Exploiting Structure in CSP-related Problems

Tommy Färnqvist
Dedicated to the memory of Jan Christer Färnqvist (1945 – 2008)
Abstract

In this thesis we investigate the computational complexity and approximability of computational problems from the constraint satisfaction framework. An instance of a constraint satisfaction problem (CSP) has three components; a set $V$ of variables, a set $D$ of domain values, and a set of constraints $C$. The constraints specify a set of variables and associated local conditions on the domain values allowed for each variable, and the objective of a CSP is to assign domain values to the variables, subject to these constraints.

The first main part of the thesis is concerned with studying restrictions on the structure induced by the constraints on the variables for different computational problems related to the CