mathbf{x}^i, \mathbf{z}^i)$ and $(\mathbf{y}^j, \bar{\mathbf{z}}^j)$ one to one in such a way that $\mathbf{z}^i = \bar{\mathbf{z}}^j$ for all the paired vectors. By merging each such pair of vectors, we obtain r new integer vectors $(\mathbf{x}^l, \mathbf{y}^l, \mathbf{z}^l) \in R$, for $l \in [r]$, that satisfy $(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{l=1}^{r} (\mathbf{x}^l, \mathbf{y}^l, \mathbf{z}^l)$, concluding the proof. □

We remark that this proof is very similar to the proof of Lemma 3.1.1. In fact, having the insight of that proof allowed us to realize has the nice properties shown in Lemmas 5.1.2 and 5.1.6.

It is typical that we construct some polytope and apply an integer projection to it. The following lemma will be useful in such situations:

Lemma 5.1.3 (Decomposability and projections)**.** *Let $Q \subseteq \mathbb{R}^{n'}$ be a polyhedron which is constructively decomposable, let $\pi : \mathbb{R}^{n'} \to \mathbb{R}^n$ be a linear projection with integer coefficients and let $P = \pi(Q)$. Then the polytope $R = \operatorname{conv}(\{(\mathbf{y}, \pi(\mathbf{y})) \mid \mathbf{y} \in Q\})$ is constructively decomposable.*

Proof. Consider an integer r and an integer vector $(\mathbf{x}, \mathbf{y}) \in rR$. Since Q is constructively decomposable, we can find in polynomial time r vectors $\mathbf{y}^i \in Q \cap \mathbb{Z}^{n'}$, for $i \in [r]$, such that $\mathbf{y} = \sum_{i=1}^{r} \mathbf{y}^i$. For every i, let $\mathbf{x}^i = \pi(\mathbf{y}^i)$; note that every $\mathbf{x}^i$ is integral. Since $\mathbf{x} = \pi(\mathbf{y}) = \sum_i \pi(\mathbf{y}^i) = \sum_i \mathbf{x}^i$, we conclude that $(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{r} ((\mathbf{x}^i, \mathbf{y}^i))$, proving that R is constructively decomposable. □

Obviously, not all integer polyhedra are decomposable: consider the three-dimensional parity polytope $P = \operatorname{conv}(\{(0,0,0), (1,1,0), (1,0,1), (0,1,1)\})$ and the point $(1,1,1) \in 2P$ – there is no way to express it as a sum of integral points in P. However, the following lemma shows that every integer polyhedron has an extension that is decomposable.

Lemma 5.1.4 (Decomposable extension)**.** *Every integer polytope P has an extension R that is constructively decomposable. Moreover, a description of such an extension can be computed in time $\mathcal{O}(d + n)$ where d is the dimension of P and n is the number of vertices of P if the vertices of P are given.*

Proof. Recall that $\operatorname{vert}(P) = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ denotes all the vertices of P and let $Q = \{\boldsymbol{\lambda} \mid \sum_{i=1}^{n} \lambda_i = 1, \boldsymbol{\lambda} \geq \mathbf{0}\}$ be the n-dimensional simplex. Then, for the linear projection $\pi(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i$, the polytope $R = \{(\pi(\boldsymbol{\lambda}), \boldsymbol{\lambda}) \mid \boldsymbol{\lambda} \in Q\}$ is an extended formulation of P (note that the same extended formulation of P is used also in the proof of Proposition 2.6.11).

Consider an arbitrary integer r and an integral point $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n) \in rQ$. As $\boldsymbol{\lambda} = \sum_{j=1}^{n} \lambda_j \mathbf{e}_j$ where $\mathbf{e}_j$ is the j-the unit vector, and as $\sum_{j=1}^{n} \lambda_j = r$, we see that $\boldsymbol{\lambda}$ can be written as a sum of at most r integral points from Q, and such a decomposition can be found in time polynomial in n and d. Thus, Q is constructively decomposable. Then, applying the previous lemma to the simplex Q and the linear projection π, we see that the polytope R is constructively decomposable. □

Given a polytope P, it is an interesting problem to determine the minimum size of an extension of P that is decomposable. This is an analogue of extension complexity:

Definition 5.1.5 (Decomposable extension complexity)**.** *The* decomposable extension complexity *of a polytope P, denoted* $\mathrm{xc}_{dec}(P)$*, is the minimum size of an extension of P that is decomposable. A polytope Q which is an extension of P and is decomposable is called a* decomposable extension of P*.*

Obviously, $\mathrm{xc}(P) \leq \mathrm{xc}_{dec}(P)$. Using Lemma 5.1.3 and Proposition 2.6.11 we see that if a polytope P has n vertices, then $\mathrm{xc}_{dec}(P) \leq n$. It is an interesting problem to determine for which polytopes $\mathrm{xc}(P) = \mathrm{xc}_{dec}(P)$, or, on the other hand, when $\mathrm{xc}(P) < \mathrm{xc}_{dec}(P)$ and by how much they can differ.

5.1.2 Treewidth of Gaifman Graphs of Extended Formulations

Recall that the treewidth of a matrix is the treewidth of its Gaifman graph, as defined in Subsection 2.3.1.

Lemma 5.1.6 (Treewidth and glued product)**.** *Let P and Q be $0/1$-polytopes and let the k glued coordinates in P be labeled $z_1, \ldots, z_k$ and the k glued coordinates in Q be labeled $w_1, \ldots, w_k$. Suppose that $\mathbf{1}^\intercal \mathbf{z} \leq 1$ is valid for P and $\mathbf{1}^\intercal \mathbf{w} \leq 1$ is valid for Q. Let $\mathbf{Ax} + \mathbf{Cz} \geq \mathbf{a}$ be inequalities describing an extended formulation of P and $\mathbf{Dw} + \mathbf{Ey} \geq \mathbf{b}$ be inequalities describing an extended formulation of Q. Then there exists an extended formulation of $P \times_k Q$ described by inequalities $\mathbf{Fv} \geq \mathbf{c}$ such that $\mathrm{tw}(\mathbf{F}) \leq \max\{\mathrm{tw}(\mathbf{A}\ \mathbf{C}), \mathrm{tw}(\mathbf{D}\ \mathbf{E}), k\}$.*

Moreover, if (T_P, B_P) is a tree decomposition of $G(\mathbf{A}\ \mathbf{C})$ of treewidth $\mathrm{tw}(\mathbf{A}\ \mathbf{C})$ with a node d with "columns of $\mathbf{C}$" $\subseteq B_P(d)$ and (T_Q, B_Q) is a tree decomposition of $G(\mathbf{D}\ \mathbf{E})$ of treewidth $\mathrm{tw}(\mathbf{D}\ \mathbf{E})$ containing a node d' with "columns of $\mathbf{D}$" $\supseteq B_Q(d')$, then we can choose $\mathbf{F}$ and $\mathbf{c}$ in such a way that a tree decomposition (T_R, B_R) of $G(\mathbf{F})$ of treewidth $\max\{\mathrm{tw}(\mathbf{A}\ \mathbf{C}), \mathrm{tw}(\mathbf{D}\ \mathbf{E}), k\}$ exists, where T_R is obtained from T_P and T_Q by identifying the nodes d and d' and $B_R = B_P \cup (B_Q \setminus \{B_Q(d')\})$.

Proof. We start by observing that the assumptions and the gluing lemma imply that the inequalities

$$\begin{aligned} \mathbf{Ax}+\mathbf{Cz} &\geq \mathbf{a} \\ \mathbf{Dz}+\mathbf{Ey} &\geq \mathbf{b} \end{aligned}$$

describe an extended formulation of $P \times_k Q$. Consider the treewidth of the matrix $\mathbf{F} = \left(\begin{smallmatrix} \mathbf{A} & \mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} & \mathbf{E} \end{smallmatrix}\right)$. The Gaifman graph $G(\mathbf{F})$ of $\mathbf{F}$ can be obtained by taking $G(\mathbf{A}\ \mathbf{C})$ and $G(\mathbf{D}\ \mathbf{E})$ and identifying the vertices corresponding to the variables $\mathbf{z}$ and $\mathbf{w}$ in the above formulation. It is easy to observe that the treewidth of $G(\mathbf{F})$ is $\max\{\mathrm{tw}(\mathbf{A}\ \mathbf{C}), \mathrm{tw}(\mathbf{D}\ \mathbf{E}), k\}$, as desired, and that if (T_P, B_P) and (T_Q, B_Q) are as assumed, the tuple (T_R, B_R) obtained in the aforementioned way is indeed a tree decomposition of $G(\mathbf{F})$. □

Lemma 5.1.7 (Trivial treewidth bound)**.** *Let $P \subseteq \mathbb{R}^d$ be a polytope with n vertices. Then there exist an extension of P that can be described by inequalities of treewidth at most $n + d$.*

Proof. Consider again the description of the extension of P used in the proof of Proposition 2.6.11:

$$\begin{aligned} \mathbf{1}\boldsymbol{\lambda} &= 1 \\ \mathbf{V}\boldsymbol{\lambda} - \mathbf{I}\mathbf{x} &= \mathbf{0} \\ \boldsymbol{\lambda} &\geq \mathbf{0} \end{aligned}$$

where $\mathbf{0}$ and $\mathbf{1}$ are the all-0 and all-1 vectors of appropriate dimensions, respectively, $\mathbf{I}$ is the identity matrix and $\mathbf{V}$ is a matrix whose i-th column is the i-th vertex of P; each equality is just an abbreviation of two opposing inequalities. Since the number of columns in the system is $n + d$, its treewidth is by definition also at most $n + d$. □

Putting Lemmas 5.1.4 and 5.1.7, together gives the following corollary:

Corollary 5.1.8. *Let $P \subseteq \mathbb{R}^m$ be an integral polytope with n vertices. Then there exists a constructively decomposable extension of P that can be described by inequalities of treewidth at most $n + m$.*

Proof. It suffices to notice that the extended formulations of Lemmas 5.1.4 and 5.1.7 are identical and thus simultaneously have small treewidth and are decomposable. □

5.1.3 MSO Polytope: Take Two

Using our results about the glued product in Section 4.2 we can extend Theorem 4.4.4 to guarantee a couple of additional non-trivial properties of a certain extended formulation of $P_\varphi(G)$. Recall that $\mathcal{C} = \{\alpha_1, \ldots, \alpha_w\}$ is the set of equivalence classes of the relation $\equiv_k^{MSO}$, and that for a given formula φ, a graph G and a tree decomposition (T, B) of G, for every node a of T, we denote the set of feasible types of the node a by $\mathcal{F}(a)$, for every introduce and every forget node a of T the set of feasible pairs of the node a by $\mathcal{F}_p(a)$ and for every join node a of T the set of feasible triples of the node a by $\mathcal{F}_t(a)$. Moreover, let V_{IF} denote the set of introduce and forget nodes in T, V_J the set of join nodes and V_L the set of leaves, and let $\mathcal{F} = \bigcup_{b \in V_{IF}} \{\{b\} \times \mathcal{F}_p(b)\} \cup \bigcup_{b \in V_J} \{\{b\} \times \mathcal{F}_t(b)\} \cup \bigcup_{b \in V_L} \{\{b\} \times \mathcal{F}(b)\}$, that is, $\mathcal{F}$ is a set containing for every node $b \in V(T)$ a pair $\{b, \mathcal{F}'(b)\}$ where $\mathcal{F}'(b)$ is the set of feasible pairs $\mathcal{F}_p(b)$ for introduce and forget nodes, the set of feasible triples $\mathcal{F}_t(b)$ for join nodes and the set of feasible types $\mathcal{F}(b)$ for leaves.

As in the case of Theorem 4.3.1, for the sake of simplicity we again formulate and prove the main theorem of this section in terms of graphs represented as σ_2-structures; the extension to arbitrary σ_2-structures is straightforward.

Theorem 5.1.9. *Let $G = (V, \emptyset, \{E, L_V, L_E\})$ be a σ_2-structure of treewidth τ representing a graph, let $n = |V|$ and let (T, B) be a nice tree decomposition of G of treewidth τ and let φ be an MSO_1 σ_2-formula with m free variables.*

Then there exist matrices $\mathbf{A}, \mathbf{D}, \mathbf{C}$, a vector $\mathbf{e}$, a function $\nu : \mathcal{C} \times V(T) \times V \times [m] \to \{0, 1\}$ and a tree decomposition (T^*, B^*) of the Gaifman graph $G(\mathbf{A}\ \mathbf{D}\ \mathbf{C})$ such that the following claims hold:

1. *The polytope $P = \{(\mathbf{y}, \mathbf{t}, \mathbf{f}) \in \mathbb{R}^{V \times [m]} \times \mathbb{R}^{\mathcal{C} \times V(T)} \times \mathbb{R}^{\mathcal{F}} \mid \mathbf{A}\mathbf{y} + \mathbf{D}\mathbf{t} + \mathbf{C}\mathbf{f} = \mathbf{e}, \quad \mathbf{t}, \mathbf{f} \geq \mathbf{0}\}$ is a 0/1-polytope and $P_\varphi(G) = \{\mathbf{y} \mid \exists \mathbf{t}, \mathbf{f} : (\mathbf{y}, \mathbf{t}, \mathbf{f}) \in P\}$.*

2. *P is constructively decomposable.*

3. *For any $(\mathbf{y}, \mathbf{t}, \mathbf{f}) \in \mathrm{vert}(P)$, for any $j \in \mathcal{C}$, $b \in V(T)$, $v \in B(b)$ and $i \in [m]$, equalities $t_b(j) = 1$ and $\nu(j, b, v, i) = 1$ imply that $y_v^i = 1$.*

4. *(a) The treewidth of (T^*, B^*) is $\mathcal{O}(|\mathcal{C}|^3)$,*
   (b) $T^* = T$,
   (c) for every node $b \in V(T^*)$, $\bigcup_{j \in \mathcal{C}} \{t_b(j)\} \subseteq B^*(b)$, and,
   (d) $\bigcup_{j \in \mathcal{C}} \{t_b(j)\} \cap B^*(a) = \emptyset$ for every $a \notin N_{T^*}(b)$.

5. *$\mathbf{A}, \mathbf{D}, \mathbf{C}, \mathbf{e}, \nu$ can be computed in time $\mathcal{O}(|\mathcal{C}|^3 \cdot n)$.*

Let us first comment on the meaning and usefulness of the various points of the theorem. Point (1) simply states that P is an extended formulation of $P_\varphi(G)$. However, there are variables $\mathbf{t}$ which allow some interpretation of integer points of P as we will discuss further (point (3)), and there are also variables f, which are used to ensure constructive decomposability (point (2)).

Our results in Chapters 6 and 7 can be viewed as applications of this theorem. We shall now preview ideas that we develop further on in this chapter, even though we end up taking a slightly different path. It is possible to view the system of linear inequalities $\mathbf{Ay} + \mathbf{Dt} + \mathbf{Cf} = \mathbf{e}$ with $\mathbf{t}, \mathbf{f} \geq \mathbf{0}$, which defines P, as an ILP, because we are only interested in its integer solutions. Then, it is easy to observe that ILP is a special case of CSP where all constraints are linear. Thus, viewing the ILP as a CSP allows adding nonlinear constraints and optimizing a nonconvex objective function. Finally, by point (4a), this CSP instance has bounded treewidth, and thus can be efficiently solved by Freuder's algorithm (Theorem 2.4.4).

Point (3) is closely related to Lemma 4.3.2 which we needed for the proof of Theorem 4.3.1. Intuitively, it says that we can view the variables $\mathbf{t}$ of integer points of P as an assignment from $V(T)$ to $\mathcal{C}$ (i.e., each node is assigned a type) and that knowing a type of a node b is sufficient for knowing, for each vertex $v \in B(b)$, to which free variables X_i vertex v belongs. This makes it possible to extend the CSP instance corresponding to the system defining P with further constraints modeling various extensions of MSO; see Chapter 6. Point (4c) is crucial here, since it allows us to add new constraints in a way which does not increase the treewidth of the resulting CSP instance by much.

Point (2) essentially says that integer points of rP correspond to r-multisets (i.e., multisets of size r) of vertices of P. We use this in Chapter 7, where connect separable optimization over the r-dilate of a decomposable 0/1 polyhedron Q to shifted combinatorial optimization. Thus, when $S = S_\varphi(G)$ is the set of satisfying assignments of a formula φ on a graph G, one can optimize over integer points of rP in order to optimize over r-multisubsets of $S_\varphi(G)$.

Proof of Theorem 5.1.9. Let us give an outline of the proof first. The construction of the polytope P, and of the corresponding system of linear inequalities describing it, is done in three phases. In each phase we construct and examine three related objects: a certain polytope, a system of linear inequalities defining it, and a tree decomposition of the Gaifman graph of the system of linear inequalities. We closely follow along the lines of the proof of Theorem 4.3.1 but we

modify and extend it in a way that will make it possible to prove the additional properties. In the first phase, we construct a polytope Q'_c, an analogue of the polytope Q_c from the aforementioned proof. The vertices of this polytope correspond to assignments of feasible types to the nodes of the tree decomposition T. In the second phase, we define a polytope Q'_φ as a properly chosen face of the polytope Q'_c, analogously to the choice of the face Q_φ of the polytope Q_c. The third phase consists only of introducing the variables y^i_v as a suitable linear combination of z – this way we obtain the polytope P from the polytope Q'_φ.

Phase 1: Constructing Q'_c. In the proof of Theorem 4.3.1, we obtain the polytope Q_c by gluing together polytopes P_b, $b \in V(T)$, in a bottom-up fashion over nodes of a nice tree decomposition of G. Recall that every P_b is a 0/1-polytope, has dimension at most $3|\mathcal{C}|$ and its number of vertices is at most $|\mathcal{C}|^3$. Thus, by Corollary 5.1.8, there exists a constructively decomposable extension P'_b of P_b describable by inequalities of treewidth at most $|\mathcal{C}|^3 + 3|\mathcal{C}| = \mathcal{O}(|\mathcal{C}|^3)$.

We proceed in the same way as in the construction in the proof of Theorem 4.3.1 but instead of P_b, we glue together the polytopes P'_b. At the same time, by Lemma 5.1.6 we combine, again in the bottom-up fashion over nodes of the nice tree decomposition (T, B) of G, the systems of inequalities that describe the polytopes P'_b and also the tree decompositions of the corresponding Gaifman graphs. Let c denote the root of the decomposition tree T as in the proof of Theorem 4.3.1, let $\mathbf{D't} + \mathbf{C'f} = \mathbf{e'}$ with $\mathbf{t}, \mathbf{f} \geq \mathbf{0}$ denote the resulting system of inequalities describing the polytope Q'_c and let (T', B') denote the resulting tree decomposition of the Gaifman graph $G(\mathbf{D'}\ \mathbf{C'})$.

We shall now prove that the conditions (4) hold for (T', B') by induction. Then, it will be sufficient in the later stages of the proof to show that they will not be violated.

Lemma 5.1.10. *For each $b \in V(T)$, there are matrices $\mathbf{C}'_b$ and $\mathbf{D}'_b$ such that $\mathbf{C}'\mathbf{t}_b + \mathbf{D}'_b(\bar{\mathbf{t}}, \bar{\mathbf{f}}) = \mathbf{e}'_b$ with $\mathbf{t}_b, \bar{\mathbf{t}}, \bar{\mathbf{f}} \geq 0$ describes the intermediate polytope Q'_b obtained in the bottom-up construction, $G(\mathbf{C}'_b\ \mathbf{D}'_b)$ has a tree decomposition (T_b, B_b) of treewidth $\mathcal{O}(|\mathcal{C}|^3)$, T_b is as defined before (i.e., a subtree of T rooted in b), and for every node $a \in V(T_b)$, it holds that $\bigcup_{i\in\mathcal{C}}\{t_a(i)\} \subseteq B_b(a)$, and $\bigcup_{i\in\mathcal{C}}\{t_a(i)\} \cap B_b(a') = \emptyset$ for every $a' \notin N_{T_a}(a)$.*

Clearly, if these conditions hold for the root c, the conditions (4) are satisfied.

Proof. First, let b be a leaf. By Lemma 5.1.7, there is a system $\mathbf{C}'_b\mathbf{t}_b + \mathbf{D}'_b\mathbf{f}_b = \mathbf{e}_b$ with $\mathbf{f}_b \geq \mathbf{0}$ describing P'_b (which contains just one point), and there is a trivial tree decomposition of $G(\mathbf{C}'_b\ \mathbf{D}'_b)$ with one bag containing all vertices. This establishes the base case of the induction.

Consider an introduce or forget node b of T with a child a, and we glue the polytopes Q'_a and P'_b. By Lemma 5.1.7, P'_b is described by $\mathbf{A}_b\mathbf{d}_b + \mathbf{C}_b\mathbf{t}_b + \mathbf{D}_b\mathbf{f}_b = \mathbf{e}_b$ with $\mathbf{f}_b \geq \mathbf{0}$, and $G(\mathbf{A}_b\ \mathbf{C}_b\ \mathbf{D}_b)$ has a trivial tree decomposition with one bag containing all its vertices. By the induction hypothesis, Q'_a is described by $\mathbf{C}'_a\mathbf{t}_a + \mathbf{D}'_a(\bar{\mathbf{t}}, \bar{\mathbf{f}}) = \mathbf{e}_a$ with $\mathbf{t}_a, \bar{\mathbf{t}}, \bar{\mathbf{f}} \geq \mathbf{0}$, where $\bar{\mathbf{t}}$ and $\bar{\mathbf{f}}$ are the $\mathbf{t}$ and $\mathbf{f}$ variables associated with the nodes of T_a. Thus, the following system, where the columns $\begin{smallmatrix}\mathbf{A}_b\\ \mathbf{C}'_a\end{smallmatrix}$ correspond to the matrix $\mathbf{C}'_b$ and the remaining columns to the matrix $\mathbf{D}'_b$, describes Q'_b:

$$\begin{aligned} \mathbf{A}_b\mathbf{d}_b + \mathbf{C}_b\mathbf{t}_b + \mathbf{D}_b\mathbf{f}_b & \geq \mathbf{e}_b \\ \mathbf{C}'_a\mathbf{t}_a + \mathbf{D}'_a(\bar{\mathbf{t}}, \bar{\mathbf{f}}) & \geq \mathbf{e}_a \end{aligned}$$

Moreover, because there is a tree decomposition (T_a, B_a) of the Gaifman graph $G(\mathbf{A}_a\ \mathbf{E}_a)$ such that $\bigcup_{i\in\mathcal{C}}\{t_a(i)\} \subseteq B_a(a)$, we are in the situation of the second part of Lemma 5.1.6 with $\mathbf{A} = (\mathbf{A}_b\ \mathbf{D}_b)$, $\mathbf{C} = \mathbf{C}_b$, $\mathbf{D} = \mathbf{C}'_a$, $\mathbf{E} = \mathbf{D}'_a$. This implies that (T_b, B_b) with B_b defined by $B_b(a') = B_a(a')$ for all $a' \in T_a$, and $B_b(b) = V(G(\mathbf{A}_b\ \mathbf{C}_b\ \mathbf{D}_b))$, is a tree decomposition of $G(C'\ D')$ which has the desired properties.

The situation is analogous for the join node. By the first part of Lemma 5.1.6 together with the induction hypothesis, the treewidth of (T_b, B_b) is clearly at most $\mathcal{O}(|\mathcal{C}|^3)$. □

Phase 2: Taking the face Q'_φ. We take the face Q'_φ of Q'_c corresponding to the valid inequality $\sum_{j\in\mathcal{C}} \rho_\varphi(k)\cdot t_c(j) \leq 1$. That corresponds to adding the equality $\sum_{j\in\mathcal{C}} \rho_\varphi(j)\cdot t_c(j) = 1$ to the system $\mathbf{D}'\mathbf{t} + \mathbf{C}'\mathbf{f} = \mathbf{e}'$. Let us denote $\mathbf{D}''\mathbf{t} + \mathbf{C}''\mathbf{f} = \mathbf{e}''$ the system obtained from $\mathbf{D}'\mathbf{t} + \mathbf{C}'\mathbf{f} = \mathbf{e}'$ by adding the aforementioned equality. Adding the new equality corresponds to adding edges to $G(\mathbf{D}'\ \mathbf{C}')$ which are connecting vertices $\mathbf{t}_c(j)$. Since all variables $t_c(j)$ belong to the bag $B'(c)$, the tree decomposition conditions are not violated by adding these edges and (T', B') is a tree decomposition of $G(\mathbf{D}''\ \mathbf{C}'')$ as well. Thus, the treewidth of $G(\mathbf{D}''\ \mathbf{C}'')$ is the same as the treewidth of $G(\mathbf{D}'\ \mathbf{C}')$.

We will now show that since Q'_c is constructively decomposable and a 0/1-polytope, Q'_φ is constructively decomposable as well. Let $r \in \mathbb{N}$ and consider any $\mathbf{x} \in rQ'_\varphi$. Because $\mathbf{x} \in rQ'_\varphi \subseteq rQ'_c$, there exist $\mathbf{x}^1, \ldots, \mathbf{x}^r \in Q'_c$ such that $\sum_i \mathbf{x}^i = \mathbf{x}$. Because $\mathbf{x} \in rQ'_\varphi$, it satisfies $\sum_{j\in\mathcal{C}} \rho_\varphi(j)\cdot t_c(j) = r$, and because Q'_φ is a 0/1-polytope, every $\mathbf{x}^i$ satisfies $\sum_{j\in\mathcal{C}} \rho_\varphi(t)\cdot t_c(j) = 1$. This implies that $\mathbf{x}^i \in Q'_\varphi$ for all $i \in [r]$.

Phase 3: Obtaining P by adding variables y. To obtain P from Q'_φ, it remains to add projections $y^i_v = \sum_{j\in\mathcal{C}} \mu(j, v, i)\cdot t_{\mathrm{top}(v)}(j)$ for each $v \in V$ and $i \in [m]$. Now consider the system $\mathbf{A}\mathbf{y} + \mathbf{D}\mathbf{t} + \mathbf{C}\mathbf{f} = \mathbf{e}$ which is thus obtained.

Regarding the treewidth of $G(\mathbf{A}\ \mathbf{D}\ \mathbf{C})$, note that the sum defining each y^i_v only involves variables associated with the node $\mathrm{top}(v)$. Specifically, $\mathbf{A}\mathbf{y}+\mathbf{D}\mathbf{t}+\mathbf{C}\mathbf{f} = \mathbf{e}$ can be written as

$$\begin{aligned} \mathbf{0}\mathbf{y} + \mathbf{D}''\mathbf{t} + \mathbf{C}''\mathbf{f} &= \mathbf{e}'' \\ -\mathbf{I}\mathbf{y} + \mathbf{\Lambda}\ \mathbf{t} + \mathbf{0}\ \ \mathbf{f} &= \mathbf{0} \end{aligned}$$

where the block $(-\mathbf{I}\ \mathbf{\Lambda}\ \mathbf{0})$ corresponds to the projections to y^i_v.

Fix $v \in V$. Then in $G(\mathbf{A}\ \mathbf{D}\ \mathbf{C})$ the variable y^i_v corresponds to a vertex connected to vertices $t_{\mathrm{top}(v)}(j)$ for which $\mu(j, v, i) = 1$, all of which belong to one bag $B = B'(\mathrm{top}(v))$. A decomposition (T^*, B^*) of $G(\mathbf{A}\ \mathbf{D}\ \mathbf{C})$ can be obtained from (T', B') by adding, for every $i \in [m]$, the vertex corresponding to y^i_v to the bag B. This increases the width of B by at most $\mathbf{m}$. Since $\mathrm{top}(v)$ is distinct for every $v \in V$, this operation can be performed independently for every v, resulting

in a decomposition of width at most $\mathcal{O}(|\mathcal{C}|^3)+m = \mathcal{O}(|\mathcal{C}|^3)$, satisfying the claimed properties (4).

Regarding constructive decomposability, we use Lemma 5.1.3. The polytope Q'_φ is constructively decomposable, and there is a linear projection π with integer coefficients such that $P_\varphi(G) = \pi(Q'_\varphi)$. Thus $P = \{(\pi(\mathbf{t}), \mathbf{t}, \mathbf{f}) \mid (\mathbf{t}, \mathbf{f}) \in Q'_\varphi\}$ is constructively decomposable, satisfying property (2).

Finally, Theorem 4.4.4 shows that $\mathbf{A}, \mathbf{D}, \mathbf{C}, \mathbf{e}$ and ν can be constructed in the claimed time, satisfying property (5), and by the definition of ν and Lemma 4.3.2, condition (3) is also satisfied, completing the proof. □

As a corollary, we have the following.

Corollary 5.1.11 (Decomposable extension complexity of the MSO polytope)**.** *Under the assumptions of Theorem 4.3.1,* $\mathrm{xc}_{dec}(P_\varphi(G)) \leq f(|\varphi|, \tau) \cdot n$.

5.2 Combining MSO and CSP

In this section we shall connect Courcelle's Theorem and CSP. First, we will use our theory of the glued product to give an alternative proof of Theorem 3.0.1. Using the glued product immediatelly provides some additional properties, like we saw in Theorem 5.1.9. Moreover, we will show that a strong optimization oracle is realizable for the r-dilate of the CSP polytope. Afterwards, we will show that Courcelle's theorem can actually be cast as an instance of CSP. This implies that Theorem 4.3.1 follows from Theorem 3.0.1. Since by then we will have a much stronger theorem for CSPs, we will obtain a stronger theorem also for MSO-definable sets.

5.2.1 CSP Polytope via Glued Product

Definition 5.2.1 (Separable function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is* (additively) separable *if there exist functions $f_1, \ldots, f_n$ such that $f(\mathbf{x}) = \sum_{i=1}^n f_i(x_i)$.*

Definition 5.2.2 (Integer separable (convex) minimization oracle)**.** *Let $P \subseteq \mathbb{R}^n$. An* integer separable minimization oracle *for P is one that, queried on a separable function f, either reports that $P \cap \mathbb{Z}^n$ is empty, or that it is unbounded, or returns a point $\mathbf{x} \in P \cap \mathbb{Z}^n$ which minimizes $f(\mathbf{x})$.*

An integer separable convex minimization oracle *for P is an integer separable minimization oracle for P which can only be queried on functions f as above with all f_i convex.*

Then the main result of this subsection is the following.

Theorem 5.2.3 (Master CSP theorem)**.** *Let $I = (V, \mathcal{D}, \mathcal{H}, \mathcal{S})$ be a CSP instance of treewidth τ and let (T, B) be a nice tree decomposition of $G(I)$ of width τ. Then there exist matrices $\mathbf{A}, \mathbf{C}, \mathbf{D}$, a vector $\mathbf{e}$ and a tree decomposition (T^*, B^*) of the Gaifman graph $G(\mathbf{A}\ \mathbf{C}\ \mathbf{D})$ such that the following claims hold:*

1. *The polytope $P = \{(\mathbf{y}, \mathbf{f}, \mathbf{h}) \mid \mathbf{Ay} + \mathbf{Cf} + \mathbf{Dh} = \mathbf{e}, \mathbf{f}, \mathbf{h} \geq \mathbf{0}\} \subseteq \mathbb{R}^{\mathcal{O}(nD+D^{2\tau})}$ is a 0/1-polytope and $CSP(I) = \{\mathbf{y} \mid \exists(\mathbf{f}, \mathbf{h}) : (\mathbf{y}, \mathbf{f}, \mathbf{h}) \in P\}$, i.e., P is an extension of $CSP(I)$.*

2. *P is constructively decomposable.*

3. *The treewidth of (T^*, B^*) is at most $D^{2\tau}$ and $T = T^*$.*

4. *$\mathbf{A}, \mathbf{C}, \mathbf{D}, \mathbf{e}$ and (T^*, B^*) can be computed in time $D^{\mathcal{O}(\tau)}n$.*

5. *An integer separable minimization oracle for rP is realizable in time $\min(D^{\mathcal{O}(r\tau)}, r^{D^{\mathcal{O}(\tau)}}) \cdot |V| + \|\mathcal{H}\| + \langle f \rangle$, where $\langle f \rangle$ is the encoding length of f.*

Proof. **Points 1–4.** The idea of our proof is the same as in Theorem 5.1.9, except that we replace the sets of feasible types and pairs or triples of types with appropriate sets of configurations. Then we verify that all our arguments go through as expected. More precisely:

- for each node $a \in V(T)$, we replace the set of feasible types $\mathcal{F}(a)$ with the set of configurations of the bag $B(a)$, that is, $\mathcal{F}^{CSP}(a) = \mathcal{K}(B(a))$,
- for each introduce node $b \in V(T)$ with a child $a \in V(T)$, we replace the set of feasible pairs $\mathcal{F}_p(c)$ with the set $\mathcal{F}_p^{CSP}(c) = \{(\mathbf{k}, \mathbf{k}') \in \mathcal{K}(B(b)) \times \mathcal{K}(B(a)) \mid \mathbf{k}\restriction_{B(a)} = \mathbf{k}'\}$, and symmetrically for a forget node b with the only difference that $\mathbf{k}'\restriction_{B(b)} = \mathbf{k}'$, and,
- for each join node $c \in V(T)$ with children $a, b \in V(T)$, we replace the set of feasible triples $\mathcal{F}_t(c)$ with the set $\mathcal{F}_t^{CSP}(c) = \{(\mathbf{k}, \mathbf{k}, \mathbf{k}) \mid \mathbf{k} \in \mathcal{K}(B(c))\}$.

All our arguments from Phase 1 of the proof of Theorem 5.1.9 carry over and thus we obtain a polytope Q'_c which is constructively decomposable and there is a system $\mathbf{C}'\mathbf{f} + \mathbf{D}'\mathbf{h} = \mathbf{e}'$, $\mathbf{f}, \mathbf{h} \geq \mathbf{0}$ defining it such that $\mathrm{tw}(\mathbf{C}'\ \mathbf{D}') \leq D^{2\tau}$ by the fact that $|\mathcal{K}(B(b))|^2 = |\mathcal{F}^{CSP}(b)|^2 \leq |\mathcal{F}_p^{CSP}(b)| \leq D^{2\tau}$ for each $b \in V(T)$; note that for join nodes, $|\mathcal{F}^{CSP}(b)| = |\mathcal{F}_t^{CSP}(b)|$. Phase 2 does not exist in the case of CSP. Finally, in Phase 3 (obtaining variables y_v^i) we have argued that the treewidth is not significantly increased by adding these variables in a suitable way and that the resulting polytope is still constructively decomposable. These arguments also carry over completely by replacing the indicator function $\mu(\beta, v, i)$ with an analogous function $\mu'(\mathbf{k}, v, i)$ defined for each variable $v \in V$, each configuration $\mathbf{k} \in \mathcal{K}(B(\mathrm{top}(v))$ and each $i \in D(v)$ as $\mu'(\mathbf{k}, v, i) = 1$ if $\mathbf{k}|_{\{v\}} = i$ and 0 otherwise.

Point 5. The following exposition is heavy on notation. However, there is an easier way to obtain only a slightly weaker version of Point 5 which is sufficient for all of our purposes. We provide it after we are finished with this proof.

We will set up an auxiliary CSP instance J to realize an integer separable minimization oracle for rP. Let us first review the meaning of the variables $\mathbf{f}$ and $\mathbf{h}$, as the CSP instance J will follow it closely. By the construction above, for each node b and each configuration $\mathbf{k} \in \mathcal{K}(B(b)) = \mathcal{F}^{CSP}(b)$, there is a binary variable $f_b[\mathbf{k}]$ indicating whether b has configuration $\mathbf{k}$ in the solution. Moreover, for each introduce or forget node $b \in V(T)$ with a child $a \in V(T)$ and each $(\mathbf{k}, \mathbf{k}') \in \mathcal{F}_p^{CSP}(b)$, there is a binary variable $h_b[(\mathbf{k}, \mathbf{k}')]$ indicating that a has configuration $\mathbf{k}'$ and b has configuration $\mathbf{k}$; analogously for a join node $c \in V(T)$ with children $a, b \in V(T)$ and a variable $h_c[(\mathbf{k}, \mathbf{k}, \mathbf{k})]$.

In the following exposition we use multisets, so let us review a few relevant notions first. Let X be a set of size n and k be an integer. Then the number

of multisubsets of X of size k is precisely $\binom{n+k-1}{k}$, which we denote $\left(\!\binom{n}{k}\!\right)$ ("n multichoose k"). It is common to denote by $\binom{X}{k}$ the set of all subsets of X of size k. We extend this notation to multisubsets and let $\left(\!\binom{X}{k}\!\right)$ be the set of all multisubsets of X of size k. For a multiset A and its element a, we denote by $\mathrm{mult}(a, A)$ the multiplicity of a in A.

Let us distinguish the components of two CSP instances I and J with subscripts, e.g. the variables of I are V_I and the variables of J are V_J. Because the variables of J are analogous to the variables of rP, we shall denote them with capital letters $\mathbf{Y}, \mathbf{F}$ and $\mathbf{H}$.

- For each $v \in V_I$ and each $i \in D(v)$ we have a variable Y_v^i with domain $[r]$; this variable plays exactly the same role as the variable y_v^i.
- For each $b \in V(T)$ we have a variable F_b with domain $\left(\!\binom{\mathcal{K}(B(b))}{r}\!\right)$; this variable plays the role of the variables $\mathbf{f}_b$, the only difference is that $\mathbf{f}_b$ is essentially a unary encoding of F_b.
- For each node $c \in V(T)$ with a child $a \in V(T)$ (i.e., for an introduce or forget node) or children $a, b \in V(T)$ (i.e., for a join node), we have a variable H_c with domain $\left(\!\binom{\mathcal{F}_p^{CSP}(b)}{r}\!\right)$ or $\left(\!\binom{\mathcal{F}_t^{CSP}(b)}{r}\!\right)$ for introduce and forget nodes, or join nodes, respectively. Again, the variables $\mathbf{h}_c$ are essentially a unary encoding of H_c. Observe that the elements of $\left(\!\binom{\mathcal{F}_p^{CSP}(b)}{r}\!\right)$ and $\left(\!\binom{\mathcal{F}_t^{CSP}(b)}{r}\!\right)$ are multisets of pairs or triples. For a multiset of tuples K, we abuse notation and denote $K[i] = \{\mathbf{k}[i] \mid \mathbf{k} \in K\}$; e.g., for $K \in \left(\!\binom{\mathcal{F}_p^{CSP}(b)}{r}\!\right)$, $K[1]$ is the multiset of first coordinates of elements of K.

The hard constraints $\mathcal{H}_J$ are as follows.

- For each node $c \in V(T)$ with a child $a \in V(T)$ or children $a, b \in V(T)$, we have the following hard constraints:
  - $\{(F_c, H_c) \mid H_c[1] = F_c\}$ and $\{(F_a, H_c) \mid H_c[2] = F_a\}$ if a is the only child of c, and,
  - $\{(F_c, H_c) \mid H_c[1] = F_c\}$, $\{(F_a, H_c) \mid H_c[2] = F_a\}$ and $\{(F_b, H_c) \mid H_c[3] = F_b\}$, if a, b are the two children of b.
- For each Y_v^i, we have a hard constraint $\{(F_{\mathrm{top}(v)}, Y_v^i) \mid F_{\mathrm{top}(v)} \in \mathcal{K}(\mathrm{top}(v)), Y_v^i = M(F_{\mathrm{top}(v)}, v, i)\}$, where $M(K, v, i)$ is defined for all $K \in \left(\!\binom{\mathcal{F}^{CSP}(\mathrm{top}(v))}{r}\!\right)$ as the cardinality of the multiset $\{\mathbf{k} \in K \mid \mu'(\mathbf{k}, v, i) = 1\}$.

Consider now the given separable function f. We define the following weighted soft constraints $\mathcal{C}_J$ based on f.

- For each variable y_v^i, let $w_{Y_v^i}(Y_v^i) = f_{y_v^i}(Y_v^i)$.
- For each node $b \in V(T)$, let $w_{F_b}(F_b) = \sum_{j=0}^{r} \sum_{\mathbf{k} \in F_b : \mathrm{mult}(\mathbf{k}, F_b) = j} f_{f_b[\mathbf{k}]}(j)$.
- For each node $b \in V(T)$, let $w_{H_b}(H_b) = \sum_{j=0}^{r} \sum_{K \in H_b : \mathrm{mult}(K, H_b) = j} f_{h_b[K]}(j)$.

Then, we use Freuder's algorithm (Theorem 2.4.4) to find a minimal solution to J. It is easy to find a point of rP based on this solution as follows. Let:

- $y_v^i = Y_v^i$,
- $f_b[\mathbf{k}] = \text{mult}(\mathbf{k}, F_b)$, and
- $h_b[K] = \text{mult}(K, H_b)$, where K is a pair or a triple of configurations when b is a introduce and forget or join node, respectively.

Regarding the time complexity, observe that J has treewidth 1 because $G(J)$ is a tree by a similar argument as we have used in the alternative proof of Theorem 4.3.1. Moreover, the size of its domains is upper bounded by the maximum size of $\left(\!\!\binom{\mathcal{F}_p^{CSP}(b)}{r}\!\!\right)$ over all nodes $b \in V(T)$. By standard identities, $\left(\!\!\binom{n}{k}\!\!\right) = \binom{n+k-1}{k} = \binom{n+k-1}{n-1} = \left(\!\!\binom{k+1}{n-1}\!\!\right)$. Thus, by the bound $\binom{n}{k} \leq n^k$, we have that $\left(\!\!\binom{n}{k}\!\!\right) \leq \min(n^k, k^n)$. This implies that the maximum domain size of J satisfies $D_J \leq \min(D_I^{r\tau}, r^{D_I^\tau})$; similarly we have that $\|\mathcal{H}_J\| \leq \min(D_I^{2r\tau}, r^{D_I^{2\tau}})$. □

The polytope P of the last theorem is derived not just from the CSP instance I, but also the nice tree decompostition (T, B) of $G(I)$. Nevertheless, with slight abuse of notation, we shall denote it from now on as $P(I)$.

Viewing ILP as CSP. Note that we could have obtained a slightly weaker version of Point 5 of Theorem 5.2.3 by a different argument. Observe that we have used the bound $\left(\!\!\binom{n}{k}\!\!\right) \leq \min(n^k, k^n)$ to prove Point 5. In situations when bounding by n^k suffices, we can do the following. Since the polytope $rP(I)$ is described by an LP of small treewidth, and we are only interested in its integer solutions, we can view it as an ILP of small treewidth. Moreover, it is easy to see that an ILP is a special case of CSP where all constraints are linear. Let us make this argument explicit.

Definition 5.2.4 (CSP-ILP equivalence)**.** *A CSP instance I is equivalent to an ILP $\mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \in \mathbb{Z}^n$ if*

$$\{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} \leq \mathbf{b}\} = \text{Feas}(I).$$

Lemma 5.2.5. *Let $\mathbf{A} \in \mathbb{Z}^{n \times m}, \mathbf{b} \in \mathbb{Z}^m, \boldsymbol{\ell}, \mathbf{u} \in \mathbb{Z}^n$ be given s.t. $tw(\mathbf{A}) = \tau$, and let $D = \|\mathbf{u} - \boldsymbol{\ell}\|_\infty$. Then an integer separable minimization oracle over $P = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} = \mathbf{b}, \boldsymbol{\ell} \leq \mathbf{x} \leq \mathbf{u}\}$ is realizable in time $D^\tau(n+m) + \langle f \rangle$.*

Proof. The proof proceeds as follows. First, we construct a CSP instance I equivalent to the ILP $\mathbf{Ax} = \mathbf{b}, \boldsymbol{\ell} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^n$ such that their Gaifman graphs $G(\mathbf{A})$ and $G(I)$ are equal, and thus $\text{tw}(\mathbf{A}) = \text{tw}(I)$. Moreover, the maximum domain size of I is D. Second, we use the fact that an integer separable function is easily encoded in a CSP instance by weighted soft constraints. Then, Theorem 2.4.4 does the job.

We now construct I. Let $V = \{x_1, \ldots, x_n\}$. For every $i \in [n]$, let $D_i = [\ell_i, u_i]$ and $\mathcal{D} = \{D_i \mid i \in [n]\}$. Observe that $\max_i |D_i| = \|\mathbf{u} - \boldsymbol{\ell}\|_\infty = D$. Regarding hard constraints $\mathcal{H}$, observe that every row $\mathbf{A}_j$ of $\mathbf{A}$ contains at most $\tau + 1$ nonzeros, since otherwise the Gaifman graph of $\mathbf{A}$ would contain a clique of size $\tau + 2$, contradicting its treewidth of τ. Let $U_j = \{i_1, \ldots, i_k\}$, where $k \leq \tau + 1$, be the set of indices of non-zero elements of $\mathbf{A}_j$, and let $x_c = 0$ for all $c \notin U_j$. Let C_{U_j} be the set of assignments from $D_{i_1} \times \cdots \times D_{i_k}$ to $x_{i_1}, \ldots, x_{i_k}$ that satisfy

$\mathbf{A}_j\mathbf{x} = b_j$; obviously $|C_{U_j}| \leq D^k$ and it can be constructed in time $\mathcal{O}(D^k)$. Let $\mathcal{H} = \{C_{U_j} \mid j \in [m]\}$. Finally, for a given separable function f such that $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$, let $\mathrm{SS} = \{w_{\{x_i\}} \mid i \in [n]\}$ where $w_{\{x_i\}} = f_i$ for all i.

It is easy to verify that the feasible assignments of I correspond to integer solutions of $\mathbf{Ax} = \mathbf{b}, \boldsymbol{\ell} \leq \mathbf{x} \leq \mathbf{u}$, that its maximum domain size is D and its weighted soft constraints SS sum up to exactly f. Finally, the treewidth of I is τ, since the Gaifman graph of $G(I)$ of I is exactly $G(\mathbf{A})$. Then Theorem 2.4.4 solves I in time $\mathcal{O}(D^\tau(n + |\mathcal{H}| + |\mathcal{C}|) = \mathcal{O}(D^\tau(n + m + \langle f \rangle))$, concluding the proof. □

Returning to Theorem 5.2.3, notice that if a polytope P is represented by an LP $\mathbf{Ax} \leq \mathbf{b}$, it is easy to see that rP is represented by $\mathbf{Ax} \leq r\mathbf{b}$. Applying Lemma 5.2.5 to the ILP $\mathbf{Ax} \leq r\mathbf{b}$ and the objective function f does the job.

5.2.2 Courcelle's Theorem as CSP

Now we will show that it is possible to view Theorem 4.3.1 about the extension complexity of the MSO polytope as a consequence of Theorem 3.0.1 about the extension complexity of the CSP polytope. Because we will build on this work further, let us first introduce the notion of a *CSP extension*, modeled after the terminology of polytopes and their extended formulations. Recall (Subsection 2.6) that for a vector $\mathbf{x}$ and a subset of its coordinates I, $\mathbf{x}|_I$ is the $|I|$-dimensional projection of $\mathbf{x}$ to the coordinates I, which is the subvector of $\mathbf{x}$ specified by the coordinates I.

Definition 5.2.6 (CSP extension)**.** *Let $I = (V_I, \mathcal{D}_I, \mathcal{H}_I, \mathcal{S}_I)$ be a CSP instance. We say that $J = (V_J, \mathcal{D}_J, \mathcal{H}_J, \mathcal{S}_J)$ with $V_I \subseteq V_J$ is an* extension of I *(or that J* extends *I) if* $\mathrm{Feas}(I) = \left\{\mathbf{z}|_{V_I} \;\middle|\; \mathbf{z} \in \mathrm{Feas}(J)\right\}$.

Our motivation for introducing CSP extensions is the following. A typical task we face in the following chapters is that we are interested in a certain set S of feasible solutions and our goal is to optimize over S or its r-dilate $\left(\!\binom{S}{r}\!\right)$. Then, our approach is this:

1. We formulate a CSP instance I, possibly of large size and treewidth, such that $S = \mathrm{Feas}(I)$.

2. We apply Theorem 5.2.13 which implies the existence of an extension J of I with bounded treewidth and domain sizes, and with an integer separable minimization oracle for $rCSP(J)$.

3. By the fact that $\mathrm{Feas}(I) = \{\mathbf{z}|_{V_I} \mid \mathbf{z} \in \mathrm{Feas}(J)\}$, we can optimize over $\mathrm{Feas}(I)$ by optimizing over $\mathrm{Feas}(J)$. Analogously, since the integer points of $CSP(J)$ correspond to the elements of $\mathrm{Feas}(J)$, we can optimize over $\left(\!\binom{\mathrm{Feas}(I)}{r}\!\right)$ by optimizing over the decomposable integer points of $rCSP(J)$.

Let us sketch our plan for the rest of this section. Definition 5.2.7 introduces the CSP instance $CSP_\varphi(G)$ with the property that $\mathrm{Feas}(CSP_\varphi(G)) = S_\varphi(G)$. Then, Lemma 5.2.8 shows that $CSP_\varphi(G)$ has an extension of treewidth 2 and with domain sizes bounded by a function of $\mathrm{tw}(G)$ and $|\varphi|$. Definition 5.2.9 introduces the local scope property which is used in Lemma 5.2.10. There, we show that if a CSP instace I with $\mathrm{Feas}(I) \subseteq S_\varphi(G)$ can be described as $CSP_\varphi(G)$

with additional variables and constraints satisfying the local scope property, then I has an extension J with good bounds on its treewidth and domain sizes. Finally, Theorem 5.2.13 applies Theorem 5.2.3 to the extension J of Lemma 5.2.10.

Definition 5.2.7 ($CSP_\varphi(G)$ instance)**.** *Let G with $\mathrm{tw}(G) = \tau$ be a σ_2-structure representing a graph and let φ be an MSO_1 σ_2-formula with m free variables. By $CSP_\varphi(G)$ we denote the CSP instance $(V, \mathcal{D}, \mathcal{H}, \emptyset)$ with $V = \{y_v^i \mid v \in V(G), i \in [m]\}$, all domains equal $\{0,1\}$, and with a hard constraint $\{\mathbf{y} \mid G, \mathbf{y} \models \varphi\}$.*

Observe that $\mathrm{Feas}(CSP_\varphi(G)) = S_\varphi(G)$.

Lemma 5.2.8 (Extension of $\mathrm{CSP}_\varphi(G)$)**.** *Let G with $\mathrm{tw}(G) = \tau$ be a σ_2-structure representing a graph, let (T, B) be a nice tree decomposition of G of treewidth τ, and let φ be an MSO_1 σ_2-formula with m free variables. Then $\mathrm{CSP}_\varphi(G)$ has a CSP extension J with $\mathrm{tw}(J) = 2$, $G(J)$ has a tree decomposition (T, B^*) of width $2 + m$, $D_J = \mathcal{O}(|\mathcal{C}|)$ and $\|\mathcal{H}_J\| = \mathcal{O}(|\mathcal{C}|^3 \cdot n)$.*

Proof. Let us first give a brief clarifying comment on the seemingly mismatching treewidths of J and its tree decomposition (T, B^*). As we shall show further, it is indeed not difficult to obtain a tree decomposition (T', B') of J of width 2. However, we no longer have $T = T'$ as in the statement of the lemma, and it is not obvious that a tree decomposition (T', B') of width 2 with $T' = T$ even exists. Moreover, having the tree decomposition (T, B^*) will make the following expositions cleaner, and the additional "$+m$" factor does not matter asymptotically.

Let (T, B) be a nice decomposition of G of treewidth τ. Consider the following CSP instance $J = (V_J, \mathcal{D}_J, \mathcal{H}_J, \emptyset)$ derived from G, (T, B) and φ. We have these variables in V_J:

- For each vertex $v \in V(G)$ and each $i \in [m]$ a variable y_v^i with domain $\{0,1\}$,
- for each node $a \in V(T)$ except the root a variable t_a with domain $\mathcal{F}(a)$, and,
- for the root node $c \in V(T)$ a variable t_c with domain $\{\beta \in \mathcal{F}(c) \mid \rho_\varphi(\beta) = 1\}$.

We have the following hard constraints in $\mathcal{H}_J$:

- For each introduce or forget node $b \in V(T)$ with child $a \in V(T)$ a constraint $\{(t_b, t_a) \in \mathcal{F}_p(b)\}$.
- For each join node $c \in V(T)$ with children $a, b \in V(T)$ a constraint $\{(t_c, t_a, t_b) \in \mathcal{F}_t(c)\}$.
- For each vertex $v \in V(G)$ and each $i \in [m]$, a constraint $\Big\{(t_{\mathrm{top}(v)}, y_v^i)) \in \{(\beta, \mu(\beta, v, i) \mid \beta \in \mathcal{F}(\mathrm{top}(v))\}\Big\}$.

It is not difficult to observe that the variables t_a encode the feasible type of each node a and thus the feasible assignments of variables y_v^i exactly correspond to the satisfying assignments of $\varphi(X_1, \ldots, X_m)$ in G.

Let us turn our attention to the treewidth of $G(J)$. Its vertices are the variables t_a and y_v^i. First, consider the variables $\mathbf{t}$. The constraints imposed because of introduce and forget nodes only contain edges of T, and the constraints imposed because of join nodes only include t_c and its children t_a, t_b. For each node $b \in V(T)$, let $B'(b) = \{t_a \mid a \in N_T(b)\}$. It is easy to check that (T, B') is a tree decomposition of $G(J)[\{t_a \mid a \in V(T)\}]$, and that it has treewidth 2[1]. Turning our attention to the variables $\mathbf{y}$, we see that each variable y_v^i is connected to one variable $t_{\mathrm{top}(v)}$ and the variables $\mathbf{y}$ are not connected among themselves. Thus, for each $v \in V(G)$, we let $B^*(\mathrm{top}(v)) = B'(\mathrm{top}(v)) \cup \{y_v^i \mid i \in [m]\}$, implying that (T, B^*) has treewidth $2 + m$. However, it is not difficult to see that $G(J)$ itself has treewidth just 2: it is possible to "stretch" the node $\mathrm{top}(v)$ into m copies and add a different y_v^i into the bag of every copy. Note that this creates a tree decomposition whose tree is no longer T. Since $|\mathcal{F}(b)| \leq |\mathcal{F}_p(b)| \leq |\mathcal{C}|^2$ holds for any introduce or forget node $b \in V(T)$, and $|\mathcal{F}(b)| \leq |\mathcal{F}_t(b)| \leq |\mathcal{C}|^3$ holds for any join node $b \in V(T)$, clearly $\|\mathcal{H}_J\| \leq n \cdot |\mathcal{C}|^3$. □

Alternative proof of Theorem 4.3.1. Consider the instance J of Lemma 5.2.8. By Theorem 3.0.1, there exists a polytope $P(J)$ with $\mathrm{xc}(P(J)) \leq \mathcal{O}(|\mathcal{C}| \cdot \|\mathcal{H}_J\| \cdot nm \cdot |V(T)|) \leq \mathcal{O}(|\mathcal{C}|^3 \cdot nm)$. Since $\mathrm{Feas}(I) = S_\varphi(G)$ and J is an extension of I, $P(J)$ is an extension of $P_\varphi(G)$. □

Moreover, using the much stronger Theorem 5.2.3, it is possible to obtain an analogue of Theorem 5.1.9 together with an integer separable minimization oracle. We will leave this result without a formal statement, because we are about to prove an even stronger result, tying together all our developments so far.

Definition 5.2.9 (Local scope property)**.** *Let $m, k \in \mathbb{N}$, G be a graph, (T, B) be its nice tree decomposition, and S be a set of vectors of elements indexed by $V(G) \times [m]$ and $V(T) \times [k]$. We say that S has the* local scope property *if*

$$\forall \mathbf{s} \in S\ \exists a \in V(T):\ \mathrm{supp}(\mathbf{s}) \subseteq \Big(\{(v,i) \mid v \in B(a), i \in [m]\} \cup \{(b,j) \mid b \in N_T(a), j \in [k]\}\Big).$$

Lemma 5.2.10 (On extension of $CSP_\varphi(G)$)**.** *Let G with $\mathrm{tw}(G) = \tau$ be a σ_2-structure representing a graph, $\varphi \in \mathsf{MSO}_1$ with m free variables, (T, B) a nice tree decomposition of G of treewidth τ, and $k \in \mathbb{N}$. Furthermore, let $I = (V, \mathcal{D}, \mathcal{H}, \mathcal{S})$ be a CSP instance with $V = \{y_v^i \mid v \in V(G), i \in [m]\} \cup \{x_a^j \mid a \in V(T), j \in [k]\}$, and $\mathcal{H} = \Big\{\{\mathbf{y} \mid G, \mathbf{y} \models \varphi\}\Big\} \cup \mathcal{H}'$, and $\mathcal{H}' \cup \mathcal{S}$ have the local scope property, i.e., the scope of all constraints is restricted to variables corresponding to the descendants of some node $a \in V(T)$.*

Then there exists a CSP instance $J = (V_J, \mathcal{D}_J, \mathcal{H}_J, \mathcal{S}_J)$ which extends I, and a computable function f such that

1. $\mathrm{tw}(J) \leq 2k + m(\tau + 1)$,
2. $\|\mathcal{H}_J\| + \|\mathcal{S}_J\| \leq f(|\varphi|, \tau) \cdot |V| + (\|\mathcal{H}'\| + \|\mathcal{S}\|)$,

[1] We could have constructed J differently to obtain treewidth 1 by replacing arity-3 hard constraints with additional vertices whose domains would be these hard constraints. Such a construction is similar to the one in the proof of Theorem 5.2.3.

3. $D_J = \max(D_I, f(|\varphi|, \tau))$,

Proof. The proof proceeds as follows. First, we use Lemma 5.2.8 to obtain a CSP instance J' which is an extension of $CSP_\varphi(G)$ (recall that $\mathrm{Feas}(CSP_\varphi(G)) = S_\varphi(G)$). Then, we show that if $\mathcal{H}'$ and $\mathcal{S}$ have the local scope property, it is possible to add new constraints derived from $\mathcal{H}'$ and $\mathcal{S}$ to J' in such a way that J has the claimed properties.

Let I' be the $\mathrm{CSP}_\varphi(G)$ instance over variables $\mathbf{y}$ with $\mathcal{H} = \Big\{\{\mathbf{y} \mid G, \mathbf{y} \models \varphi\}\Big\}$. By Lemma 5.2.8 we obtain a CSP instance J' which is an extension of I', and (T, B^*) is a tree decomposition of $G(J')$.

We introduce auxiliary binary variables $f_v^{i,a}$ for each $a \in V(T)$, $i \in [m]$ and $v \in B(a)$, and we impose a hard constraint on $f_v^{i,a}$ and t_a enforcing $f_v^{i,a} = \eta(t_a, a, v, i)$. Recall that by the definition of η, this implies $f_v^{i,a} = y_v^i$. For any subset U of variables of I, let U_a be the set U where each variable y_v^i is replaced by $f_v^{i,a}$. Then, for every constraint $C_U \in \mathcal{H}'$ and $w_U \in \mathcal{S}$ let $a \in V(T)$ be the node such that the local scope property of C_U and w_U, respectively, is fulfilled by a, add to J' an identical constraint with scope U replaced by U_a. We call the resulting instance J. Observe that, for any constraint C_U or w_U, $f_v^{i,a} = y_v^i$ implies that replacing its scope with U_a does not change the set of feasible assignments, and J is an extension of I.

By Lemma 5.2.8, we have that $\|\mathcal{H}_{J'}\| \leq f(|\varphi|, \tau) \cdot n$. Since $\|\mathcal{H}_J \setminus \mathcal{H}_{J'}\| = \|\mathcal{H}'\|$ and $\|\mathcal{S}_J\| = \|\mathcal{S}\|$, we have that $\|\mathcal{H}_J\| + \|\mathcal{S}_J\| \leq f(|\varphi|, \tau) \cdot n + \|\mathcal{H}'\| + \|\mathcal{S}\|$. Clearly, $D_J = \max(D_I, D_{J'}) = \max(D_I, f(|\varphi|, \tau))$.

It remains to show that $\mathrm{tw}(J) \leq 2 + m + 2k$. A lemma will help us see that.

Lemma 5.2.11. *Let $T = (I, F)$ be a rooted binary tree, let (T, B) be a tree decomposition of a graph $G = (V, E)$ of width κ, and let $H = (V \cup W, E \cup Y)$ be a supergraph of G such that:*

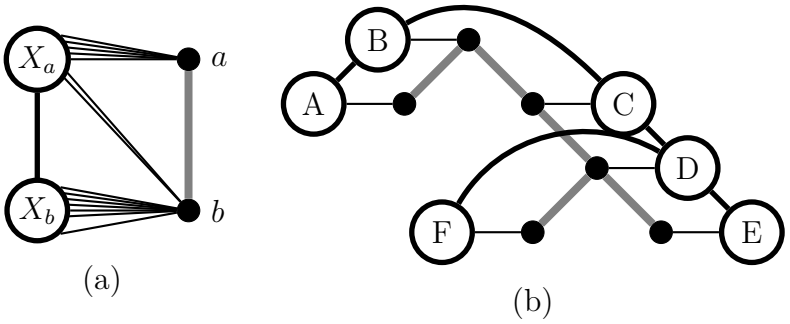
- *$W = \bigcup_{a \in I} W_a$, $Y = \bigcup_{ab \in F} Y_{ab}$, and all W_a and Y_{ab} are mutually disjoint,*
- *$|W_a| \leq \kappa'$ for all $a \in I$,*
- *if $a \in I$ has only child b, then $\bigcup Y_{ab} \subseteq (B(a) \cup W_a \cup W_b)$, and,*
- *if a has two children b, b', then $\bigcup (Y_{ab} \cup Y_{ab'}) \subseteq (B(a) \cup W_a \cup W_b \cup W_{b'})$.*

Then there is a tree decomposition (T, B') of H of width at most $\kappa + 2\kappa'$.

Let B' be obtained from B by, for every edge $ab \in F$, adding W_a to the bags $B(a)$ and $B(b)$. We will verify that (T, B') is a tree decomposition of H of width at most $\kappa + 2\kappa'$; we shall denote by B' the bags of (T, B'). The conditions of a tree decomposition obviously hold for all vertices and edges of G, so we only check it for new vertices and edges.

Edge condition. Let $uv \in Y_{ab}$ be an edge in $H \setminus G$ with $u \in W_a$. Either $v \in B(a)$ and then $\{u, v\} \subseteq B(a) \cup W_a \subseteq B'(a)$, or $v \in X_b$ and then $\{u, v\} \subseteq B(b) \cup W_a \subseteq B'(b)$.

Connectedness condition. Let $v \in W_a$ and let a have children b, b', with possibly $b = b'$. Notice that v does not appear in the bag of any node above a and any node below b and b'. Since we have added W_a to all of a, b and b', the connectedness condition holds.



Proof.

Figure 5.1: The situation of Lemma 5.2.11: the additional vertices X are partitioned along the nodes of T, and only connect to original or new vertices associated with adjacent nodes.

We have added to each node b (except the root) two sets W_a, W_b where a is the parent of b, and because $|W_a| + |W_b| \leq 2\kappa'$, $\mathrm{tw}((T, B')) \leq \kappa + 2\kappa'$. □

Let us consider how the constraint graph $G(J)$ relates to $G(J')$. Since J is obtained by adding new variables and constraints, $G(J)$ is a supergraph of $G(J')$. The vertices $X = V(G(J)) \setminus V(G(J'))$ corresponding to the new variables can be partitioned into sets W_a for every node $a \in V(T)$, and $|W_a| \leq k$. Moreover, the new edges $Y = E(G(J)) \setminus E(G(J'))$ can be partitioned into sets Y_{ab} for each $ab \in E(T)$, such that for each $uv \in Y_{ab}$, $\{u, v\} \subseteq (W_a \cup W_b)$, because, for each node $a \in V(T)$, the new constraints only contain variables associated with a and its neighbors by the local scope property. The tree decomposition (T, B^*) of $G(J')$ is such that we are precisely in the situation of Lemma 5.2.11 with $G := G(J')$, $H := G(J)$, $\kappa := \mathrm{tw}(J') = 2 + m$ and $\kappa' := k + m\tau$, which then implies that $G(J)$ has a tree decomposition (T, B') of width $2k + m + m\tau \leq 2k + m(\tau + 1)$. □

Definition 5.2.12 (Integer separable minimization oracle over r-multisets)**.** *Let $S \subseteq \mathbb{N}^n$. Abusing our terminology slightly, we say that* an integer separable minimization oracle over the r-multisets of S *is an integer separable minimization oracle for $rS = \{\mathbf{x}^1 + \cdots + \mathbf{x}^r \mid \{\mathbf{x}^1, \ldots, \mathbf{x}^r\} \in \left(\!\!\binom{S}{r}\!\!\right)\}$, which, together with a minimum $\mathbf{x}$, returns its* decomposition $\mathbf{x}^1 + \cdots + \mathbf{x}^r = \mathbf{x}$.

Theorem 5.2.13 (Master MSO-CSP theorem)**.** *Let G with $\mathrm{tw}(G) = \tau$ be a σ_2-structure representing a graph, φ be an MSO_1 σ_2-formula with m free variables, (T, B) a nice tree decomposition of G of width τ, and $k \in \mathbb{N}$. Furthermore, let $I = (V, \mathcal{D}, \mathcal{H}, \mathcal{S})$ be a CSP instance with $V = \{y_v^i \mid v \in V(G), i \in [m]\} \cup \{x_a^j \mid a \in V(T), j \in [k]\}$, and $\mathcal{H} = \left\{\{\mathbf{y} \mid G, \mathbf{y} \models \varphi\}\right\} \cup \mathcal{H}'$, and $\mathcal{H}'$ and $\mathcal{S}$ have the local scope property.*

Then there exists a constructively decomposable extended formulation $P(J)$ of the polytope $CSP(J)$ of size $D_I^{(f(|\varphi|,\tau)+2k)}$. Moreover, it is possible to realize an integer separable minimization oracle over the set $rP(J)$, and $\left(\!\!\binom{\mathrm{Feas}(J)}{r}\!\!\right)$ in time $\min(D_I^{r\cdot(f(|\varphi|,\tau)+2k)}, r^{D_I^{(f(|\varphi|,\tau)+2k)}}) \cdot (\|\mathcal{H}'\| + \langle g \rangle + nmk)$ when queried on a sepable function g.

Proof. The proof is a straightforward application of Theorem 5.2.3 to the instance obtained by Lemma 5.2.10. An integer separable minimization oracle for $rP(J)$

is provided directly by Theorem 5.1.9. Regarding minimization over $\left(\!\!\binom{\mathrm{Feas}(J)}{r}\!\!\right)$, let $(\mathbf{y}, \mathbf{f}, \mathbf{h}) \in rP(J)$ be a minimum when the oracle for $rP(J)$ is queried on g. Since $rP(J)$ is constructively decomposable, we find in polynomial time a decomposition $(\mathbf{y}, \mathbf{f}, \mathbf{h}) = (\mathbf{y}^1, \mathbf{f}^1, \mathbf{h}^1) + \cdots + (\mathbf{y}^r, \mathbf{f}^r, \mathbf{h}^r)$. Clearly $\{\mathbf{y}^1, \ldots, \mathbf{y}^r\} \in \left(\!\!\binom{\mathrm{Feas}(J)}{r}\!\!\right)$, and it is minimal such r-multisubset with respect to g.

Regarding time complexity, combining the bounds of Theorem 5.2.3 and Lemma 5.2.10 gives

$$\begin{aligned}\min(D_J^{\mathcal{O}(r\cdot\mathrm{tw}(J))}, r^{D_J^{\mathcal{O}(\mathrm{tw}(J))}}) \cdot (\|\mathcal{H}_J\| + \langle g\rangle + nmk) \\ \leq \min(D_I^{r\cdot(f(|\varphi|,\tau)+2k)}, r^{D_I^{(f(|\varphi|,\tau)+2k)}}) \cdot (\|\mathcal{H}'\| + \langle g\rangle + nmk)\end{aligned}$$

□

6. MSO extensions

We start this chapter by introducing several extensions of the MSO logic in Section 6.1. This provides a summary of prior work and allows us to spot distinguishing properties of extensions with respect to computational complexity. Then, in Section 6.2 we apply Theorem 5.2.13 to obtain an XP algorithm for the logic $\mathsf{MSO}^{\mathsf{GL}}$ on graphs of bounded treewidth. Finally, Section 6.3 deals with graphs of bounded neighborhood diversity, providing two hardness results, an FPT algorithm for the linear extension $\mathsf{MSO}^{\mathsf{GL}}_{\mathsf{lin}}$, and an XP algorithm for the $\mathsf{MSO}^{\mathsf{GL}}$ logic.

6.1 MSO Extensions

Let us introduce extensions of the MSO logic. We consider two orthogonal ways to extend MSO. In what follows, φ is a formula with m free set variables.

Global cardinality constraints. We introduce a new type of atomic formulae called *global cardinality constraints* (*global constraints* for short). An MSO formula with c global cardinality constraints contains m-ary predicates $R_1, \ldots, R_c$ where each predicate takes as argument only the free variables of φ. The input to the model checking problem is a graph $G = (V, E)$ on n vertices and a tuple $(R_1^G, \ldots, R_c^G)$, where $R_i^G \subseteq [n]^m$.

To define the semantics of the extension, it is enough to define the truth of the newly introduced atomic formulae. A formula $R_i(X_1, \ldots, X_m)$ is true under an assignment $\mu\colon \{X_1, \ldots, X_m\} \to 2^V$ if and only if $(|\mu(X_1)|, \ldots, |\mu(X_m)|) \in R_i^G$.

We allow the relations to be represented either explicitly as a list of tuples, or implicitly as a linear constraint $a_1|X_1| + \cdots a_m|X_m| \leq b$, where $(a_1, \ldots, a_m, b) \in \mathbb{R}^{m+1}$.

For example, suppose we want to satisfy a formula $\varphi(X_1, X_2)$ with two sets for which $|X_1| \geq |X_2|^2$ holds. Then, we solve the $\mathsf{MSO}^{\mathsf{G}}$ Model Checking problem with a formula $\varphi' := \varphi \wedge [|X_1| \geq |X_2|^2]$. Notice that we write the relation as a part of the formula, as this is a more convenient way to think of the problem. However, formally the relation is a part of the input.

Local cardinality constraints. Local cardinality constraints are additional cardinality requirements such that every variable assignment has to satisfy the cardinality constraint for every vertex and for every free variable. Specifically, we want to control the size of $\mu(X) \cap N(v)$ for every $v \in V(G)$; we define a shorthand $S(v) = S \cap N(v)$ for a subset $S \subseteq V$ and vertex $v \in V(G)$.[1] *Local cardinality constraints* for a graph $G = (V, E)$ on n vertices and a formula φ with m free variables are mappings $\alpha_1, \ldots, \alpha_m$, where each α_i is a mapping from V to $2^{[n]}$.

We say that an assignment μ *obeys local cardinality constraints* $\alpha_i, \ldots, \alpha_m$ if for every $i \in [m]$ and every $v \in V(G)$ it holds that $|\mu(X_i)(v)| \in \alpha_i(v)$.

[1] More accurately, Szeider [111] introduces local cardinality constraints in a way which constrain the size of $S(v)$ also when S is an *edge* set $S \subseteq E(G)$. In Subsection 6.1.2 we show that instead of G, we can equivalently work with a different structure which has elements for both vertices and edges of the original graph G, and preserves their incidence. Then, our definition also captures the more general original definition.

The logic that incorporates both of these extensions is denoted as $\mathsf{MSO}^{\mathsf{GL}}$. Let φ be an $\mathsf{MSO}^{\mathsf{GL}}$ formula with c global cardinality constraints.

$\mathsf{MSO}^{\mathsf{GL}}$ Model Checking
Input: A graph $G = (V, E)$ on n vertices, relations $R_1^G, \ldots, R_c^G \subseteq [n]^m$, and mappings $\alpha_1, \ldots, \alpha_m : V \to 2^{[n]}$.
Task: Find an assignment μ that obeys local cardinality constraints and such that φ is true under μ by the semantics defined above.

The $\mathsf{MSO}^{\mathsf{GL}}$ logic is very powerful and, as we later show, does not admit an FPT model checking algorithm for the parameterization even by the very restrictive parameter vertex cover number, which rules out an FPT algorithm for both treewidth and neighborhood diversity. It is therefore relevant to consider weakenings of the $\mathsf{MSO}^{\mathsf{GL}}$ logic:

$\mathsf{MSO}^{\mathsf{G}}$ Only global cardinality constraints are allowed.

$\mathsf{MSO}^{\mathsf{L}}$ (**originally** MSO-LCC [**111**]) Only local cardinality constraints are allowed.

$\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$ (**originally** CardMSO [**47**]) The cardinality constraints can only be linear; that is, we allow constraints in the form $[e_1 \geq e_2]$, where e_i is linear expression over $|X_1|, \ldots |X_m|$.

$\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$ Only local cardinality constraints are allowed; furthermore every local cardinality constraint α_i must be of the form $\alpha_i(v) = [l_i^v, u_i^v]$, (i.e., an interval) where $l_i^v, u_i^v \in [n]$. Those constraints are referred to as *linear local cardinality constraints*.[2]

$\mathsf{MSO}^{\mathsf{GL}}_{\mathsf{lin}}$ A combination of $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$ and $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$; both local and global constraints are allowed, but only in their linear variants.

The model checking problem for the considered fragments is defined in a natural way analogously to $\mathsf{MSO}^{\mathsf{GL}}$ model checking.

By saying that φ is an MSO^* formula, we mean that φ is a formula from any of the extensions defined above.

6.1.1 Pre-evaluations

Many techniques used for designing MSO model checking algorithms fail when applied to MSO extensions. A common workaround is first transforming the given MSO^* formula into an MSO formula by fixing values of all global constraints to either `true` or `false`. Once we determine which variable assignments satisfy the transformed MSO formula, we can by other means (e.g, ILP or CSP) ensure that they obey the constraints imposed by fixing the values to `true` or `false`. This approach was first used for CardMSO by Ganian and Obdržálek [49]. We formally describe this technique as *pre-evaluations*:

[2]The logic fairMSO introduced by Kolman et al. [72] is a further restriction of $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$; now we only allow $\alpha_i(v) = [0, u_i^v]$. Since the hardness results for $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$ already hold for fairMSO [90], and our positive results hold for the more general $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$, we do not study fairMSO any further.

Definition 6.1.1 (Pre-evaluation)**.** *Let φ be an* MSO^* *formula. Denote by $C(\varphi)$ the list of all global constraints. A mapping $\beta\colon C(\varphi) \to \{\texttt{true}, \texttt{false}\}$ is called a* pre-evaluation function on φ. *The* MSO *formula obtained by replacing each global constraint $c_i \in C(\varphi)$ by $\beta(c_i)$ is denoted by $\beta(\varphi)$ and is referred to as a* pre-evaluation of φ.

Definition 6.1.2 (Coplying with a pre-evaluation)**.** *A variable assignment μ of an* MSO^* *formula φ* complies *with a pre-evaluation function β if every global constraint $c_i \in C(\varphi)$ evaluates to $\beta(c_i)$ under the assignment μ.*

6.1.2 Regarding MSO_1 and MSO_2

Despite the fact that MSO_2 is strictly stronger than MSO_1 (hamiltonicity is expressible in MSO_2 but not in MSO_1 [83]), it is known [69] that on graphs with bounded treewidth their power is equal by an argument we shall review below. We will show that only a small change in the argument still works even for our extensions of MSO.

As we discussed in Section 2.2, a graph $G = (V, E)$ is typically viewed as either a σ_1 or a σ_2-structure. Then, every MSO_2 σ_1-formula about G can be rewritten into an equivalent MSO_1 σ_2-formula about its *incidence graph* $I(G) = (V_I, \emptyset, \{E_I, L_V, L_E\})$, with $V_I = V \cup E$, $E_I = \{\{v, e\} \mid v \in e, e \in E\}$, $L_V = V$ and $L_E = E$ (cf. Section 2.2). The crucial observation typically used in the literature is that the treewidth of the structure $I(G)$ is equal to the original graph structure G [69].

Let us examine this observation more closely. The σ_2-structure $I(G)$ is essentially obtained from G by subdividing every edge and labelling the original vertices by L_V and the new vertices by L_E. However, the local cardinality constraints in MSO_2 require counting the number of both incident edges *and* vertices, which is impossible in $I(G)$, since two vertices u, v incident in G are no longer incident in $I(G)$. (Global cardinality constraints pose no problems.) What we need is to reintroduce the edge uv into $I(G)$; let us call the resulting structure $I^{\mathsf{L}}(G)$.

Fortunatelly, we will show this does not increase treewidth by more than one. Observe that we can equivalently view the process obtaining $I^{\mathsf{L}}(G)$ as copying each edge e of G and then subdividing one copy of e and letting the other as is. Then, if (T, B) is a tree decomposition of G, there must exist a node b with $e \subseteq B(b)$. We subdivide T such that there is a distinct such node b_e for each edge $e \in E(G)$; then, for each edge $e \in E(G)$, we insert the vertex obtained by subdividing e into $B(b_e)$. This results in a tree decomposition (T', B') which is a tree decomposition of $I^{\mathsf{L}}(G)$ (or, more specifically, its Gaifman graph) and whose treewidth is 1 larger than that of (T, B).

For this reason, when considering graphs of bounded treewidth, we focus on MSO_1 formulae over σ_2-structures (i.e., graphs with labels L_V and L_E), rather than working with MSO_2.

On graphs of bounded neighborhood diversity, MSO_2 is strictly more powerful than MSO_1; however, model checking of an MSO_2 formula is not even in XP unless $\mathsf{E} = \mathsf{NE}$ [25, 76]. Thus, here too we restrict our attention to MSO_1 and use MSO as a shortcut for MSO_1.

6.2 XP Algorithm For $\mathsf{MSO}^{\mathsf{GL}}$ on Bounded Treewidth

We consider a natural optimization version of $\mathsf{MSO}^{\mathsf{GL}}$ model checking:

Weighted $\mathsf{MSO}^{\mathsf{GL}}$ Model Checking
Input: An $\mathsf{MSO}^{\mathsf{GL}}$ model checking instance, weights $\mathbf{w}^1, \ldots, \mathbf{w}^m \in \mathbb{Z}^n$
Task: Find an assignment $X_1, \ldots, X_m$ satisfying the $\mathsf{MSO}^{\mathsf{GL}}$ Model Checking instance and minimizing $\sum_{j=1}^{m} \sum_{v \in X_j} w_v^j$

Theorem 6.2.1. *There is an algorithm that solves the* Weighted $\mathsf{MSO}^{\mathsf{GL}}$ Model Checking *problem in time* $n^{f(|\varphi|,\tau)}$*, where* $\tau = \operatorname{tw}(G)$ *and* f *is a computable function.*

The merit of Theorem 6.2.1 lies not only in being a very general tractability result, but also in showcasing a simplified way to prove a metatheorem extending MSO using Theorem 5.2.13. In the $\mathsf{MSO}^{\mathsf{GL}}$ Model Checking problem, we wish to find a satisfying assignment of some formula φ which satisfies further constraints. Simply put, Theorem 5.2.13 says that it is possible to restrict the set of satisfying assignments of a formula $\varphi \in \mathsf{MSO}_1$ with CSP constraints under the condition that these additional constraints are structured along the tree decomposition of G. This allows the proof of Theorem 6.2.1 to simply be a CSP formulation satisfying this property.

Proof of Theorem 6.2.1. As is standard when dealing with MSO extensions (cf. Subsection 6.1.1), we first observe that there are at most $2^{|\varphi|}$ different pre-evaluations $\beta(\varphi)$ of φ, so we can try each and choose the best result. Let a pre-evaluation $\beta(\varphi)$ be fixed from now on.

Let (T, B) be a nice tree decomposition of G. We will now construct a CSP instance I satisfying the conditions of Theorem 5.2.13. Let y_v^i be the variables as in Theorem 5.2.13; we use the hard constraint

$$G, \mathbf{y} \models \beta(\varphi)$$

to enforce that each feasible solution complies with the pre-evaluation $\beta(\varphi)$. Now we will introduce additional CSP variables and constraints in two ways to enforce the local and global cardinality constraints. Observe that we introduce the additional CSP variables and constraints in such a way that they have the local scope property of Theorem 5.2.13, that is, their scopes will always be limited to the neighborhood of some node $a \in V(T)$.

Global cardinality constraints. In addition to the original $\mathbf{y}$ variables, we introduce, for each node $a \in V(T)$ and each $j \in [m]$, a variable s_a^j with domain $[n]$. The meaning of this variable is $s_a^j = |X_j \cap V(G_a)|$. Thus, in the root node r, s_r^j is exactly $|X_j|$. To enforce the desired meaning of the variables $\mathbf{s}$, we add the following hard constraints:

$$s_a^j = 0 \quad \text{For all leaves } a \qquad \bigwedge \qquad s_a^j = s_b^j + y_v^j \quad \text{For all } a = b * (v)$$

$$s_a^j = s_b^j \quad \text{For all } a = b \dagger (v) \qquad \bigwedge \qquad s_a^j = s_b^j + s_{b'}^j - \sum_{v \in B(a)} y_v^j \quad \text{For all } a = \Lambda(b, b')$$

To enforce the cardinality constraints themselves, we add:

$$(s_r^1, \ldots, s_r^m) \in R \qquad \forall R : \beta(R) = \texttt{true}$$
$$(s_r^1, \ldots, s_r^m) \in ([n]^m \setminus R) \qquad \forall R : \beta(R) = \texttt{false}$$

Local cardinality constraints. For every node $a \in V(T)$, every $j \in [m]$ and every vertex $v \in B(a)$, introduce a variable λ_a^{vj} with domain $[n]$, with the meaning $\lambda_a^{vj} = |N_{G_a}(v) \cap X_j|$. This is enforced by setting:

$$\lambda_a^{vj} = \sum_{\substack{u : v \neq u \in B(a) \\ uv \in E}} y_u^j \qquad \forall a = b * (v)$$
$$\lambda_a^{uj} = \lambda_b^{uj} + y_v^j \qquad \forall a = b * (v), u \in B(a), uv \in E$$
$$\lambda_a^{vj} = \lambda_b^{vj} \qquad \forall a = b \dagger (u), u \neq v, v \in B(a)$$
$$\lambda_a^{vj} = \lambda_b^{vj} + \lambda_{b'}^{vj} - \sum_{\substack{u : v \neq u \in B(a) \\ uv \in E}} y_v^j \qquad \forall a = \Lambda(b, b')$$

Now the local cardinality constraints themselves are enforced by setting:

$$\lambda_{\mathrm{top}(v)}^{vj} \in \alpha^j(v) \qquad \text{For all } v \in V, \quad j \in [m] \ .$$

Objective function. Use the integer separable minimization oracle for $\mathrm{Feas}(I) = \left(\!\!\binom{\mathrm{Feas}(I)}{1}\!\!\right)$ with $f(\mathbf{y}) = \sum_{j=1}^m \sum_{v \in V(G)} y_v^j w_v^j$, which is a linear (and thus separable) function.

Since the CSP instance I we have constructed satisfies the local scope property, we are ready to apply Theorem 5.2.13. Let us determine the necessary parameters. We have introduced m variables s per node, and $m\tau$ variables λ per node. Thus, $k = (\tau + 1)m$. Let $N = \sum_{j=1}^c |R_j^G| + \sum_{j=1}^m \sum_{v \in V(G)} |\alpha_j(v)|$ be the input length of the global and local cardinality constraints. Since $\|\mathcal{H}'\| \leq N$, we have that the optimum can be found in time $n^{f(|\varphi|,\tau)} + N$, finishing the proof. □

Conditional cardinality constraints. As Szeider [111] points out, it is easy to extend his XP result for $\mathsf{MSO}^{\mathsf{L}}$ in such a way that the local cardinality constraint $|X(v)| \in \alpha(v)$ is conditioned on the fact that $v \in X$. Observe that our approach can be extended in such a way as well. Moreover, in our setting with multiple set variables, we can even condition on an arbitrary predicate $\psi(v, X_1, \ldots, X_m)$ describing how vertex v relates to the set variables.

Extended formulations. Since the main engine of our proof was Theorem 5.2.13, it is easy to see the following:

Corollary 6.2.2. *Let G be a σ_2-structure of treewidth τ, $I = (\varphi, (R_1, \ldots, R_c), \alpha)$ be an instance of* $\mathsf{MSO}^{\mathsf{GL}}$ model checking*, and $P_\varphi(G) = \mathrm{conv}(S_\varphi(G))$ be the* $\mathsf{MSO}^{\mathsf{GL}}$ *polytope of satisfying assignments of instance I. Then $\mathrm{xc}(P_\varphi(G)) \leq n^{f(|\varphi|,\tau)}$ for some computable function f.*

6.2.1 Applications

Let us briefly sketch some consequences of Theorem 6.2.1 and Corollary 6.2.2. We focus on showing how to encode various W[1]-hard (w.r.t. treewidth) problems

using the notions we have provided. The parameterized complexity statements which follow are not very surprising and in many cases were known. Still, we believe that our approach captures and summarizes them nicely. On the other hand, these are the first compact extended formulations for such polytopes as the polytope of equitable colorings, capacitated dominating sets etc., to the best of our knowledge.

Local constraints. While introducing $\mathsf{MSO}^{\mathsf{L}}$, Szeider [111] points out that the problems General Factor, Equitable r-Coloring and Minimum Maximum Outdegree are expressible in $\mathsf{MSO}^{\mathsf{L}}$. Let us now observe that using the extension to conditional local constraints, we can also express the problems Capacitated Dominating Set, Capacitated Vertex Cover, Vector Dominating Set and Generalized Domination.

Take for example the Capacitated Dominating Set problem. There, we are given a graph $G = (V, E)$ together with a capacity function $c : V \to \mathbb{N}$, and our goal is to find a subset $D \subseteq V$ and a mapping $f : V \setminus D \to D$ such that for each $v \in D$, $|f^{-1}(v)| \leq c(v)$. Essentially, $f(w) = v$ means that the vertex w is dominated by the vertex v, and the condition $|f^{-1}(v)| \leq c(v)$ ensures that the mapping f respects the capacities. To encode this problem using $\mathsf{MSO}^{\mathsf{L}}$, we need to view G as its incidence graph $I(G)$ whose vertex set $V_I = V \cup E$. Then, we let $\varphi(D, F)$ and α be a formula and local cardinality constraints enforcing that:

- $D \subseteq V$,
- $F \subseteq E$,
- each $v \in V$ is either in D or has a neighbor in F,
- each $e \in F$ has a neighbor in D, and,
- if $v \in D$, then $|N(v) \cap F| \in \alpha_F(v) = [0, c(v)]$.

Then D encodes a dominating set and F can be used to construct the mapping f since for each edge $e \in F$, at most one of its endpoints is not in D, and each $v \in D$ sees at most $c(v)$ edges from F.

Let us define the remaining problems; their $\mathsf{MSO}^{\mathsf{L}}$ formulations are analogous to the one above. The Vector Dominating Set problem is similar to Capacitated Dominating Set, except now each vertex v has a *demand* $d(v)$ and if $v \notin D$, then it must have at least $d(v)$ neighbors in D. In Generalized Domination, we are given two sets $\sigma, \rho \subseteq \mathbb{N}$ and for each vertex v in D or in $V \setminus D$, it must hold that $|N(v) \cap D| \in \sigma$ or $|N(v) \cap D| \in \rho$, respectively. Finally, the Capacitated Vertex Cover problem is the following. We are given a capacitated graph, and the task is to find a vertex cover $C \subseteq V$ and an assignment $f : E \to C$ such that for each v, $|f^{-1}(v)| \leq c(v)$.

Global constraints: r-Balanced Partitioning. Ganian and Obdržálek [49] introduce $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$ and show that also this logic expresses Equitable r-Coloring, and moreover the Equitable connected r-Partition problem. Interestingly, they also discuss the complexity of the r-Balanced Partitioning problem, where our goal is to find an equitable (all parts of size differing by at most one) r-partition and, moreover, minimize the number of edges between partites. They provide an FPT algorithm for graphs of bounded vertex cover,

but are unable to express the problem in $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$, and thus pose as an open problem the task of finding a more expressive formalism which would capture this problem. They also state that no parameterized algorithm exists for graphs of bounded treewidth, but that is no longer true due to the results of van Bevern et al. [114]. On the other hand, the question of capturing r-Balanced Partitioning by some MSO extension stands. Here we show that it can be expressed as an instance of Weighted $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$ Model Checking when we use edge set variables (thus this is not applicable to graphs of bounded neighborhood diversity).

Let φ be an $\mathsf{MSO}^{\mathsf{G}}$ formula with r free vertex set variables $X_1, \ldots, X_r$ and one free edge set variable Y. We use φ to express that $X_1, \ldots, X_r$ is an equitable r-partition; this is easily done using the global constraints. Furthermore, we enforce that Y is the set of edges with one endpoint in X_i and another in X_j for any $i \neq j$. For a satisfying assignment $X_1, \ldots, X_r, Y$, let $(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{r|V|} \times \{0,1\}^{|E|}$ be its characteristic vector. To minimize the number of edges between partites, it suffices to minimize $\mathbf{y}$. This also clearly extends to the case studied in the literature when edges are assigned weights.

Moreover, it is not difficult to see that the CSP instance in the proof of Theorem 6.2.1 could be extended to enforce constraints such as $\sum_{v \in V} |X(v)| \in A$ for some set $A \subseteq \mathbb{N}$. With such constraints, it is possible to directly express the set of minimal (by appropriate choice of A) r-balanced partitions, which in turn provides a compact extended formulation.

Global constraints: Graph Motif. Fellows et al. [36] study the Graph Motif problem from the perspective of parameterized complexity, especially on graphs with bounded widths. In Graph Motif, we are given a vertex-colored graph G and a multiset of colors M, and the task is to find a *motif*, that is, a connected subset of vertices $S \subseteq V$ such that the multiset of colors of S is exactly M. This problem is most naturally expressed when the vertex-colored graph G is encoded as a general relational structure over a finite vocabulary. Since we have not explicitely phrased our results for such structures, Graph Motif does not directly fit any of our notions; however, the extension of our results to general structures is straightforward.

Then, let us consider the number of colors χ a parameter and introduce additional unary relations (labels) $L_1, \ldots, L_\chi$. It is easy to see that Graph Motif is encoded by the following $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$ formula $\varphi(S)$:

$$\varphi(S) \equiv \text{connected}(S) \wedge \bigwedge_{i=1}^{\chi} [|S \cap L_i| = \text{mult}(i, M)],$$

where connected(S) is a formula which holds if S is connected, and $\text{mult}(i, M) \in \mathbb{N}$ is the multiplicity of color i in the motif M.

Theorem 6.2.3. *Let G be a graph of treewidth τ and with $n = |V(G)|$.*

(XP) *The following problems have algorithms with runtime $n^{f(\tau)}$ and extended formulations of the same size:* General Factor, Minimum Maximum Outdegree, Capacitated Dominating Set, Capacitated Vertex Cover, Vector Dominating Set, Generalized Domination.

(FPT) *The following problems have algorithms with runtime $f(\tau + k) \cdot n^{\mathcal{O}(1)}$ and extended formulations of the same size, with k specified further:*

- Equitable r-Coloring, Equitable connected r-Partition, r-Balanced Partitioning, *with* $k = r$,
- Graph Motif, *with* $k = \chi$, *where* χ *is the number of colors.*

6.3 Graphs of Bounded Neighborhood Diversity

For graphs of bounded neighborhood diversity we prove two negative results (Theorems 6.3.1 and 6.3.2) and two positive results (Theorems 6.3.5 and 6.3.15).

6.3.1 W[1]-hardness of $\mathsf{MSO}^{\mathsf{L}}$ and $\mathsf{MSO}^{\mathsf{G}}$

Theorem 6.3.1. *The* $\mathsf{MSO}^{\mathsf{G}}$ Model Checking *problem is* W[1]-hard *when parameterized by* $\mathrm{vc}(G)$ *and* $|\varphi|$.

Theorem 6.3.2. *The* $\mathsf{MSO}^{\mathsf{L}}$ Model Checking *problem is* W[1]-hard *when parameterized by* $\mathrm{vc}(G)$ *and* $|\varphi|$.

We begin with a definition of an auxiliary problem:

LCC Subset
Input: Graph $G = (V, E)$ with $|V| = n$ and a function $f\colon V \to 2^{[n]}$.
Task: Find a set $U \subseteq V$ such that for each vertex $v \in V$ it holds that $|U(v)| \in f(v)$.

Obviously LCC Subset is equivalent to $\mathsf{MSO}^{\mathsf{L}}$ with an empty formula φ with one free variable. We call an LCC Subset instance *uniform* if, on G with $\mathrm{nd}(G) = k$, the demand function f can be written as $f : [k] \to 2^{[n]}$, such that vertices of one type have the same demand set. We show that already uniform LCC Subset is W[1]-hard by a reduction from the W[1]-hard k-Multicolored Clique problem [28].

k-Multicolored Clique *Parameter:* k
Input: k-partite graph $G = (V_1 \dot\cup \cdots \dot\cup V_k, E)$, where V_a is an independent set for every $a \in [k]$.
Task: Find a clique of size k.

We refer to a set V_a as to a *colorclass* of G. Our proof is actually a simplified proof of W[1]-hardness for the Target Set Selection problem [31].

Theorem 6.3.3. *The* LCC Subset *problem is* W[1]-hard *when parameterized by the vertex cover number, already in the case when* $f(v) = \{0\}$ *for all* v *not belonging to the vertex cover.*

Proof. Denote $G = (V_1 \cup \cdots \cup V_k, E)$ the instance graph for k-Multicolored Clique. We naturally split the set of edges E into sets $E_{\{a,b\}}$ by which we denote the edges between colorclasses V_a and V_b. We may assume that all colorclasses are of the same size which we denote n, and similarly for the number of edges between any two colorclasses which we denote m. Fix $N > n$, say $N = n^2$, and distinct $a, b \in [k]$.

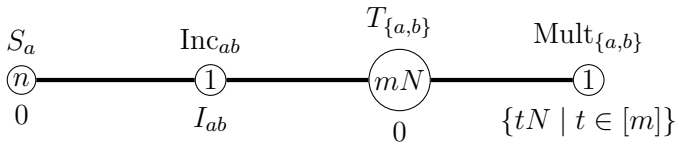


Figure 6.1: An overview of the decomposition of a gadget used in the proof of Theorem 6.3.3. Numbers inside nodes denote the number of vertices in the independent set represented by the node. Below each node a description of the respective set of admissible numbers is shown.

Description of the reduction. We numerate vertices in each color class V_a for $a \in [k]$ using numbers in $[n]$ and denote the numeration of vertex v as n_v^a. We also numerate the edges between color classes a and b by numbers in $[m]$ and denote the numeration of edge e as $m_e^{\{a,b\}}$. Let $I_{ab} = \left\{n_v^a + Nm_e^{\{a,b\}} \mid v \in e, e \in E_{\{a,b\}}\right\}$. We build the graph using the following groups of vertices (refer to Figure 6.1):

- an independent set S_a of size n for each color class V_a and set $f(v) = \{0\}$ for every $v \in S_a$,
- an independent set $T_{\{a,b\}}$ of size mN for each edge set $E_{\{a,b\}}$, with $f(v) = \{0\}$ for every $v \in T_{\{a,b\}}$,
- a single vertex $\mathrm{Mult}_{\{a,b\}}$ with $f(\mathrm{Mult}_{\{a,b\}}) = \{tN \mid t \in [m]\}$,
- a single vertex Inc_{ab} with $f(\mathrm{Inc}_{ab}) = I_{ab}$.

Finally, we add a complete bipartite graphs between S_a and Inc_{ab}, between Inc_{ab} and $T_{\{a,b\}}$, and between $T_{\{a,b\}}$ and $\mathrm{Mult}_{\{a,b\}}$. Denote the resulting graph H. It is straightforward to check that the vertices $\mathrm{Mult}_{\{a,b\}}$ together with vertices Inc_{ab} form a vertex cover of H. It follows that $\mathrm{vc}(H) = \binom{k}{2} + k(k-1)$.

Correctness of the reduction. Suppose there is a clique of size k in G with set of vertices $\{v_1, \ldots, v_k\}$. We select n_{v_a} vertices in the set S_a and $m_e^{\{v_a,v_b\}}$ vertices in the set $T_{\{a,b\}}$. It is straightforward to check that this is a solution respecting demands in H.

Suppose the LCC Subset problem has a solution U in H. First note that none of vertices $\mathrm{Mult}_{\{a,b\}}$, Inc_{ab} is selected as their neighborhood demands are set to 0. Denote $s_a = |U \cap S_a|$ and $t_{\{a,b\}} = |T_{\{a,b\}} \cap U|$. Now observe that because the demand of vertex $\mathrm{Mult}_{\{a,b\}}$ is fulfilled, then there are $t_{\{a,b\}} = tN$ vertices with $0 \leq t < m$. We denote by e_{ab} the edge with numeration $m_{e_{\{a,b\}}} = t$. As the demand of vertex Inc_{ab} is fulfilled the vertex v_a with $n_{v_a} = s_a$ as well as for the vertex Inc_{ba} and vertex v_b. This implies that both v_a and v_b are incident to edge $e_{\{a,b\}}$. □

Note that Theorem 6.3.2 follows easily from Theorem 6.3.3.

Multidemand Set Cover. Recently, Bredereck et al. [17] showed important applications of the Weighted Set Multicover problem when the size of the universe is a parameter. The hardness result above allows us to show a hardness result for a variant of the Set Cover problem, in contrast with the positive results of [17]:

Multidemand Set Multicover
Input: Universe $U = [k]$, set of multidemands $d_1, \ldots, d_k \subseteq [n]$, a covering system represented by a multisubset $\mathcal{F} = \{F_1, \ldots, F_n\} \subseteq 2^U$, weights $w_1, \ldots, w_n \in \mathbb{Z}$.
Task: Find a multisubset $\mathcal{F}' \subseteq \mathcal{F}$ of minimum weight satisfying $(\sum_{j:u_i \in F_j} m_j) \in d_i$ for each $i \in [k]$ or report there is none.

Corollary 6.3.4. Multidemand Set Multicover *is* W[1]-hard *with respect to parameter k, already when $n = k$ and when the task is to just decide whether a multicover exists.*

Proof. Given a uniform instance of LCC Subset on a graph G with $\mathrm{nd}(G) = \nu$ with every type being an independent set[3], and , let $U = [\nu]$, $\mathcal{F} = \{N(v) \mid \forall v \in T(G)\}$, all weights are 1, and $d_i = f(i)$. Then if the instance of Multidemand Set Multicover which we have just defined has a solution, the given LCC Subset instance is a Yes instance, and otherwise it is a No instance. □

Proof of Theorem 6.3.1. Let $(G = (V, E), f, k)$ be an instance of the LCC Subset problem parameterized by the vertex cover number resulting from Theorem 6.3.3. Let $C \subseteq V$ be the vertex cover in G. Note that it follows from the proof of Theorem 6.3.3 that we may assume that the independent set $V \setminus C$ is divided into $\mathcal{O}(k)$ groups, where each group shares the neighborhood in C. Observe further that in fact the graph G is bipartite (i.e., the set C is also an independent set), in particular, the largest clique subgraph of G is of size 2.

By Theorem 6.3.3 we know that it is W[1]-hard to find a subset $X \subseteq V \setminus C$ such that $|X(v)| \in f(v)$ for all $v \in C$. Our goal now is to build an $\mathsf{MSO}^{\mathsf{G}}$ formula expressing exactly this.

First, we take G and construct a graph G' by, for each $v \in C$, attaching a $K_{2+\eta(v)}$ to $N(v)$, where $\eta \colon C \to [k]$ is a bijective mapping. We will call the clique $K_{2+\eta(v)}$ a *marker* because it will allow us to recognize exactly the vertices of $N(v)$. Note that markers are the only cliques present in G' of size at least 3. Note further that by this we have added $\mathcal{O}(k)$ cliques of size $\mathcal{O}(k)$ and thus the resulting graph has a vertex cover of size $\mathcal{O}(k^2)$.

Let us describe some auxiliary formulae which we then use to define the desired formula φ. We reserve X for the set that will represent the set X from the LCC Subset problem.

- Z **is a clique:**

  $\texttt{clique}(Z) := (\forall x, y \in Z)(x \neq y \implies xy \in E)$

- u **and** v **are of the same neighborhood type:**

  $\texttt{same}(u, v) := (\forall w \in V)(w = u \vee w = v \vee (wu \in E \iff wv \in E))$

- Z **is a neighborhood type:**

  $\texttt{type}(Z) := (Z \neq \emptyset) \wedge (\forall u, v \in Z)(\texttt{same}(u, v)) \wedge (\forall u \in Z, v \notin Z)(\neg\texttt{same}(u, v))$

[3]We can indeed assume that every type is an independent set: Theorem 6.3.3 shows hardness for graphs with a small vertex cover, which implies a small neighborhood diversity decomposition where every type is an independent set.

- Z **is $\eta(v)$-th marker:**

  $\texttt{marker}_v(Z) := (|Z| = 2 + \eta(v)) \wedge \texttt{clique}(Z) \wedge \texttt{type}(Z)$

- Z **is $N(v)$:**

  $\texttt{neigh}_v(Z) := \texttt{type}(Z) \wedge (\exists Q \subseteq V)(\texttt{marker}_v(Q) \wedge (\forall u \in Z, w \in Q)(uw \in E))$

- Z **is exactly X_v:**

  $\texttt{sel-neigh}_v(Z, X) := (\exists Z_v)(\texttt{neigh}_v(Z_v) \wedge Z = Z_v \cap X)$

Now $\varphi(X, (X_v)_{v \in C}) := \bigwedge_{v \in C}\Big(\texttt{sel-neigh}_v(X_v, X) \wedge |X_v| \in f(v)\Big)$. We note that only a *unary* global constraint was required. □

6.3.2 FPT Algorithm for $\mathsf{MSO}^{\mathsf{GL}}_{\mathsf{lin}}$

Theorem 6.3.5. *There is an algorithm that solves the* $\mathsf{MSO}^{\mathsf{GL}}_{\mathsf{lin}}$ Model Checking *problem in time $f(|\varphi|, \nu) \cdot n^{\mathcal{O}(1)}$, where $\nu = \mathrm{nd}(G)$ and f is a computable function.*

Essentially, we are modifying the algorithm of Ganian and Obdržálek [49] for $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$ model checking so that it can deal with the additional constraints introduced by $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$.

Definition 6.3.6 (Signature of a variable assignment)**.** *Let φ be a $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$ formula with free set variables $X_1, \ldots, X_m$, let $G = (V, E)$ be a graph with $\mathrm{nd}(G) = \nu$ with types $T_1, \ldots, T_\nu$, and let $\mu \colon \{X_1, \ldots, X_m\} \to 2^V$ be a variable assignment. The* signature of μ *is a mapping from $S_\mu \colon [\nu] \times 2^{[m]} \to \mathbb{N}$ defined by*

$$S_\mu(j, I) = \left| \bigcap_{i \in I} \mu(X_i) \cap T_j \right|.$$

Clearly, if we have two variable assignments μ, μ' with the same signature, then $G, \mu \models \varphi$ if and only if $G, \mu' \models \varphi$.

However, for MSO formulae and graphs of bounded neighborhood diversity, much more is true. Informally speaking, the formula cannot distinguish between two cardinalities if both of them are large. This is formally stated in the next lemma, which is a direct consequence of [75, Lemma 5]:

Lemma 6.3.7. *Let φ be an MSO formula with free set variables $X_1, \ldots, X_m$ that has q_S set quantifiers and q_e element quantifiers. Let G be a graph with $\mathrm{nd}(G) = \nu$. Denote by t the number $2^{q_S} \cdot q_e$. Suppose that μ, μ' are two variable assignments such that for every $I \subseteq [m]$, $j \in [\nu]$, we have either*

- *$S_\mu(j, I) = S_{\mu'}(j, I)$, or*
- *both $S_\mu(j, I), S_{\mu'}(j, I) > t$.*

Then $G, \mu \models \varphi$ if and only if $G, \mu' \models \varphi$.

The last lemma leads to the following definition.

Definition 6.3.8 (Shape of a variable assignment)**.** *Let φ, G and t be as before. A* shape *of a variable assignment μ is a mapping $sh_\mu\colon [\nu] \times 2^{[m]} \to [0,t] \cup \{\uparrow\}$ defined by*

$$sh_\mu(j,I) = \begin{cases} S_\mu(j,I) & \text{if } S_\mu(j,I) \leq t \\ \uparrow & \text{if } S_\mu(j,I) > t \end{cases}$$

Since t depends only on the formula φ, the number of shapes can be bounded by some function of $|\varphi|$ and $\mathrm{nd}(G)$. Note that there are mappings from $[\nu] \times 2^{[m]}$ to $[0,t] \cup \{\uparrow\}$ that do not correspond to shape of any variable assignment μ for a particular graph G. For example, if $sh(j,I) = \uparrow$ for some j and I but $|T_j| < t$, clearly there is no assignment of shape sh.

However, Lemma 6.3.7 cannot be used directly, as the global linear constraints allow us to distinguish small differences in cardinalities, even if the cardinalities are large; consider the constraint $[|X_1| = |X_2| + 1]$. We use the approach outlined in Subsection 6.1.1, Pre-evaluations. This approach relies upon Definitions 6.1.1 and 6.1.2. We simply guess all possible outcomes of the cardinality constraints (there number of such outcomes is clearly bounded by $2^{|\varphi|}$) and later, we ensure that our assignment obeys those constraints by an ILP formulation.

Moreover, the following definition is required for the proof.

Definition 6.3.9 (Admissible shape)**.** *The shape sh is* admissible *with respect to pre-evaluation β, if for any variable assignment μ of shape sh we have $G, \mu \models \beta(\varphi)$.*

Theorem 6.3.10. *There exists an algorithm that, given an $\mathsf{MSO}^{\mathsf{G}}_{\mathsf{lin}}$ formula φ with free set variables $X_1, \ldots, X_m$, a graph $G = (V,E)$ with $\nu = \mathrm{nd}(G)$ and local linear constraints $\alpha_i(v)$, either outputs an assignment μ such that $G, \mu \models \varphi$ and for every $v \in V$ and every $i \in [m]$ we have $|\mu(X_i)(v)| \in \alpha_i(v)$, or correctly reports that no such assignment exists. The algorithm terminates in time $f(|\varphi|, \nu)n^{\mathcal{O}(1)}$ for some computable function f.*

Proof. Denote by $T_1, \ldots, T_\nu$ the types of graph G. Note that the types can be computed in polynomial time.

The algorithm works as follows. For every pre-evaluation function β and every mapping $sh_\mu\colon [\nu] \times 2^{[m]} \to [0,t] \cup \{\uparrow\}$, we test whether sh is admissible. This can be done by picking arbitrary variable assignment μ of shape sh (if one exists) and testing whether $G, \mu \models \beta(\varphi)$ by an FPT model checking algorithm for MSO formulae [75].

If the shape sh is admissible with respect to β, we need to find a variable assignment μ such that

- μ complies with β,
- μ has shape sh, and
- μ satisfies the local linear constraints.

This is done by following ILP.

- for every $I \subseteq [m], j \in [\nu]$, we introduce an integer variable x^j_I (these correspond to $S_\mu(j,I)$ of the variable assignment μ we are trying to find),

- for every $i \in [m], j \in [\nu]$, we introduce an auxiliary variable y_i^j corresponding to $|\mu(X_i) \cap T_j|$, and,
- for every $i \in [m]$, we add an auxiliary variable z_i corresponding to $|\mu(X_i)|$.

We note that, technically, the variables y_i^j and z_i are not needed since they completely depend on $\mathbf{x}$, but will simplify the presentation. To ensure that μ has the required properties, we add these constraints:

$$\sum_{I \subseteq [m]} x_I^j = |T_j| \qquad \text{for every } j \in [\nu] \qquad (0)$$

$$y_i^j = \sum_{\substack{I \subseteq [m]: \\ i \in I}} x_I^j \qquad \text{for every } j \in [\nu] \text{ and every } i \in [m] \qquad (a1)$$

$$z_i = \sum_{j=1}^{\nu} y_i^j \qquad \text{for every } i \in [m] \qquad (a2)$$

$$x_I^j = sh(I, j) \qquad \text{for every } j \in [\nu], I \subseteq [m] \text{ such that } sh(I, j) \neq \uparrow \qquad (sh1)$$

$$x_I^j > t \qquad \text{for every } j \in [\nu], I \subseteq [m] \text{ such that } sh(I, j) = \uparrow \qquad (sh2)$$

The constraints (0) ensure that variables x_I^j encode a variable assignment on a graph G. The constraints (a1) and (a2) set auxiliary variables y_i^j and z_i to the desired values. The constraints (sh1) and (sh2) guarantee that μ has the shape sh.

Local linear cardinality constraints. What remains is to enforce the local linear cardinality constraints. It is relatively easy to see that if a neighborhood diversity decomposition is uniform with respect to the local cardinality constraints, i.e., if all vertices of one type have the same local cardinality constraint, it is possible to incorporate these to the ILP above. Most of the following is dedicated to showing that we can obtain different local cardinality constraints α' and a refinement $\mathcal{T}'$ of the given neighborhood diversity decomposition which is uniform with respect to α', but nonetheless there is a direct correspondence between solutions in the uniform instance (G, α') and the original instance (G, α).

For a graph G and its neighborhood diversity decomposition $\mathcal{T}$ we define $\nu_\alpha(\mathcal{T})$ as the number of nonuniform types in $\mathcal{T}$ with respect to α. Observe that if $\mathcal{T}'$ is a refinement of $\mathcal{T}$, then $\nu_\alpha(\mathcal{T}') \leq \nu_\alpha(\mathcal{T})$. A type $T \in \mathcal{T}$ is said to be *nonuniform* with respect to local linear cardinality constraints α if there exist vertices $u, v \in T$ with $\alpha(u) \neq \alpha(v)$.

Proposition 6.3.11. *Let $G = (V, E)$ be a graph and let $\mathcal{T}$ be a neighborhood diversity decomposition. For every $T \in \mathcal{T}$ and for every $X \subseteq V$ there exists a nonnegative integer z such that for every vertex $w \in T$*

- *if T is an independent set, then $|X(w)| = z$, and,*
- *if T is a clique, then $|X(w)| \in \{z, z + 1\}$.*

Proof. First assume that T is an independent set. Then $N(v) = N(w)$ for all $v, w \in T$. On the other hand, assume that T is a clique, and let $v \in T$, with possibly $v = w$. If both v and w either belong to X or not belong to X, then

clearly $|X(v)| = |X(w)|$. If $v \in X$ and $w \notin X$, then $\Big||X(v)| - |X(w)|\Big| = 1$, and symmetrically for $w \in X$ and $v \notin X$. Since no other possibilities exist we are done. □

Lemma 6.3.12 (Local refinement lemma)**.** *Let $G = (V, E)$ be a graph and $\mathcal{T}$ be its neighborhood diversity decomposition, and let α be local linear cardinality constraints. Let $T \in \mathcal{T}$ be a nonuniform type of G. Then there exists partition $\mathcal{T}'$ of T and local linear cardinality constraints α' such that the following holds*

1. *$|\mathcal{T}'| \leq 4$,*
2. *$\nu_{\alpha'}((\mathcal{T} \setminus \{T\}) \cup \mathcal{T}') < \nu_\alpha(\mathcal{T})$, and*
3. *for each $X \subseteq V$, X satisfies α if and only if X satisfies α'.*

Proof. Let us first argue about the case when T is an independent type. In this case it suffices to set $\alpha'(u) = \bigcap_{v \in T} \alpha(v)$ for each $u \in T$. Now $X \subseteq V$ satisfies α if and only if X satisfies α' as the value $|N(T) \cap X|$ has to be the same for all vertices of T and thus, by Proposition 6.3.11, it has to be in $\alpha'(v)$ for $v \in T$

Let T be a clique type of $\mathcal{T}$. We define

$$\ell = \max_{v \in T} \min \alpha(v), \text{and,}$$
$$u = \min_{v \in T} \max \alpha(v) \ .$$

If $u \leq \ell - 2$, then, by Proposition 6.3.11, α cannot be satisfied. Denote the new local linear constraints $\alpha'(v) = \alpha(v) \cap [\ell - 1, u + 1]$ for $v \in T$ and define $\alpha'(v) = \alpha(v)$ for $v \in V \setminus T$. We get that:

- $\alpha'(v) \subseteq [\ell - 1, u + 1]$ for each $v \in T$ and
- $[\ell, u] \subseteq \alpha'(v)$ for each $v \in T$.

This yields at most 4 possibilities for $\alpha'(v)$. We can refine T into at most 4 subtypes such that all the vertices of a subtype of T have the same $\alpha'(v)$. As all newly introduced types are uniform we have replaced a nonuniform type T with at most 4 uniform types. We have proven Points (1) and (2); in order to prove Point (3) we use the following proposition.

Proposition 6.3.13. *Let $p \in [n]$, let ℓ be defined as above. If there exists $v \in T$ such that $p \in \alpha(v)$ and $p \leq \ell - 2$, then for each X satisfying α, it holds that $p \neq |X(v)|$.*

Proof. This easily follows as for each X satisfying the local cardinality constraints α there exists a $z = z(X, T)$ and by Proposition 6.3.11 each $w \in T$ must contain z or $z+1$ in $\alpha(w)$. Suppose for contradiction that $|X(v)| = p$ and let s be a vertex with $\alpha(s) \subseteq [\ell, n]$ (such s exists from the definition of ℓ). As $p \leq \ell - 2$, it follows that X cannot satisfy $\alpha(s)$. From Proposition 6.3.11, there are two possible options $\{p-1, p\}$ and $\{p, p+1\}$. Observe that $\{p-1, p, p+1\} \cap \alpha(s) = \emptyset$ holds. This finishes the proof of the claim. □

Clearly if $X \subseteq V$ does not satisfy α, then it does not satisfy α' as $\alpha'(v) \subseteq \alpha(v)$ for every $v \in V$. Assume that X satisfies α. By the above claim and its symmetric version for $p \geq u + 2$ it follows that $\ell - 1 \leq X(v) \leq u + 1$. By the definition of α' it follows that X satisfies α'. □

Lemma 6.3.14 (Refinement lemma)**.** *Given a graph $G = (V, E)$ with $\mathrm{nd}(G) = \nu$ and with local linear cardinality constraints α_i for each $i \in [m]$, there exist a neighborhood decomposition $\mathcal{T}$ of G of size at most $\nu 4^m$ and local linear cardinality constraints α'_i for each $i \in [m]$ such that:*

- *the constraints α'_i, for all $i \in [m]$, are uniform with respect to $\mathcal{T}$, and,*
- *for each $(X_1, \ldots, X_m) \subseteq V^m$, X_i satisfies α_i for all $i \in [m]$ if and only if X_i satisfies α'_i for all $i \in [m]$.*

Proof. The proof goes by repeatedly applying Lemma 6.3.12. We start with the neighborhood decomposition $\mathcal{T}$ of size ν that is guaranteed by $\mathrm{nd}(G) = \nu$, and with the local linear cardinality constraints α_i.

First let $i = 1$, and go sequentially over the types $T_1, \ldots, T_\nu$. Apply Lemma 6.3.12 to the presently processed type T_j and the local linear cardinality constraint α'_1 and decomposition $\mathcal{T}'$ resulting from the previous application of the lemma, using α_1 and $\mathcal{T}$ in the beginning. Clearly after we are done we have a neighborhood decomposition $\mathcal{T}'$ of size at most 4ν and local linear cardinality constraints α'_1 which are uniform with respect to $\mathcal{T}'$.

Then, continuing with $i \in [2, m]$, we do the same, finally resulting in a decomposition $\mathcal{T}''$ of size $\nu 4^m$ and local linear cardinality constraints $\alpha''_1, \ldots, \alpha''_m$ which are uniform with respect to $\mathcal{T}''$. □

By the previous lemma, we can without loss of generality assume that each type has uniform constraints, i.e. for every $i \in [m], j \in [\nu]$, and $v, w \in T_j$ we have $\alpha_i(v) = \alpha_i(w)$. For convenience, we denote this unique integer interval by $\alpha_{i,j}$ and furthermore set $\mathrm{lb}^i_j := \min \alpha_{i,j}$ and $\mathrm{ub}^i_j := \max \alpha_{i,j}$.

If T_j is an independent type, we need to ensure that for every $v \in T_j$ we have $|\mu(X_i)(v)| \in \alpha_{i,j}(v)$. It is easy to see that the quantity $|\mu(X_i)(v)|$ is the same for every $v \in T_j$ and it can be expressed as

$$\sum_{j' : \{j', j\} \in E(T_G)} |\mu(X_i) \cap T_{j'}| \ .$$

By the definition of auxiliary variables y^j_I, we have $|\mu(X_i) \cap T_{j'}| = y^{j'}_i$, so the local linear condition for the variable X_i can be rewritten as

$$\mathrm{lb}^i_j \leq \sum_{j' : \{j', j\} \in E(T_G)} y^j_i \leq \mathrm{ub}^i_j. \tag{lli}$$

If T_j is a clique type, we have to be slightly more careful. The quantity $|\mu(X_i)(v)|$ depends on whether v belongs to $\mu(X_i)$ or not; the set $N(v)$ does not include v itself, so if $|\mu(X_i)(v')| = |\mu(X_i)(v)| + 1$ for every $v \in T_j \cap \mu(X_i), v' \in T_j \setminus \mu(X_i)$. Similarly as before, we have equations

$$|\mu(X_i)(v)| = \left(\sum_{j' : \{j', j\} \in E(T_G)} |\mu(X_i) \cap T_{j'}| \right) - 1$$

for $v \in T_j \cap \mu(X_i)$, and

$$|\mu(X_i)(v)| = \sum_{j' : \{j', j\} \in E(T_G)} |\mu(X_i) \cap T_{j'}|$$

for $v \in T_j \setminus \mu(X_i)$.

This means that we need to add the constraint

$$\mathrm{lb}_j^i \leq \sum_{j':\{j',j\}\in E(T_G)} y_i^{j'} \leq \mathrm{ub}_j^i \tag{llc1}$$

if $|\mu(X_i) \cap T_j| \geq 1$ and add the constraint

$$\mathrm{lb}_j^i \leq \sum_{j':\{j',j\}\in E(T_G)} y_i^{j'} - 1 \leq \mathrm{ub}_j^i \tag{llc2}$$

if $|T_j \setminus \mu(X_i)| \geq 1$.

Fortunately, we can deduce whether the conditions $|\mu(X_i) \cap T_j| \geq 1$ or $|T_j \setminus \mu(X_i)| \geq 1$ hold from the shape sh. If we have

$$\sum_{I:i\in I} sh(I,j) > 0, \tag{sh1}$$

then $\mu(X_i)$ necessarily intersects T_j, while if we have

$$\sum_{I:i\notin I} sh(I,j) > 0, \tag{sh2}$$

then there exists vertex in $T_j \setminus X_i$.

This means that the local linear constraints for type T_j and variable X_i can be enforced by adding constraint (llc1) if (sh1) holds, and by adding constraint (llc2) if (sh2) holds.

Let us turn our attention to the analysis of the running time of the algorithm. There are $(t+2)^{\nu\cdot 8^m}$ different shapes (after the refinement) and $2^{|C(\varphi)|}$ pre-evaluation functions. Since t depends only on the number of quantifiers in the formula φ, both numbers can be bounded by a function of $|\varphi|$ and ν. For each such function, we construct an ILP with $\nu \cdot 8^m$ integer variables, and $\mathcal{O}(\nu)$ constraints. By Lenstra's algorithm [81], such an ILP can be solved in FPT time with respect to $|\varphi|$ and ν. □

6.3.3 XP Algorithm for $\mathsf{MSO}^{\mathsf{GL}}$

Theorem 6.3.15. *There is an algorithm that solves the* $\mathsf{MSO}^{\mathsf{GL}}$ Model Checking *problem in time* $n^{f(|\varphi|,\nu)}$*, where* $\nu = \mathrm{nd}(G)$ *and* f *is a computable function.*

We first describe the idea for a formula φ with only one free variable X, and later show how this approach can be generalized for a formula with m free variables $X_1, \ldots, X_m$. As before, we go over all at most $2^{|\varphi|}$ pre-evaluations β of φ. Let us fix a pre-evaluation $\beta(\varphi)$ of the formula φ from now on.

Let $G = (V, E)$ be a graph on n vertices with $\nu = \mathrm{nd}(G)$ and with types $T_1, \ldots, T_\nu$, and let $\alpha(v) \subseteq [0, |V|]$ be a set of integers for every vertex $v \in V$. We want to find a subset X of vertices of G such that $\beta(\varphi)(X)$ is true and for each $v \in G$, $|X(v)| \in \alpha(v)$. We give an algorithm for finding such X running in time $n^{O(\nu)}$.

We fix integers $x_1, \ldots, x_\nu$ such that $x_1 + \cdots + x_\nu \leq n$, and our aim is to find a set X with $|X \cap T_j| = x_j$. Note that there are $n^{\mathcal{O}(\nu)}$ choices of such x_i. Then we test if a set X with $|X \cap T_j| = x_j$ satisfies $G \models \beta(\varphi)(X)$ and, for each global

constraint R_j whether $|X| \in R_j$ if $\beta(R_j) = \texttt{true}$ and $|X| \notin R_j$ if $\beta(R_j) = \texttt{false}$. Afterwards, we want to pick x_j vertices in each T_j in such a way that the local cardinality constraints are satisfied.

Note that if we have two sets X, X' of the same size and they differ only on some type T_j, then every local cardinality constraint of $v \notin T_j$ is satisfied for X if and only if it is satisfied for X'. This means that changes of X in one type cannot influence the validity of local cardinality constraints in other types. This allows us to pick x_i vertices in T_i separately in each type.

Let us now focus on type T_j. We set

$$c_j = \sum_{i:(i,j)\in E(T_G)} x_i \ .$$

If T_j is an independent type, then every vertex $v \in T_j$ has exactly c_j neighbors in X. Therefore, every vertex $v \in T_j$ must satisfy $c_j \in \alpha(v)$, otherwise there is no solution with $|X \cap T_j| = x_j$. If T_j is a clique type, every vertex in $v \in T_j \cap X$ has $c_j - 1$ neighbors in X, while every vertex $v \in T_j \setminus X$ has c_j neighbors in X. Let us partition the type T_j into the following sets:

$$\begin{aligned}
T_j^0 &= \{v \in T_j \mid c_j - 1 \notin \alpha(v), c_j \notin \alpha(v)\},\\
T_j^1 &= \{v \in T_j \mid c_j - 1 \notin \alpha(v), c_j \in \alpha(v)\},\\
T_j^2 &= \{v \in T_j \mid c_j - 1 \in \alpha(v), c_j \notin \alpha(v)\},\\
T_j^3 &= \{v \in T_j \mid c_j - 1 \in \alpha(v), c_j \in \alpha(v)\} \ .
\end{aligned}$$

From the previous, it follows that any $v \in T_j^0$ cannot lie in $T_j \cap X$ because $c_j - 1 \notin \alpha(v)$. Similarly, v cannot lie in $T_j \setminus X$, because $c_j \notin \alpha(v)$. Therefore, if T_j^0 is nonempty, there is no solution with $|X \cap T_j| = x_j$.

By similar reasoning, we see that every vertex in T_j^1 must be outside X and every vertex in T_j^2 must be in X. Combining these observations, we see that we can satisfy local cardinality constraints in T_j if and only if

- T_j^0 is empty, and
- $|T_j^2| \leq x_j \leq |T_j^2| + |T_j^3|$.

We now describe how to extend this idea when we have a formula with m free variables. Observe that it is enough to determine whether there is a satisfying assignment that obeys local constraints for a fixed signature; as there are at most $n^{2^m k}$ signatures, we can simply try all of them. Analogously for the global constraints.

Given a signature S, we check whether there is a satisfying assignment in the same manner as in the case of one free variable.

Let us set

$$x_j^i = \sum_{\substack{I\subseteq[m]\\ i\in I}} S(I,j), \text{and}$$

$$c_j^i = \sum_{\substack{j'\\ (j',j)\in E(T_G)}} x_j^i \ .$$

Those are analogous to x_j and c_j in the case of formula with one free variable. The value x_j^i denotes the $|\mu(X_i) \cap T_j|$. As before, if T_j is an independent type then every vertex $v \in T_j$ has exactly c_j^i neighbors in X_i.

We immediately see that if T_j is an independent type and there is a vertex v for which $c_j^i \notin \alpha_i(v)$, then we cannot fulfill the local cardinality constraint of v given the signature S.

For the case of a clique type, we refine the approach with sets $T_j^0, \ldots, T_j^3$. We define an auxiliary notion of *the kind of a vertex with respect to* X_i; we say that

- vertex v is of kind 0 if $c_j^i - 1 \notin \alpha_i(v)$ and $c_j^i \notin \alpha_i(v)$,
- vertex v is of kind 1 if $c_j^i - 1 \notin \alpha_i(v)$ and $c_j^i \in \alpha_i(v)$,
- vertex v is of kind 2 if $c_j^i - 1 \in \alpha_i(v)$ and $c_j^i \notin \alpha_i(v)$,
- vertex v is of kind 3 if $c_j^i - 1 \notin \alpha_i(v)$ and $c_j^i \in \alpha_i(v)$.

Finally, the *kind of a vertex* v is an m-tuple $(k_1, \ldots, k_m)$, where k_i is the kind of v with respect to X_i.

The kinds correspond to sets $T_j^0, \ldots, T_j^3$ from the previous case. By the same reasoning, we see that a vertex whose kind contains a 0 is a witness that we cannot satisfy the local cardinality constraints, a vertex of kind 1 with respect to X_i must be outside $\mu(X_i)$, and a vertex of kind 2 with respect to X_i must lie inside $\mu(X_i)$. We assume that there is no vertex whose kind contains 0 with respect to any X_i, as no solution exist in such case.

We subdivide each type T_j according to kinds. For a kind $K = (k_1, \ldots, k_m)$ we denote by T_j^K the set of all vertices of T_j of kind K.

We now describe a linear program that finds a satisfying assignment if one exists. For every $j \in [\nu]$, $I \subseteq [m]$, and $K \in [3]^m$, we introduce a variable $z_{I,j}^K$. The purpose of the variable $z_{I,j}^K$ is to represent the value

$$\left| \bigcap_{i \in I} \mu(X_i) \cap T_j^K \right| .$$

We add the following constraints:

$$\sum_{K \in [3]^m} z_{I,j}^K = S(I,j) \qquad \text{for every } I \subseteq [m] \text{ and every } j \in [\nu] \quad \text{(sc)}$$

$$\sum_{I \subseteq [m]} z_{I,j}^K = |T_j^K| \qquad \text{for every } K \in [3]^m \text{ and every } j \in [\nu] \quad \text{(kc)}$$

$$\sum_{\substack{I \subseteq [m] \\ i \in I}} z_{I,j}^K = 0 \qquad \text{for every } K \in [3]^m \text{ such that } k_i = 1 \text{ and every } j \in [\nu] \quad \text{(k1)}$$

$$\sum_{\substack{I \subseteq [m] \\ i \in I}} z_{I,j}^K = |T_j^K| \qquad \text{for every } K \in [3]^m \text{ such that } k_i = 2 \text{ and every } j \in [\nu] \quad \text{(k2)}$$

The constraints (sc) maintain "shape consistency"; that is, if we construct an assignment according to $z^K_{I,j}$, it will be of shape S. The constraints (kc) ensure "kind consistency". The constraint (k1) ensures that no vertex of kind 1 with respect to X_i lies in $\mu(X_i)$, while the constraint (k2) guarantees that all vertices of kind 2 with respect to X_i are in $\mu(X_i)$.

If there is no solution to the given ILP then no assignment of given shape obeys linear cardinality constraints. On the other hand, every solution of the given program describes an assignment of given shape that obeys the linear cardinality constraints.

The program has $3^m 2^m \nu$ variables and hence can be solved in FPT time with respect to $\beta(\varphi)$ and ν by Lenstra's algorithm [81]. Running the program $n^{2^m \nu}$ times, once for each signature, determines whether there is a satisfying assignment obeying the local cardinality constraints.

7. Parameterized Shifted Combinatorial Optimization

In this chapter we initiate the study of shifted combinatorial optimization from the perspective of parameterized complexity. In Section 7.1, we consider the case when the set S is given explicitely, and show that the complexity depends mainly on the objective function c. This leads to the complexity being P, FPT or XP with respect to the parameter $|S|$ when $-\mathbf{c} = \overline{\mathbf{c}}$, $\mathbf{c} = \overline{\mathbf{c}}$ or when neither holds, respectively. In Section 7.2, we consider the case when S is an MSO-definable set $S_\varphi(G)$ and G is a graph of bounded treewidth or cliquewidth. As a part of this investigation, we connect SCO to integer separable optimization over constructively decomposable polyhedra, which then allows us to use our results from Chapter 5. Moreover, it provides further insight into prior work on SCO. Finally, in Section 7.3 we show that the MSO PARTITIONING problem is W[1]-hard even when φ is an FO formula and the graph G has bounded treedepth.

7.1 Sets Given Explicitly

In this section we consider the shifted problem (1.2) over an explicitly given set $S = \{\mathbf{s}^1, \ldots, \mathbf{s}^m\}$. We demonstrate that already this seemingly simple case is in fact nontrivial and interesting. First, notice that with $S \subseteq \{0,1\}^n$ given explicitly the problem is generally NP-hard, which follows by the reduction from DOMINATING SET which we gave in the introduction (and give below). Moreover, it follows from known lower bounds on the DOMINATING SET problem that the brute-force algorithm which tries all possible r-subsets of S is likely close to optimal:

Proposition 7.1.1. *The SCO problem* (1.2) *is* NP-hard *for* 0/1 *shifted matrices* $\mathbf{c} = \overline{\mathbf{c}} \in \{0,1\}^{n\times r}$ *and explicitly given* 0/1 *sets* $S = \{\mathbf{s}^1, \ldots, \mathbf{s}^m\} \subseteq \{0,1\}^n$. *Moreover, unless the* Exponential Time Hypothesis *(ETH) fails, it cannot be solved in time* $n^{o(r)}$.

Proof. The NP-complete DOMINATING SET problem is to decide whether, given a graph $G = (V, E)$, there is a subset of vertices $D \subseteq V$ of size r such that every vertex $v \in V$ is either in D, or has a neighbor in D. Let $S = \{N[v] \mid v \in V\} \subseteq \{0,1\}^n$, where $N[v]$ is the characteristic vector of the closed neighborhood of v, i.e. including v itself, and let $c^1_i = 1$ for all i and $c^j_i = 0$ for all i and all $j \geq 2$. Then the optimal objective function value of (1.2) is n if and only if G has a dominating set of size r.

Moreover, Chen et al. [21] proved that unless ETH fails, there is no $n^{o(r)}$ algorithm solving *Dominating set*; thus, under the same assumption, there is no $n^{o(r)}$ algorithm solving SCO even when c is 0/1 and $\mathbf{c} = \overline{\mathbf{c}}$. □

Note that the next results in this section concerning Shifted IP apply to the more general situation in which S may consist of arbitrary integer vectors, not necessarily 0/1. This is formulated as follows.

Shifted Integer Programming. Given $S \subseteq \mathbb{Z}^n$ and $\mathbf{c} \in \mathbb{Z}^{n\times r}$, similarly to (1.2), solve

$$\max\{\mathbf{c}\overline{\mathbf{x}} \mid \mathbf{x} \in S^r\}\,. \tag{7.1}$$

For $S = \{\mathbf{s}^1, \ldots, \mathbf{s}^m\}$ and nonnegative integers $r_1, \ldots, r_m$ with $\sum_{i=1}^m r_i = r$, let $\mathbf{x}(r_1, \ldots, r_m)$ be the matrix in S^r with first r_1 columns equal to $\mathbf{s}^1$, next r_2 columns equal to $\mathbf{s}^2$, and so on, with last r_m columns equal to $\mathbf{s}^m$, and define $f(r_1, \ldots, r_m) = \mathbf{c}\overline{\mathbf{x}}(r_1, \ldots, r_m)$.

We have the following effective theorem in contrast with Proposition 7.1.1.

Theorem 7.1.2. *The shifted integer programming problem* (7.1) *over an explicitly given set* $S = \{\mathbf{s}^1, \ldots, \mathbf{s}^m\} \subseteq \mathbb{Z}^n$ *reduces to the following nonlinear integer programming problem over a simplex,*

$$\max\left\{ f(r_1, \ldots, r_m) \;\middle|\; r_1, \ldots, r_m \in \mathbb{N}\,,\ \sum_{k=1}^m r_k = r \right\}. \tag{7.2}$$

If $\mathbf{c} = \overline{\mathbf{c}}$ *is shifted then* f *is concave, and if* $-\mathbf{c}$ *is shifted then* f *is convex.*

Moreover, the following hold:

1. *With* m *parameter and* $\mathbf{c}$ *arbitrary, problem* (7.1) *is in* XP. *Furthermore, the problem is* W[1]-hard *with parameter* m *even for 0/1 sets* S.
2. *With* m *parameter and* $\mathbf{c}$ *shifted, problem* (7.1) *is in* FPT.
3. *With* m *variable and* $-\mathbf{c}$ *shifted, problem* (7.1) *is in* P.

Proof. Consider any $\mathbf{x} \in S^r$. For $k \in [m]$ let $r_k = |\{j \mid \mathbf{x}^j = \mathbf{s}^k\}|$ be the number of columns of $\mathbf{x}$ equal to $\mathbf{s}^k$. Then $\mathbf{x} \sim \mathbf{x}(r_1, \ldots, r_m)$ so $\overline{\mathbf{x}} = \overline{\mathbf{x}}(r_1, \ldots, r_m)$ and $\mathbf{c}\overline{\mathbf{x}} = f(r_1, \ldots, r_m)$. So an optimal solution $r_1, \ldots, r_m$ to (7.2) gives an optimal solution $\mathbf{x}(r_1, \ldots, r_m)$ to the shifted problem (7.1), proving the first statement.

We next show that if $\mathbf{c}$ is shifted then f is concave in the r_k variables. Suppose first that $n = 1$ so that $\mathbf{c}^1, \ldots, \mathbf{c}^r$ and $\mathbf{s}^1, \ldots, \mathbf{s}^m$ are scalars. For $k \in [m]$, define functions $g_k(r_1, \ldots, r_m) = \sum_{j=1}^k r_j$ which are linear in $r_1, \ldots, r_m$, and define a function h by $h(0) = 0$ and $h(l) = \sum_{j=1}^l \mathbf{c}^j$ for $l \in [r]$, which is concave since $\mathbf{c}^1 \geq \cdots \geq \mathbf{c}^r$.

Let π be a permutation of $[m]$ such that $\mathbf{s}^{\pi(1)} \geq \cdots \geq \mathbf{s}^{\pi(m)}$. Consider any $r_1, \ldots, r_m$ feasible in (7.2) and let $\mathbf{x} = \mathbf{x}(r_1, \ldots, r_m)$. Note that $\overline{\mathbf{x}}$ is the row vector with first $r_{\pi(1)}$ entries equal to $\mathbf{s}^{\pi(1)}$, next $r_{\pi(2)}$ entries equal to $\mathbf{s}^{\pi(2)}$, and so on, with last $r_{\pi(m)}$ entries equal to $\mathbf{s}^{\pi(m)}$. Let $g_k = g_k(r_{\pi(1)}, \ldots, r_{\pi(m)})$ for $k \in [m]$ and $t^k = \mathbf{s}^{\pi(k)} - \mathbf{s}^{\pi(k+1)} \geq 0$ for $k \in [m-1]$. Then we have that

$$\begin{aligned}
f(r_1, \ldots, r_m) &= \mathbf{c}\overline{\mathbf{x}} \\
&= \mathbf{s}^{\pi(1)} h(g_1) + \mathbf{s}^{\pi(2)} \left(h(g_2) - h(g_1)\right) + \cdots + \mathbf{s}^{\pi(m)} \left(h(g_m) - h(g_{m-1})\right) \\
&= \mathbf{t}^1 h(g_1) + \mathbf{t}^2 h(g_2) + \cdots + \mathbf{t}^{m-1} h(g_{m-1}) + \mathbf{s}^{\pi(m)} h(g_m) \\
&= \sum_{k=1}^{m-1} \mathbf{t}^k h\Big(g_k(r_{\pi(1)}, \ldots, r_{\pi(m)})\Big) + \mathbf{s}^{\pi(m)} \sum_{j=1}^{r} \mathbf{c}^j\,.
\end{aligned} \tag{7.3}$$

Now, g_k are linear functions of r_k, and h is concave, and so each composition $h(g_k(r_{\pi(1)}, \ldots, r_{\pi(m)}))$ is also concave. So $f(r_1, \ldots, r_m)$, which is a constant plus a nonnegative combination of concave functions, is a concave function of the r_k.

We continue with general n. Consider any $r_1, \ldots, r_m$ which are feasible in (7.2) and let $\mathbf{x} = \mathbf{x}(r_1, \ldots, r_m)$. For each $i \in [n]$ proceed as follows. Let $f_i(r_1, \ldots, r_m) = \mathbf{c}_i \overline{\mathbf{x}}_i$ with $\mathbf{c}_i$ the i-th row of $\mathbf{c}$ and $\overline{\mathbf{x}}_i$ the i-th row of the shift $\overline{\mathbf{x}}$. Let π_i be a permutation of $[m]$ such that $\mathbf{s}_i^{\pi_i(1)} \geq \cdots \geq \mathbf{s}_i^{\pi_i(m)}$. Repeating the above procedure with this 1-dimensional data we see that $f_i(r_1, \ldots, r_m)$ is concave. So $f(r_1, \ldots, r_m)$ is also concave in the r_k, being the following sum of concave functions,

$$f(r_1, \ldots, r_m) \;=\; \mathbf{c}\overline{\mathbf{x}} \;=\; \sum_{i=1}^{d} \mathbf{c}_i \overline{\mathbf{x}}_i \;=\; \sum_{i=1}^{d} f_i(r_1, \ldots, r_m) \;.$$

This also shows that if $-\mathbf{c}$ is shifted then $-f$ is concave and hence f is convex.

We proceed with the (positive) algorithmic statements of the theorem. For Point (1), which was also proved in [82], just note that for fixed m, there are $\mathcal{O}(r^{m-1})$ feasible solutions in (7.2), obtained by taking integers $0 \leq r_1, \ldots, r_{m-1} \leq r$ with $\sum_{i=1}^{m-1} r_i \leq r$ and setting $r_m = r - \sum_{i=1}^{m-1} r_i$. Hence, in polynomial time we can enumerate all, pick the best, and obtain an optimal solution $\mathbf{x}(r_1, \ldots, r_m)$ to the shifted problem (7.1).

For Point (2), if $\mathbf{c}$ is shifted, then, as just shown, f is concave. So the integer program (7.2) is to maximize a concave function with the number m of variables as a parameter. By known results on convex integer minimization, see [93], this problem is FPT and solvable in time $p(m)(\log r)^q$ for some computable function p of m and some constant q. Because it is enough to present the objective function by an oracle, we can extend this to the case when $\mathbf{c}$ is not given explicitly, but by a partial sums oracle $\gamma(i, j) = \sum_{\ell=1}^{j} c_i^\ell$, in which case r can be given in binary.

For Point (3), if $-\mathbf{c}$ is shifted, then, as just shown, f is convex. Therefore, the maximum in (7.2) is attained at a vertex of the simplex. These vertices are the m vectors $r\mathbf{e}_1, \ldots, r\mathbf{e}_m$, where $\mathbf{e}_k$ is the k-th unit vector in $\mathbb{R}^m$. Hence, in polynomial time we can pick the best vertex $r\mathbf{e}_k$ and obtain again an optimal solution $\mathbf{x}(r\mathbf{e}_k)$ to (7.1).

To complete the proof of the theorem, we return to the hardness claim in Point (1). We proceed by reduction from the Multidemand Set Multicover (MSM) problem, which we introduced in the previous chapter. We use it here in a slightly different formulation, which is nonetheless still hard by Corollary 6.3.4. Given is a universe $U = \{u_1, \ldots, u_n\}$, a collection of multidemands $\mathbf{d} = (d_1, \ldots, d_n)$ where $d_i \subseteq \mathbb{N}$ for $i \in [n]$, a covering set system $\mathcal{F} = \{F_1, \ldots, F_k\} \subseteq 2^U$, and an integer $r \in \mathbb{N}$. The goal is to find an integer partition $r = p_1 + \cdots + p_k$ such that, for all $i \in [n]$, we have $(\sum_{j:u_i \in F_j} p_j) \in d_i$. Corollary 6.3.4 shows that MSM is $\mathsf{W[1]\text{-}hard}$ with respect to the parameter n, even when $n = k$.

For clarity, we briefly and informally remark on a meaning of the MSM problem. We wish to take each set F_j, $j \in [k]$, with multiplicity p_j, and we demand that for each universe element u_i, $i \in [n]$, the total sum of multiplicities of sets u_i belongs to, falls into the constraint set d_i.

Given an instance $U, \mathbf{d}, \mathcal{F}$ and r of MSM, let $S \subseteq \{0,1\}^n$ be the set of characteristic vectors of $\mathcal{F}$, where $|S| = m$ in our case. We will define $\mathbf{c}$ inductively. Fix a row $i \in [n]$ and let $c_i^1 = 1$ if $1 \in d_i$ and $c_i^1 = 0$ otherwise. For $j \in [r-1]$, let $c_i^{j+1} = 1 - \sum_{\ell=1}^{j} c_i^\ell$ if $j+1 \in d_i$, and $c_i^{j+1} = -\sum_{\ell=1}^{j} c_i^\ell$ otherwise. Then $\max \mathbf{c}\overline{\mathbf{x}} \leq n$

and $\mathbf{c}\overline{\mathbf{x}} = n$ exactly when the multiplicities $p_1, \ldots, p_m$ of the vectors of S in $\mathbf{x} \in S^r$ are such that the number of 1's in each row i of $\overline{\mathbf{x}}$ falls into d_i, by our choice of $\mathbf{c}$. This is the case if and only if the MSM instance is a YES instance. □

In the rest of this section we provide several supplementary results related to the cases of Theorem 7.1.2.

Let us first give an exemplary application of Point (2) of Theorem 7.1.2 now. Bredereck et al. [17] study the WEIGHTED SET MULTICOVER (WSM) problem, which is as follows. Given a universe $U = \{u_1, \ldots, u_k\}$, integer demands $d_1, \ldots, d_k \in \mathbb{N}$ and a multiset $\mathcal{F} = \{F_1, \ldots, F_n\} \subseteq 2^U$ with weights $w_1, \ldots, w_n \in \mathbb{N}$, find a multiset $\mathcal{F}' \subseteq \mathcal{F}$ of smallest weight which satisfies all the demands – that is, for all $i \in [k]$, $|\{F \in \mathcal{F}' \mid u_i \in F\}| \geq d_i$. It is shown [17] that this problem is FPT when the size of the universe is a parameter, and then several applications in computational social choice are given. This problem is also solved implicitly in several papers [38, 46, 75].

Notice that $\mathcal{F}$ can be represented in a succinct way by viewing $\mathcal{F}$ as a set $\mathcal{F}_s = \{F_1, \ldots, F_K\}$ and representing the different copies of $F \in \mathcal{F}_s$ in $\mathcal{F}$ by defining K weight functions $w_1, \ldots, w_K$ such that, for each $i \in [K]$, $w_i(j)$ returns the total weight of the first j lightest copies of F_i, or ∞ if there are less than j copies. We call this the *succinct variant*.

Bredereck et al. [17] use Lenstra's algorithm for their result, which only works when $\mathcal{F}$ is given explicitly. We note in passing that our approach allows us to extend their result to the succinct case.

Proposition 7.1.3. WEIGHTED SET MULTICOVER *is in* FPT *with respect to universe size k, even in the succinct variant.*

Proof. Let $\mathbf{d}, \mathcal{F}, \mathbf{w}$ be an instance of WEIGHTED SET MULTICOVER with universe of size k, where k is parameter, and let $K = |\mathcal{F}_s| \leq 2^k$. We will construct an SCO instance with S of size $k + K$ such that solving it will correspond to solving the original WSM problem. Since $\max \mathbf{c}\overline{\mathbf{x}}$ with $c = \overline{\mathbf{c}}$ is equivalent to $\min \mathbf{c}\overline{\mathbf{x}}$ with $\mathbf{c} = -\overline{\mathbf{c}}$, we will define a minimization instance with $\mathbf{c}$ non-decreasing.

Let $S = \Big(\{(\mathbf{f}_i, \mathbf{e}_i) \mid F_i \in \mathcal{F}_s\} \cup \{(\mathbf{0}, \mathbf{0})\}\Big) \subseteq \{0,1\}^{k+K}$ where $\mathbf{f}_i$ is the characteristic vector of the set F_i and $\mathbf{e}_i$ is the i-th unit vector. Let W be the total weight of $\mathcal{F}$ and let $D = \sum d_i$ be the total demand. Then, the first k rows of $\mathbf{c}$ are defined as $c_i^j = -W$ if $j \leq d_i$ and $c_i^j = 0$ otherwise. The corresponding partial sums oracle is $\gamma(i, j) = -jW$ if $j \leq d_i$ and $\gamma(i, j) = -d_i W$ otherwise. The remaining K rows of $\mathbf{c}$ are exactly the weight functions $w_1, \ldots, w_K$, that is, $\gamma(k + i, j) = w_i(j)$, where w_i returns DW whenever it should return ∞.

Let $r = D$ and solve the SCO given above. A solution is represented by multiplicities r_i for $i \in [0, K]$, where r_0 is the multiplicity of the $\mathbf{0}$ vector. We interpret it as a WSM solution straightforwardly: r_i means how many copies of F_i we take to the solution, and we always choose the r_i lightest. Observe that if the objective is at most $-(D-1)W$, it means that the solution "hits" all the $-W$ items in the first k rows, which in turn means all demands are satisfied. On the other hand, the objective is more than $-(D-1)W$ only if some demand was not satisfied. Also, if $\min c\overline{\mathbf{x}} \leq -(D-1)W$, then the solution never hit a DW item in the last K rows, which in turn means that, for each $i \in [K]$, we have never used more copies of F_i than there actually are. Because the objective decomposes into

$-DW + \sum_{i=1}^{K} w_i(r_i)$, where the first term is a constant and the second is exactly the weight of the solution, we have found the optimum. □

Proposition 7.1.4. *If each demand d_i is an interval $[\ell_i, u_i]$,* Multidemand Set Multicover *is in* FPT *with respect to universe size k, even in the succinct variant.*

Proof. We could prove Proposition 7.1.4 along the lines of the previous proof, with a few tweaks. However, for more clarity and the benefit of a slightly different perspective, we will directly design a convex integer program in dimension K which solves it.

Let us describe the variables of our integer program. We have variables $x_1, \ldots, x_K$, one for each $F_i \in \mathcal{F}_s$. The variable x_i encodes how many copies of F_i we take into the solution. Then the program is simply

$$\min \sum_{i=1}^{K} w_i(x_i) \tag{7.4}$$

$$\ell_j \leq \sum_{i:j \in F_i} x_i \leq u_j \qquad \text{for each } i \in [k] \tag{7.5}$$

Clearly, the objective (7.4) is equal to the objective of the MSM problem. Moreover, the constraint (7.5) ensures that, for each $i \in [k] = U$, i is present in at least ℓ_i and at most u_i of the selected covering sets. The dimension of this program is $K \leq 2^k$. Applying for example the result of Oertel et al. [93]) that convex integer minimization is FPT with respect to the dimension concludes the proof. □

Theorem 7.1.2, Point (3), can be applied also to sets S presented implicitly by an oracle.

Definition 7.1.5 (Linear optimization oracle)**.** *A* linear optimization oracle *for $S \subseteq \mathbb{Z}^n$ is one that, queried on $\mathbf{w} \in \mathbb{Z}^n$, solves the linear optimization problem $\max\{\mathbf{ws} \mid \mathbf{s} \in S\}$. Namely, the oracle either asserts that the problem is infeasible, or unbounded, or provides an optimal solution.*

As mentioned before, even for $r = 2$, the shifted problem for perfect matchings is NP-hard, and hence for general $\mathbf{c}$ the shifted problem over S presented by a linear optimization oracle is also hard even for $r = 2$. In contrast, we have the following strengthening.

Theorem 7.1.6. *The shifted problem* (7.1) *with* $\mathbf{c}$ *nondecreasing, over any set $S \subset \mathbb{Z}^n$ which is presented by a linear optimization oracle, can be solved in polynomial time.*

Proof. Let $\mathbf{w} = \sum_{j=1}^{r} \mathbf{c}^j$ be the sum of the columns of $\mathbf{c}$, and query the linear optimization oracle of S on $\mathbf{w}$. If the oracle asserts that the problem is infeasible, then $S = \emptyset$ hence $S^r = \emptyset$ hence so is the shifted problem. Suppose it asserts that the problem is unbounded. Then for every real number q there is an $\mathbf{s} \in S$ with $\mathbf{ws} \geq q$. Then the matrix $\mathbf{x} = [\mathbf{s}, \ldots, \mathbf{s}]$ with all columns equal to $\mathbf{s}$ satisfies $\overline{\mathbf{x}} = \mathbf{x}$ and hence $\mathbf{c}\overline{\mathbf{x}} = \mathbf{cx} = \sum_{j=1}^{r} \mathbf{c}^j s = \mathbf{ws} \geq q$, and therefore the shifted problem is also unbounded.

Suppose then that the oracle returns an optimal solution $\mathbf{s}^* \in S$ and define $\mathbf{x}^* = [\mathbf{s}^*, \ldots, \mathbf{s}^*]$ to be the matrix with all columns equal to $\mathbf{s}^*$. We claim that $\mathbf{x}^*$ is an optimal solution to the shifted problem. Suppose indirectly $\mathbf{x}$ is a strictly better solution. Let T be the set of columns of $\mathbf{x}$, that means $T = \{\mathbf{x}^1, \ldots, \mathbf{x}^r\} = \{\mathbf{t}^1, \ldots, \mathbf{t}^m\}$ for suitable distinct $\mathbf{t}^k \in S$, where $k \in [m]$ and $m = |T| \leq r$.

Consider the shifted problem over T. By the proof of the algorithmic Point 3 of Theorem 7.1.2, we will have an optimal solution $\mathbf{y} = \mathbf{t}(r\mathbf{e}_k) = \mathbf{t}(0, \ldots, 0, r, 0 \ldots, 0)$ for some unit vector $\mathbf{e}_k \in \mathbb{R}^m$, that is, $\mathbf{y} = [\mathbf{t}, \ldots, \mathbf{t}]$ for some $\mathbf{t} \in T$. We then obtain

$$\mathbf{wt} \;=\; \mathbf{cy} \;=\; \mathbf{c}\overline{\mathbf{y}} \;\geq\; \mathbf{c}\overline{\mathbf{x}} \;>\; \mathbf{c}\overline{\mathbf{x}}^* \;=\; \mathbf{cx}^* \;=\; \mathbf{ws}^*$$

which is a contradiction to the assumed optimality of $\mathbf{s}^*$, completing the proof. □

7.2 MSO-definable Sets: XP for Bounded Treewidth

In this section we study another tractable and rich case of shifted combinatorial optimization, namely that of the set S defined in the MSO logic of graphs. This case, in particular, includes well studied MSO PARTITIONING problem of graphs (see below) which is tractable on graphs of bounded treewidth and cliquewidth. In the course of proving our results, it is useful to study a geometric connection of 0/1 SCO problems to separable optimization over decomposable polyhedra.

7.2.1 Relating SCO to Decomposable Polyhedra

The purpose of this subsection is to demonstrate how shifted optimization over 0/1 polytopes closely relates to the established concept of decomposable polyhedra.

Lemma 7.2.1. *Let $(S, \mathbf{c}, r)$ be an instance of shifted combinatorial optimization, with $S \subseteq \{0,1\}^n$, $r \in \mathbb{N}$ and $\mathbf{c} \in \mathbb{Z}^{n \times r}$. Let $P \subseteq [0,1]^n$ be a polytope such that $S = P \cap \{0,1\}^n$ and let $Q \subseteq [0,1]^{n+n'}$ be some extension of P, that is, $P = \{\mathbf{x} \mid (\mathbf{x}, \mathbf{y}) \in Q\}$.*

Then, provided a decomposition oracle for Q and an integer separable minimization oracle for rQ, the shifted problem given by $(S, \mathbf{c}, r)$ can be solved with one call to the optimization oracle and one call to the decomposition oracle. Furthermore, if $\mathbf{c}$ is shifted, an integer separable convex minimization oracle suffices.

To demonstrate Lemma 7.2.1 we use it to give an alternative proof of the result of Kaibel et al. [63] that the shifted problem is polynomial when $S = \{\mathbf{x} \in \{0,1\}^n \mid \mathbf{Ax} = \mathbf{b},\}$ and $\mathbf{A}$ is totally unimodular. It is known that $P = \{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b}, \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}$ is decomposable and a decomposition oracle is realizable in polynomial time [108]. Moreover, it is known that an integer separable convex minimization oracle for rP is realizable in polynomial time [60]. Lemma 7.2.1 implies that the shifted problem is polynomial for this S when $\mathbf{c}$ is shifted.

Another example is in fact our proof of Part 2 of Theorem 7.1.2. Observe that equation (7.2) is an extended formulation of a polytope rP with $P = \text{conv}(S)$.

Furthermore, we demonstrate that this extended formulation can be easily decomposed, and use a result about convex integer minimization to provide an integer separable convex minimization oracle.

The reason we have formulated Lemma 7.2.1 for S given by an extension Q of the polytope P corresponding to S, is expressed in Lemma 5.1.4: while P itself might not be decomposable, it always has a decomposable extension.

Other potential candidates where Lemma 7.2.1 could be applied are classes of polytopes that are either decomposable, or allow efficient integer separable (convex) minimization. Some known decomposable polyhedra are the stable set polytopes of perfect graphs, polyhedra defined by k-balanced matrices [118], polyhedra defined by nearly totally unimodular matrices [52], etc. Some known cases where integer separable convex minimization is polynomial are for $P = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \in \{0, 1, \ldots, r\}^n\}$ where the Graver basis of $\mathbf{A}$ has small size or when $\mathbf{A}$ is highly structured, namely when $\mathbf{A}$ is either an n-fold product, a transpose of it, or a 4-block n-fold product; see the books of Onn [95] and De Loera, Hemmecke and Köppe [84].

Now we return to the proof of Lemma 7.2.1.

Proof of Lemma 7.2.1. Fix $(\mathbf{x}, \mathbf{y}) \in rQ \cap \mathbb{Z}^{n+n'}$ and consider the set

$$Q^r_{(\mathbf{x},\mathbf{y})} = \left\{ (\mathbf{a}, \mathbf{b}) = ((\mathbf{a}^i, \mathbf{b}^i))_{i=1}^r \;\middle|\; \sum_{i=1}^r (\mathbf{a}^i, \mathbf{b}^i) = (\mathbf{x}, \mathbf{y}), (\mathbf{a}^i, \mathbf{b}^i) \in Q \cap \{0,1\}^{n+n'}, i \in [r] \right\}$$
$$\subseteq \left(Q \cap \{0,1\}^{n+n'} \right)^r$$

Note that $\mathbf{c}\overline{\mathbf{a}}$ is the same for all $(\mathbf{a}, \mathbf{b}) \in Q^r_{(\mathbf{x},\mathbf{y})}$, and observe that $(Q \cap \{0,1\}^{n+n'})^r = \bigcup_{(\mathbf{x},\mathbf{y}) \in (rQ \cap \{0,1\}^{n+n'})} Q^r_{(\mathbf{x},\mathbf{y})}$.

Consider now the objective function $\mathbf{c} \in \mathbb{Z}^{n \times r}$. Define $w_i : [0, r] \to \mathbb{Z}$ for $i \in [n]$ by $w_i(k) = \sum_{j=1}^k c_i^j$. Note that if $\mathbf{c}$ is shifted, every w_i is concave as it is a partial sum of a non-increasing sequence. Observe that $\mathbf{c}\overline{\mathbf{a}} = \sum_{i=1}^n w_i(\sum_{j=1}^r a_i^j) = \sum_{i=1}^n w_i(x_i)$. It follows that the minimum of $\sum_{i=1}^n w_i(x_i)$ over $rQ \cap \mathbb{Z}^{n+n'}$ equals the minimum of $\mathbf{c}\overline{\mathbf{a}}$ over $(Q \cap \{0,1\}^{n+n'})^r$.

To solve the shifted problem, let $f = \sum_{i=1}^{n+n'} f_i$ with $f_i = w_i$ for $i \in [n]$ and $f_i = 0$ for $i \in [n+1, n+n']$, and query the integer separable minimization oracle on rQ with $-f$ (minimizing $-f$ maximizes f). The oracle returns that the problem is either infeasible or unbounded or returns $(\mathbf{x}, \mathbf{y}) \in rQ$ maximizing f. Next, query the decomposition oracle for Q on r and $(\mathbf{x}, \mathbf{y})$ to obtain $((\mathbf{x}^i, \mathbf{y}^i))_{i=1}^r$, and return $((\mathbf{x}^i))_{i=1}^r$ as the solution. If $\mathbf{c}$ is shifted then $-f$ is convex and an integer separable convex minimization oracle suffices for the first step. □

7.2.2 XP Algorithm for MSO-definable Sets

The aforementioned MSO PARTITIONING problem on graphs comes as follows.

MSO PARTITIONING
Input: A graph G, an MSO_2 formula φ with one free vertex-set variable and an integer r.
Task: Find a partition $U_1 \dot\cup U_2 \ldots \dot\cup U_r = V(G)$ of the vertices of G such that $G \models \varphi(U_i)$ for all $i \in [r]$, or confirm that no such partition of $V(G)$ exists.

For example, if $\varphi(X)$ expresses that X is an independent set, then the φ-MSO PARTITIONING problem decides if G has an r-coloring, and thus, finding minimum feasible r (simply by trying $r = 1, 2, \dots$) solves the CHROMATIC NUMBER problem. Similarly, if $G \models \varphi(X)$ when X is a dominating set, minimizing r solves the DOMATIC NUMBER problem, and so on.

Rao [103] showed an algorithm for MSO PARTITIONING, for any MSO_2 formula φ, on a graph G with treewidth $tw(G) = \tau$ running in time $r^{f(\varphi,\tau)}n$ (XP) for some computable function f. Our next result widely generalizes this to SCO over MSO-definable sets.

Let $\mathbf{c}$ be defined as $c_i^1 = 1$ for $i \in [n]$ and $c_i^j = -1$ for $j \in [2, r]$ and $\in [n]$. Observe then the following: deciding whether the shifted problem with $S = S_\varphi(G)$, $\mathbf{c}$ and r, has an optimum of value n is equivalent to solving the MSO PARTITIONING problem for φ.

Theorem 7.2.2. *Let G be a graph of treewidth $tw(G) = \tau$, let φ be an MSO_2 formula and $S_\varphi(G) = \{\mathbf{x} \mid \mathbf{x}$ satisfies $\varphi\}$. There is an algorithm solving the shifted problem with $S = S_\varphi(G)$ and any given $\mathbf{c}$ and r in time $r^{f(\varphi,\tau)} \cdot |V(G)|$ for some computable function f. In other words, for parameters φ and τ, the problem is in the complexity class* XP.

Theorem 7.2.2 follows immediatelly by combining Lemma 7.2.1 with Theorem 5.2.13, with the slight difference that we have phrased Theorem 5.2.13 for MSO_1 over σ_2-structure, but this detail is insignificant.

Rao's result [103] applies also to the MSO PARTITIONING problem for MSO_1 formulas and graphs of bounded cliquewidth. We show the analogous extension of Theorem 7.2.2 next.

Corollary 7.2.3. *Let G be a graph of cliquewidth $cw(G) = \gamma$ given along with its γ-expression, let ψ be an MSO_1 formula and $S_\psi(G) = \{\mathbf{x} \mid \mathbf{x}$ satisfies $\psi\}$. There is an algorithm solving the shifted problem with $S = S_\psi(G)$ and any given $\mathbf{c}$ and r in time $r^{f(\psi,\gamma)} \cdot |V(G)|$ for some computable f.*

Proof. We invoke Lemma 4.5.2 to construct the tree T and formula φ, and then apply Theorem 7.2.2 to them (for a tree, $tw(T) = \tau = 1$, but note that φ now depends on both ψ and γ). Since $S_\psi(G)$ is essentially identical to $S_\varphi(T)$, up to the coordinates corresponding to $V(T) \setminus V(G)$ which are all zero by Lemma 4.5.2, the solution to the shifted problem with $S_\psi(G)$ is the same as the computed solution to the shifted problem with $S_\varphi(T)$. □

7.2.3 Applications

We have already discussed that SCO generalizes MSO PARTITIONING. We shall now discuss that also covering and packing problems can be expressed in SCO. In a covering problem, we require that every element is covered, but it can be possibly covered multiple times. In a packing problem, we require that that no element is used more than once. We can define a covering analogue of MSO PARTITIONING as follows:

MSO COVERING
Input: A graph G, an MSO_2 formula φ with one free vertex-set variable and an integer r.
Task: Find a covering $U_1, \ldots, U_r \subseteq V(G)$ with $\bigcup_{i=1}^{r} U_i = V(G)$ such that $G \models \varphi(U_i)$ for all $i \in [r]$, or confirm that no such covering of $V(G)$ exists.

A packing problem would be similar while requiring that the sets U_i are mutually disjoint and without the condition that $\bigcup U_i = V(G)$.

To express the covering problem in SCO, we set each coordinate of the first column of $\mathbf{c}$ to a 1, and all remaining coordinates to 0. If $\max \mathbf{c}\overline{\mathbf{x}} = n$, then $\mathbf{x}$ encodes a covering, while $\max \mathbf{c}\overline{\mathbf{x}} < n$ implies that at least one vertex is not covered. For a packing problem, we can similarly set the first column of $\mathbf{c}$ to all zeros, and all remaining coordinates to -1. Then, $\mathbf{c}\overline{\mathbf{x}} = 0$ if and only if $\mathbf{x}$ encodes a packing. Weighted versions are also possible using standard tricks such as coding lexicographical ordering by large numbers etc.

Thus, Theorem 7.2.2 implies that MSO_2 COVERING and MSO_2 PACKING are XP on graphs of bounded treewidth, and Corollary 7.2.3 gives a similar result for MSO_1 and graphs of bounded cliquewidth.

Mixed Cycle Cover and Mixed Chinese Postman. We will sketch an application of Theorem 7.2.2 to the MIXED CYCLE COVER problem (MCCP) and the related MIXED CHINESE POSTMAN problem (MCP). In MIXED CYCLE COVER, we are given a bridgeless strongly connected *mixed graph* $M = (V, E, A)$ with vertex set V, set of (undirected) edges E and a set of (directed) arcs A. The task is to find cycles $C_1, \ldots, C_r$ such that $\bigcup_{i=1}^{r} C_i = E \cup A$ while minimizing the sum of their weights $\sum_{i=1}^{r} |C_i|$. In MIXED CHINESE POSTMAN, we are given a strongly connected mixed graph M and the task is to find the smallest closed walk in M containing all edges and arcs. Fernandes et al. [37] proved that MCP and MCCP are XP parameterized by treewidth, and Gutin et al. [57] showed that MCP is W[1]-hard parameterized by treewidth.

Let $M = (V, E, A)$ be an instance of MCCP. As before, we note that to deal with mixed graphs in MSO, we need to use a different vocabulary, but that does not pose a significant problem. We preprocess M slightly to make the SCO formulation easier. For every edge $uv \in E$, we introduce two additional arcs (u, v) and (v, u), obtaining a mixed (multi)graph $M' = (V, E, A')$. We refer to the arcs in A as *original* and the arcs in $A' \setminus A$ as *added*. Then, we let $\varphi(C)$ be a formula ensuring that:

- $C \subseteq E \cup A'$,
- the elements of $C \cap A'$ form a directed cycle, i.e. $\forall (u, v) \in C \cap A' : \exists! w \in V : (v, w) \in C \cap A'$ and $C \cap A'$ is connected, and,
- $(u, v) \in C \cap (A' \setminus A) \Leftrightarrow uv \in C \cap E$, i.e. if C contains the arc (u, v) added because $uv \in E$, then include uv in C as well.[1]

Then, we solve SCO with $S = S_\varphi(M')$ and $\mathbf{c}$ defined as follows. Let $m = |E| + |A|$ and let N be a large number, e.g.$N \geq mr$. Each row corresponding to

[1]To be precise here, we would need a unary symbol (label) A to distinguish original and added arcs. Implementing this is straightforward.

an original arc a or to an edge e is $(N, -1, \ldots, -1)$, and each row corresponding to an added arc a is $(0, \ldots, 0)$. Then, $\max \mathbf{c}\overline{\mathbf{x}} \leq (m-1)N$ if and only if some original arc or edge is not covered. Moreover, when $\max \mathbf{c}\overline{\mathbf{x}} > (m-1)N$, then it is exactly $\max \mathbf{c}\overline{\mathbf{x}} = Nm - (\sum_{i=1}^{r} |C_i| - m)$ and thus the solution to SCO corresponds to a smallest covering of M with r cycles. Trying all $r \in [m]$ and picking the best result then does the job.

Mixed Chinese Postman. Fernandes et al. [37] also show that MCP can be cast as finding a smallest covering by cycles and pseudocycles; a *pseudocycle* is a cycle formed by using the same edge $e \in E$ twice. We note in passing that a slightly different preprocessing of M allows to formulate φ to also deal with pseudocycles. The rest is similar.

7.3 MSO-definable Sets: W[1]-hardness

Recall that natural hard graph problems such as Chromatic number are instances of MSO partitioning and so also instances of shifted combinatorial optimization. While we have shown an XP algorithm for SCO with MSO-definable sets on graphs of bounded treewidth and cliquewidth in Theorem 7.2.2 and Corollary 7.2.3, it is a natural question whether an FPT algorithm could exist for this problem, perhaps under a more restrictive width measure.

Here we give a strong negative answer to this question. First, we point out the result of Fomin et al. [42] proving W[1]-hardness of Chromatic number parameterized by the cliquewidth of the input graph. This immediately implies that an FPT algorithm in Corollary 7.2.3 would be very unlikely (cf. Section 2.1). Although, Chromatic number is special in the sense that it is solvable in FPT time when parameterized by the treewith of the input. Here we prove that it is not the case of MSO partitioning problems and SCO in general, even when considering restricted MSO_1 formulas and shifted $\mathbf{c}$, and parameterizing by the much more restrictive treedepth parameter.

Theorem 7.3.1. *There exists a graph* FO *formula $\varphi(X)$ with a free set variable X such that the instance of the* MSO partitioning *problem given by φ is* W[1]-hard *when parameterized by the treedepth of an input simple graph G.*

Consequently, the shifted problem with $S_\varphi(G)$ is also W[1]-hard *(for suitable shifted $\mathbf{c}$) when parameterized by the treedepth of G.*

We are going to prove Theorem 7.3.1 by a reduction from W[1]-hardness of Chromatic number with respect to cliquewidth [42]. As an intermediate step for our purpose, Gajarský et al. [46] prove that the graphs constructed for the reduction in [42], can be interpreted in a special way (formal details to follow) into labeled rooted trees of height 5, where the parameter is the number of labels. We, in turn, prove here that these labels can be traded for increased height of the tree and certain additional edges belonging to the tree closure. Consequently, the property of a set X of vertices to be independent in the original graph can now be expressed by a certain fixed formula $\varphi(X)$ (independent of the parameter) over a plain simple graph which is of bounded treedepth. So the MSO partitioning instance given by φ is indeed W[1]-hard when parameterized by the treedepth.

We start with formulating the needed special reformulation of the aforementioned result of Fomin et al [42] on hardness of Chromatic number.

Definition 7.3.2 (Tree-model [48]). *We say that a graph G has a* tree-model of m labels and depth d *if there exists a rooted tree T 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, and,*
4. *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.*

Let $\mathcal{TM}_m(d)$ denote the class of all graphs with a tree-model of m labels and depth d.

Theorem 7.3.3 ([46]). *The graphs constructed as the "hard" instances of* Chromatic number *in [42] belong to $\mathcal{TM}_m(5)$ where m is the considered parameter. Consequently, the* Chromatic number *problem considered on the classes $\mathcal{TM}_m(5)$ is* W[1]-hard *when parameterized by m.*

Proof of Theorem 7.3.1. We use a reduction from the instance described in Theorem 7.3.3. Let $G \in \mathcal{TM}_m(5)$ and r be an input of Chromatic number, i.e., the question is whether G is r-colorable. Let T be a tree-model of m labels and depth 5 of G. We are going to construct a formula φ and a graph H of treedepth at most $5m + 7$ such that $V(G) \subseteq V(H)$ and $X \subseteq V(G)$ is independent if and only if $H \models \varphi(X)$. Moreover, for $Y \subseteq V(H)$ such that $Y \not\subseteq V(G)$, it must hold $H \models \varphi(Y)$ if and only if $Y = V(H) \setminus V(G)$. Then, clearly, $(H, r + 1)$ will be a Yes instance of the MSO partitioning problem given by φ if, and only if, G is r-colorable. This would be the desired reduction.

The rest of the proof is devoted to the construction of φ and H. Let $M = [m]$ be the set of labels from Definition 7.3.2 and let *lab* map $V(G)$, the set of leaves of T, into M. There exist graphs $L_1, \ldots, L_5$ (self-loops allowed), each on the vertex set M, such that the following holds for any $u, v \in V(G)$ by Definition 7.3.2: $uv \in E(G)$ if and only if the least common ancestor of u, v in T is at distance $i \leq 5$ from u and $\{lab(u), lab(v)\} \in E(L_i)$.

A graph H_1 is constructed from T as follows:

- All vertices and edges of T are included in H_1 (the labels from T are ignored), the leaves of T have no label in H_1 while all the non-leaf nodes get a new label τ in H_1.
- For every non-leaf node $x \in V(T)$ at distance $i \leq 5$ from the leaves of T, a disjoint copy L^x of the graph L_i is created and added to H_1, such that the vertices of L^x receive the same (new) label λ and x is made adjacent to all vertices of L^x.
- Every leaf z of T, for $i = 1, \ldots, 5$, is connected by an edge in H_1 to the copy of the vertex $lab(z)$ in L^x, where x is the ancestor of z at distance i from z.

First of all, it is easy to see that H_1 is of treedepth at most $5m+6 = 1+5(m+1)$, since H_1 is contained in the closure of a tree obtained from T by "splitting" each non-leaf node x to a path on $m+1$ vertices forming the set $\{x\} \cup V(L^x)$ (which is of cardinality $m+1$). Second, we observe that the graph H_1 encodes the edges of G as follows: $(*)$ for $u, v \in V(G)$, we find the least common ancestor x of u and v among the τ-labeled vertices of H_1, and then we test whether there exist λ-labeled neighbors u', v' of x (and so $u', v' \in V(L^x)$) such that $uu', vv' \in E(H_1)$ and also $u'v' \in E(H_1)$.

Assume for now that the encoding $(*)$ of the edges of G is expressed in a binary predicate γ, such that $uv \in E(G) \iff H_1 \models \gamma(u, v)$. With γ, we can easily define a desired formula φ_1 such that $H_1 \models \varphi_1(X)$ if, and only if, $X \subseteq V(G)$ is independent in G or $X = V(H_1) \setminus V(G)$. It is

$$\begin{aligned}\varphi_1(X) \equiv \Big[\forall x \in X\Big(\neg\tau(x) \wedge \neg\lambda(x)\Big) \wedge \forall x, y \in X\Big(x = y \vee \neg\gamma(x, y)\Big)\Big] \\ \vee \forall x\Big((\tau(x) \vee \lambda(x)) \longleftrightarrow x \in X\Big).\end{aligned}$$

Note that γ does not depend on the original m labels of T.

The remaining two tasks are; to express γ in FO over H_1, and to "get rid of" possible self-loops and the labels τ, λ in H_1 by transforming φ_1 over H_1 into equivalent φ over simple unlabeled H. We finish these tasks as follows.

We recursively define $\alpha_0(x, y) \equiv (x = y)$ and, for $i = 1, \ldots, 5$, $\alpha_i(x, y) \equiv \tau(y) \wedge \exists z\Big(edge(z, y) \wedge \alpha_{i-1}(x, z)\Big)$. The meaning of $\alpha_i(x, y)$ is that y is an internal node of T at distance i from x (where x will be a leaf of T but this is not enforced by α_i). The above encoding $(*)$ of the edges of G into H_1 can now be literally expressed as

$$\begin{aligned}\gamma(u, v) \equiv\ \exists x \Big[\tau(x) \wedge \Big(\textstyle\bigvee_{i=1}^{5} \alpha_i(u, x) \wedge \alpha_i(v, x) \wedge \neg\exists x'\Big(\alpha_{i-1}(u, x') \wedge \alpha_{i-1}(v, x')\Big)\Big) \\ \wedge \exists u', v'\Big(\lambda(u') \wedge \lambda(v') \wedge edge(x, u') \wedge edge(x, v') \\ \wedge edge(u, u') \wedge edge(v, v') \wedge edge(u', v')\Big)\Big],\end{aligned}$$

which is an FO formula independent of H_1 and given T.

Lastly, we observe that H_1 has no vertices of degree 1. We hence construct H from H_1 by adding one new degree-1 neighbor to every τ-labeled vertex of H_1, adding two new degree-1 neighbors to every λ-labeled vertex of H_1 without self-loop, adding three new degree-1 neighbors to every λ-labeled vertex of H_1 with self-loop, and removing all the loops. The resulting simple graph H is of treedepth at most $5m + 7$ (in fact, again $\leq 5m + 6$), and one can identify the original vertices of H_1 as those having degree > 1 in H. The labels τ, λ and the self-loops of H_1 (as used in the formula φ_1) can be routinely interpreted by FO formulas, e.g., $\tau(x) \equiv \neg\delta_1(x) \wedge \exists y\Big(\delta_1(y) \wedge edge(x, y)\Big) \wedge \forall y, y'\Big[\Big(\delta_1(y) \wedge edge(x, y) \wedge \delta_1(y') \wedge edge(x, y')\Big) \rightarrow y = y'\Big]$, where $\delta_1(y) \equiv \forall z, z'\Big[\Big(edge(y, z) \wedge edge(y, z')\Big) \rightarrow z = z'\Big]$. Such an interpretation defines desired φ from φ_1. $\square$

7.3.1 Remarks on Hardness

Convex versus Separable Convex Optimization

Separable convex optimization is, by definition, a special case of convex optimization. However, it is helpful to our intuition to have some tangible evidence that general convex optimization truly is harder than separable convex optimization. One such piece of evidence is exhibited by the hardness of Theorem 7.1.1.

As we have shown in Theorem 4.5.4, the MSO polytope $P_\varphi(G)$ can be obtained as a projection of a totally unimodular (TU) system defining a polytope $P = P_{s-t}(D)$ of s–t dipaths in a certain directed graph D. We have already mentioned that integer separable convex minimization is polynomial in TU systems [60]. Thus it is tempting to think that integer separable convex minimization should be realizible in FPT time for $P_\varphi(G)$ as well. However, this argument breaks down with the projection $\pi : P_{s-t}(D) \to P_\varphi(G)$, because adding π to the TU system $\mathbf{Ax} = \mathbf{b}$ which defines $P_{s-t}(D)$ makes it *non*-TU.

Since pathwidth is a more general parameter than treedepth, Theorem 7.1.1 shows that there likely does not exist *any* projection from $P_{s-t}(D)$ to $P_\varphi(G)$ which would preserve total unimodularity. Also, it shows that while integer separable convex minimization is in P for TU systems, this is unlikely for general integer convex minimization.

The Sets S^r and rS

A central object of SCO is the set S^r of r-tuples of elements of S. Clearly S^r is related to the set $\left(\!\binom{S}{r}\!\right)$ of r-multisubsets of S. Let $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^r\} \in \left(\!\binom{S}{r}\!\right)$. The key difference is that S^r contains all $r!$ orderings of X as individual elements, while $\left(\!\binom{S}{r}\!\right)$ does not. However, the shift operation disregards the order of X anyway, and thus SCO over S most closely resembles optimization over $\left(\!\binom{S}{r}\!\right)$ rather than S^r. We note that linear optimization over S^r is as hard as linear optimization over S, since it can be split into r separate linear optimization tasks over S.

SCO can be seen as a form of nonlinear optimization, since we are doing linear minimization after applying the *nonlinear* shift operator. It is natural to ask how difficult is the even more general problem

Given $S \subseteq \{0,1\}^n$ and a function $f : \{0,1\}^{n \times r} \to \mathbb{N}$, solve

$$\max\{f(\mathbf{x}) \mid \mathbf{x} \in S^r\}$$

in the case of $S = S_\varphi(G)$. Theorem 7.1.1 allows us to show that even for graphs of treewidth 2 and relatively well-behaved functions f, this problem is NP-hard. We merely sketch the proof here.

The reduction is from the $L(2,1)$-COLORING problem, where we seek to color a graph G with numbers from 1 to λ such that the color of neighbors differs by at least 2, and the color of distance-2 neighbors differs by at least 1. Fiala et al. [39] prove that this problem is NP-hard already on graphs of treewidth 2. By G^2 we denote the *square of* G, which is the graph obtained from G by connecting every pair of vertices in distance 2. Note that any color class in an $L(2,1)$-coloring is an independent set in G^2. Let S be the set of indicators of independent sets of G^2; it is not too difficult to see that this set S is MSO_1-definable.

The remaining task then is to formulate the objective function f such that it penalizes if two consecutive colors contain neighboring vertices and if $\mathbf{x}$ is not a partition (some vertex contained more or less than once). Fix a $\lambda \in \mathbb{N}$. Then $\mathbf{x}$ is a valid $L(2,1)$-coloring if and only if $f(\mathbf{x}) = 0$ with

$$f(\mathbf{x}) = \Big(\sum_{i=1}^{\lambda} \sum_{uv \in E} -x_u^{i-1} x_v^i - x_v^{i-1} x_u^i \Big) + \Big(\sum_{u \in V} -|1 - \sum_{i=1}^{\lambda} x_u^i| \Big) \ .$$

8. Conclusion and Open Problems

We have provided extended formulations of polytopes associated with the constraint satisfaction problem (CSP) and the monadic second order logic on graphs (MSO) when given instances have bounded treewidth. Then, we have shown that our extended formulations have various additional properties, and we have used these to provide compact extended formulations and algorithmic applications for extensions of MSO and problems expressible in these extensions. We have also complemented our findings by matching hardness results, explored other graph width parameters like cliquewidth and neighborhood diversity, and also introduced the broad shifted combinatorial optimization framework.

Of course, interesting and important open problems and research directions regarding these topics remain. We close with listing a few of them, divided according to the considered topics.

8.1 Constraint Satisfaction Problem

We find it interesting that the size of the extended formulation we derive matches the best known runtime of the algorithm for CSP (Theorem 2.4.4) and the best possible runtimes (assuming Strong ETH) of the algorithms for Vertex Cover, Independent Set etc. Is it possible to prove a matching lower bound to our extended formulation? Or, on the other hand, what is a natural problem whose time complexity does not match the extension complexity of its natural polytope when parameterized by treewidth? The question phrased like this has a trivial answer: the matching polytope has exponential extension complexity (Rothvoss [105]). We might pose a refinement of the question: which NP-hard problem is FPT parameterized by treewidth but its natural polytope does not have an FPT-size extended formulation?

In Subsection 4.5.3 we have sketched how to construct an extended formulation of the MSO polytope for bounded pathwidth graphs. Due to the similarities in our constructions of the CSP and MSO polytopes which we discuss in Chapter 5 it is reasonable to expect that the same argument goes through for the CSP polytope. The real question is if it can be extended to bounded treewidth graphs. We suspect it could: our approach now is constructing a digraph whose s–t paths essentially correspond to computations of an automaton on a string. What we need is to capture computations of a tree automaton. However, it should be possible to "serialize" even such computations.

8.2 Algorithmic Metatheorems

Limits of MSO extensions, other logics and metatheorems. We have defined extensions of MSO and extended positive and negative results for them. There is still some unexplored space in MSO extensions. For example, in our extensions we do not allow the constrained variables to be quantified. Szeider [111]

shows that $\mathsf{MSO}^{\mathsf{L}}$ where some of the constrained sets are quantified is NP-hard already on graphs of treewidth 2. We are not aware of a comparable result for $\mathsf{MSO}^{\mathsf{G}}$, and no results of this kind are known for graphs of bounded neighborhood diversity. We have sketched in Subsection 6.2.1 that our results on $\mathsf{MSO}^{\mathsf{GL}}$ parameterized by $\mathrm{tw}(G)$ can be strenghtened in some ways.

In that regard, we believe that a great merit of algorithmic metatheorems is in generalizing existing results. Many problems [47, 38] are FPT parameterized by $\mathrm{nd}(G)$ but are not expressible in any of the studied logics. Similarly, there are still a few problems which are W[1]-hard and XP parameterized by $\mathrm{tw}(G)$ but not expressible in any of the studied logics. So we ask for a metatheorem generalizing as many such positive results as possible. We believe that a promising direction is studying and extending the MSO PARTITIONING problem, as done by Ganian and Obdržálek [49] or us in Chapter 7.

Also, we have not explored other logics, as for example the modal logic considered by Pilipczuk [97], although his aim is single-exponential complexity rather than large expressivity.

Complementary Parameters and Problems. Unlike for treewidth, taking the complement of a graph preserves its neighborhood diversity. Thus our results apply also in the complementary setting, where, given a graph G and a parameter $p(G)$, we are interested in the complexity (with respect to $p(G)$) of deciding a problem P on the *complement* of G. While the complexity remains the same when parameterizing by neighborhood diversity, it is unclear for sparse graph parameters such as treewidth. It was shown very recently [32] that the HAMILTONIAN PATH problem admits an FPT algorithm with respect to the treewidth of the complement of the graph. This suggest that at least sometimes this is the case and some extension of Courcelle's theorem deciding properties of the complement may hold.

8.3 Shifted Combinatorial Optimization

Further uses of Lemma 7.2.1. For example, which interesting combinatorial sets S can be represented as n-fold integer programs [84, 95] such that the corresponding polyhedra are decomposable?

Parameterizing by the number of selected elements r. It is interesting to consider taking r as a parameter. For example, Fluschnik et al. [41] prove that the MINIMUM SHARED EDGES problem is FPT parameterized by the number of paths. Omran et al. [94] prove that the MINIMUM VULNERABILITY problem is in XP with the same parameter. Since both problems are particular cases of the shifted problem, we ask whether the shifted problem with S being the set of s–t paths of a (di)graph lies in XP or is NP-hard already for some constant r. We know that this does not generalize to S given by a totally unimodular system by the recent result of Koutecký et al. [73] who prove that such a problem is NP-hard already when S is the set of matchings of a bipartite graph and $r = 2$.

Approximation. The MINIMUM VULNERABILITY problem has also been studied from the perspective of approximation algorithms [94]. The study of approxi-

mation algorithms for shifted combinatorial optimization has been initiated only very recently [73], and we suspect more to be provable.

Going beyond 0/1**.** The results on Shifted Integer Programming in Section 7.1 are the only known ones in which S does not have to be 0/1. What can be said about the shifted problem with such sets S that are not given explicitly, e.g., when S is given by a totally unimodular system?

Bibliography

[1] Sancrey Rodrigues Alves, Konrad Kazimierz Dabrowski, Luérbio Faria, Sulamita Klein, Ignasi Sau, and Uéverton dos Santos Souza. On the (parameterized) complexity of recognizing well-covered (r, l)-graphs. In *10th International Conference on Combinatorial Optimization and Applications (COCOA'16)*, volume 10043 of *Lecture Notes in Computer Science*, pages 423–437, 2016.

[2] NR Aravind, Subrahmanyam Kalyanasundaram, Anjeneya Swami Kare, and Juho Lauri. Algorithms and hardness results for happy coloring problems. *arXiv preprint arXiv:1705.08282*, 2017.

[3] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, June 1991.

[4] Sepehr Assadi, Ehsan Emamjomeh-Zadeh, Ashkan Norouzi-Fard, Sadra Yazdanbod, and Hamid Zarrabi-Zadeh. The minimum vulnerability problem. *Algorithmica*, 70(4):718–731, 2014.

[5] Giorgio Ausiello, Pierluigi Creczenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation; Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[6] David Avis and Hans Raj Tiwary. On the extension complexity of combinatorial polytopes. In *Annual International Colloquium on Automata, Languages and Programming (ICALP'13)*, pages 57–68, 2013.

[7] Stephen Baum and Leslie E. Trotter Jr. Integer rounding and polyhedral decomposition for totally unimodular systems. In *Optimization and Operations Research*, pages 15–23. Springer, 1978.

[8] Daniel Bienstock and Gonzalo Munoz. Lp approximations to mixed-integer polynomial optimization problems. *arXiv preprint arXiv:1501.00288*, 2015.

[9] Daniel Bienstock and Nuri Özbay. Tree-width and the sherali-adams operator. *Discrete Optimization*, 1(1):13–21, 2004.

[10] Bernhard Bliem, Benjamin Kaufmann, Torsten Schaub, and Stefan Woltran. ASP for anytime dynamic programming on tree decompositions. In *25th International Joint Conference on Artificial Intelligence (IJCAI'16), New York, NY, USA, 9-15 July 2016*, pages 979–986, 2016.

[11] Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Annual ACM Symposium on Theory of Computing (STOC'93)*, pages 226–234, 1993.

[12] Hans L. Bodlaender. Treewidth: characterizations, applications, and computations. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG'06)*, volume 4271 of *Lecture Notes in Computer Science*, pages 1–14, 2006.

[13] Édouard Bonnet and Florian Sikora. The graph motif problem parameterized by the structure of the input graph. *Discrete Applied Mathematics*, 2016.

[14] Gábor Braun, Samuel Fiorini, Sebastian Pokutta, and David Steurer. Approximation limits of linear programs (beyond hierarchies). *Math. Oper. Res.*, 40(3):756–772, 2015.

[15] Gábor Braun, Rahul Jain, Troy Lee, and Sebastian Pokutta. Information-theoretic approximations of the nonnegative rank. *Computational Complexity*, 26(1):147–197, 2017.

[16] Gábor Braun, Sebastian Pokutta, and Aurko Roy. Strong reductions for extended formulations. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO'16)*, pages 350–361. Springer, 2016.

[17] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, Piotr Skowron, and Nimrod Talmon. Elections with few candidates: Prices, weights, and covering problems. In *International Conference on Algorithmic Decision Theory (ADT'15)*, volume 9346 of *Lecture Notes in Computer Science*, pages 414–431, 2015.

[18] Austin Buchanan and Segiy Butenko. Tight extended formulations for independent set, 2014. Available on Optimization Online.

[19] Andrei A Bulatov. A dichotomy theorem for nonuniform csps. *arXiv preprint arXiv:1703.03021*, 2017.

[20] Siu On Chan, James R. Lee, Prasad Raghavendra, and David Steurer. Approximate constraint satisfaction requires large LP relaxations. In *54th Annual IEEE Symposium on Foundations of Computer Science, (FOCS'13)*, pages 350–359, 2013.

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

[22] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Extended formulations in combinatorial optimization. *Annals of Operations Research*, 204(1):97–143, 2013.

[23] Michele Conforti and Kanstantsin Pashkovich. The projected faces property and polyhedral relations. *Mathematical Programming*, pages 1–12, 2015.

[24] Bruno Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[25] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[26] Bruno Courcelle and Mohamed Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109(1–2):49–82, 1 March 1993.

[27] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77–114, 2000.

[28] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

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

[30] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[31] Pavel Dvořák, Dušan Knop, and Tomáš Toufar. Target set selection in dense graph classes. *arXiv preprint arXiv:1610.07530*, 2016.

[32] Pavel Dvořák, Dušan Knop, and Tomáš Masařík. Anti-path cover on sparse graph classes. In *11th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, (MEMICS'16)*, volume 233 of *Electronic Proceedings in Theoretical Computer Science*, pages 82–86, 2016.

[33] Yuri Faenza, Samuel Fiorini, Roland Grappe, and Hans Raj Tiwary. Extended formulations, nonnegative factorizations, and randomized com. protocols. *Math. Program.*, 153(1):75–94, 2015.

[34] Feder and Vardi. Monotone monadic SNP and constraint satisfaction. In *Annual ACM Symposium on Theory of Computing (STOC'93)*, 1993.

[35] Tomás Feder and Pavol Hell. List homomorphisms to reflexive graphs. *J. Comb. Theory, Ser. B*, 72(2):236–250, 1998.

[36] Michael R. Fellows, Guillaume Fertin, Danny Hermelin, and Stéphane Vialette. Upper and lower bounds for finding connected motifs in vertex-colored graphs. *J. Comput. Syst. Sci*, 77(4):799–811, 2011.

[37] Cristina G Fernandes, Orlando Lee, and Yoshiko Wakabayashi. Minimum cycle cover and chinese postman problems on mixed graphs with bounded tree-width. *Discrete Applied Mathematics*, 157(2):272–279, 2009.

[38] Jiří Fiala, Tomáš Gavenčiak, Dušan Knop, Martin Koutecký, and Jan Kratochvíl. Parameterized complexity of distance labeling and uniform channel assignment problems. *Discrete Applied Mathematics*, pages –, 2017.

[39] Jiří Fiala, Petr Golovach, and Jan Kratochvíl. Distance constrained labelings of graphs of bounded treewidth. In *Annual International Colloquium on Automata, Languages and Programming (ICALP'05)*, 2005.

[40] Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. Exponential lower bounds for polytopes in combinatorial optimization. *J. ACM*, 62(2):17, 2015.

[41] Till Fluschnik, Stefan Kratsch, Rolf Niedermeier, and Manuel Sorge. The parameterized complexity of the minimum shared edges problem. In *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'15)*, volume 45; 45 of *LIPIcs*, pages 448–462, 2015.

[42] Fedor Fomin, Petr Golovach, Daniel Lokshtanov, and Saket Saurab. Clique-width: On the price of generality. In *Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'09)*, pages 825–834, 2009.

[43] Eugene C. Freuder. Complexity of K-tree structured constraint satisfaction problems. In *Proc. of the 8th National Conference on Artificial Intelligence*, pages 4–9, 1990.

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

[45] Jakub Gajarskỳ, Petr Hliněnỳ, Martin Kouteckỳ, and Shmuel Onn. Parameterized shifted combinatorial optimization. *arXiv preprint arXiv:1702.06844*, 2017. To appear at the 23rd Annual International Computing and Combinatorics Conference (COCOON'17).

[46] Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In *International Symposium on Parameterized and Exact Computation (IPEC'13)*, volume 8246 of *Lecture Notes in Computer Science*, pages 163–176, 2013.

[47] Robert Ganian. Using neighborhood diversity to solve hard problems. *arXiv preprint arXiv:1201.3091*, 2012.

[48] Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, Patrice Ossona de Mendez, and Reshma Ramadurai. When trees grow low: Shrubs and fast MSO_1. In *37th International Symposium on the Mathematical Foundations of Computer Science (MFCS'12)*, volume 7464 of *Lecture Notes in Computer Science*, pages 419–430, 2012.

[49] Robert Ganian and Jan Obdržálek. Expanding the expressive power of monadic second-order logic on restricted graph classes. In *24th International Workshop on Combinatorial Algorithms (IWOCA'13)*, volume 8288 of *Lecture Notes in Computer Science*, pages 164–177, 2013.

[50] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Combining Treewidth and Backdoors for CSP. In *34th Symposium on Theoretical Aspects of Computer Science (STACS'17)*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:17, Dagstuhl, Germany, 2017.

[51] Luisa Gargano and Adele A. Rescigno. Complexity of conflict-free colorings of graphs. *Theoretical Computer Science*, 566(??):39–49, February 2015.

[52] Dion Gijswijt. Integer decomposition for polyhedra defined by nearly totally unimodular matrices. *SIAM Journal on Discrete Mathematics*, 19(3):798–806, 2005.

[53] Georg Gottlob, Reinhard Pichler, and Fang Wei. Monadic datalog over finite structures with bounded treewidth. In *26th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'07)*, pages 165–174, 2007.

[54] Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta theorems. *Model Theoretic Methods in Finite Combinatorics*, 558:181–206, 2011.

[55] Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In *33rd Annual ACM Symposium on Theory of Computing (STOC'01)*, pages 657–666, 2001.

[56] Branko Grünbaum. *Convex Polytopes*. Wiley Interscience Publ., London, 1967.

[57] Gregory Gutin, Mark Jones, and Magnus Wahlström. Structural parameterizations of the mixed chinese postman problem. In *23rd European Symposium on Algorithms (ESA'15)*, pages 668–679. 2015.

[58] Pavol Hell and Jaroslav Nešetřil. Colouring, constraint satisfaction, and complexity. *Computer Science Review*, 2(3):143–163, 2008.

[59] Petr Hliněný and Sang-il Oum. Finding branch-decompositions and rank-decompositions. *SIAM J. Comput.*, 38(3):1012–1032, 2008.

[60] Dorit S. Hochbaum and J. George Shanthikumar. Convex separable optimization is not much harder than linear optimization. *J. ACM*, 37(4):843–862, October 1990.

[61] Volker Kaibel. Extended formulations in combinatorial optimization. *Optima*, 85:2–7, 2011.

[62] Volker Kaibel and Andreas Loos. Branched polyhedral systems. In *14th International Conference on Integer Programming and Combinatorial Optimization (IPCO'10)*, volume 6080 of *Lecture Notes in Computer Science*, pages 177–190, 2010.

[63] Volker Kaibel, Shmuel Onn, and Pauline Sarrabezolles. The unimodular intersection problem. *Oper. Res. Lett*, 43(6):592–594, 2015.

[64] Volker Kaibel and Kanstantsin Pashkovich. Constructing extended formulations from reflection relations. In *15th International Conference on Integer Programming and Combinatorial Optimization (IPCO'11)*, volume 6655 of *Lecture Notes in Computer Science*, pages 287–300, 2011.

[65] Subhash Khot. On the power of unique 2-Prover 1-Round games. In *34th Annual ACM Symposium on Theory of Computing (STOC'02)*, pages 767–775, 2002.

[66] Ton Kloks. *Treewidth: Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. 1994.

[67] Joachim Kneis, Alexander Langer, and Peter Rossmanith. Courcelle's theorem - A game-theoretic approach. *Discrete Optimization*, 8(4):568–594, 2011.

[68] Dušan Knop, Martin Koutecký, Tomáš Masařík, and Tomáš Toufar. Simplified algorithmic metatheorems beyond mso: Treewidth and neighborhood diversity. *arXiv preprint arXiv:1703.00544*, 2017. 43rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'17).

[69] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61(2):302 – 332, 2000.

[70] Petr Kolman and Martin Koutecký. Extended formulation for csp that is compact for instances of bounded treewidth. *The Electronic Journal of Combinatorics*, 22(4):P4–30, 2015.

[71] Petr Kolman, Martin Koutecký, and Hans Raj Tiwary. Extension complexity, mso logic, and treewidth. In *15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT'16)*, 2016.

[72] Petr Kolman, Bernard Lidický, and Jean-Sébastien Sereni. On Fair Edge Deletion Problems, 2009.

[73] Martin Koutecky, Asaf Levin, Syed M Meesum, and Shmuel Onn. Approximate shifted combinatorial optimization. *arXiv preprint arXiv:1706.02075*, 2017.

[74] Stephan Kreutzer. Algorithmic meta-theorems. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:147, 2009. Appeared at IWPEC 2008.

[75] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[76] Michael Lampis. Model checking lower bounds for simple graphs. *Logical Methods in Computer Science*, 10(1), 2014.

[77] Alexander Langer, Felix Reidl, Peter Rossmanith, and Somnath Sikdar. Evaluation of an MSO-solver. In *14th Meeting on Algorithm Engineering & Experiments, (ALENEX'12)*, pages 55–63, 2012.

[78] Alexander Langer, Felix Reidl, Peter Rossmanith, and Somnath Sikdar. Practical algorithms for MSO model-checking on tree-decomposable graphs. *Computer Science Review*, 13-14:39–74, 2014.

[79] Monique Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.

[80] James R. Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Annual ACM Symposium on Theory of Computing (STOC'15)*, pages 567–576, 2015.

[81] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, pages 538–548, 1983.

[82] Asaf Levin and Shmuel Onn. Shifted matroid optimization. *Oper. Res. Lett*, 44:535–539, 2016.

[83] Leonid Libkin. *Elements of Finite Model Theory*. Springer-Verlag, Berlin, 2004.

[84] Jesús A. De Loera, Raymond Hemmecke, and Matthias Köppe. *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*, volume 14 of *MOS-SIAM Series on Optimization*. SIAM, 2013.

[85] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In *22nd Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'11)*, pages 777–789, 2011.

[86] Johann A. Makowsky. Algorithmic uses of the feferman-vaught theorem. *Ann. Pure Appl. Logic*, 126(1-3):159–213, 2004.

[87] François Margot. *Composition de polytopes combinatoires: une approche par projection*. PhD thesis, École polytechnique fédérale de Lausanne, 1994.

[88] R. Kipp Martin, Ronald L. Rardin, and Brian A. Campbell. Polyhedral characterization of discrete dynamic programming. *Oper. Res.*, 38(1):127–138, February 1990.

[89] Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010.

[90] Tomáš Masařík and Tomáš Toufar. Parameterized complexity of fair deletion problems. In *International Conference on Theory and Applications of Models of Computation (TAMC'16)*, pages 628–642, 2017.

[91] Jaroslav Nešetřil and Patrice Ossona De Mendez. *Sparsity: graphs, structures, and algorithms*, volume 28. Springer Science & Business Media, 2012.

[92] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[93] Timm Oertel, Christian Wagner, and Robert Weismantel. Integer convex minimization by mixed integer linear optimization. *Oper. Res. Lett*, 42(6-7):424–428, 2014.

[94] Masoud T. Omran, Jörg-Rüdiger Sack, and Hamid Zarrabi-Zadeh. Finding paths with minimum shared edges. *J. Comb. Optim*, 26(4):709–722, 2013.

[95] Shmuel Onn. *Nonlinear Discrete Optimization*. Zurich Lectures in Advanced Mathematics. European Mathematical Society, 2010. available online at: `http://ie.technion.ac.il/~onn/Book/NDO.pdf`.

[96] Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

[97] Michal Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In *36th International Symposium on the Mathematical Foundations of Computer Science (MFCS'11)*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531, 2011.

[98] Gabriele Puppis. *Automata for Branching and Layered Temporal Structures: an investigation into regularities of infinite transition systems*, volume 5955. Springer Science & Business Media, 2010.

[99] Arash Rafiey, Jeff Kinne, and Tomás Feder. Dichotomy for digraph homomorphism problems. *arXiv preprint arXiv:1701.02409*, 2017.

[100] Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *40th Annual ACM Symposium on Theory of Computing (STOC'08)*, pages 245–254, 2008.

[101] Prasad Raghavendra and David Steurer. How to round any CSP. In *Annual IEEE Symposium on Foundations of Computer Science, (FOCS'09)*, pages 586–594, 2009.

[102] Prasad Raghavendra and David Steurer. Integrality gaps for strong SDP relaxations of UNIQUE GAMES. In *Annual IEEE Symposium on Foundations of Computer Science, (FOCS'09)*, pages 575–585, 2009.

[103] Michaël Rao. MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theor. Comput. Sci*, 377(1-3):260–267, 2007.

[104] Robert W Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[105] Thomas Rothvoß. The matching polytope has exponential extension complexity. In *46th ACM Symposium on Theory of Computing, (STOC'14)*, pages 263–272, 2014.

[106] Marko Samer and Stefan Szeider. Constraint satisfaction with bounded treewidth revisited. *J. Comput. Syst. Sci*, 76(2):103–114, 2010.

[107] Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, 1986.

[108] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

[109] Meinolf Sellmann. The polytope of tree-structured binary constraint satisfaction problems. In *5th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR'08)*, volume 5015 of *Lecture Notes in Computer Science*, pages 367–371, 2008.

[110] Meinolf Sellmann, Luc Mercier, and Daniel H. Leventhal. The linear programming polytope of binary constraint problems with bounded tree-width. In *4th International Conference on Integration of AI and OR Techniques*

in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR'07), volume 4510 of *Lecture Notes in Computer Science*, pages 275–287, 2007.

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

[112] G. Szekeres and Herbert S. Wilf. An inequality for the chromatic number of a graph. *Journal of Combinatorial Theory*, 4(1):1–3, 1968.

[113] Hans Raj Tiwary. Extension complexity of formal languages. *arXiv preprint arXiv:1603.07786*, 2016.

[114] René van Bevern, Andreas Emil Feldmann, Manuel Sorge, and Ondřej Suchý. On the parameterized complexity of computing balanced partitions in graphs. *Theory Comput. Syst*, 57(1):1–35, 2015.

[115] François Vanderbeck and Laurence A. Wolsey. Reformulation and decomposition of integer programs. In *50 Years of Integer Programming 1958-2008*, pages 431–502. Springer, 2010.

[116] Laurence A. Wolsey. Using extended formulations in practice. *Optima*, 85:7–9, 2011.

[117] Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991.

[118] Giacomo Zambelli. Colorings of k-balanced matrices and integer decomposition property of related polyhedra. *Oper. Res. Lett*, 35(3):353–356, 2007.

[119] Dmitriy Zhuk. The proof of csp dichotomy conjecture. *arXiv preprint arXiv:1704.01914*, 2017.

[120] Günter M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, 1995.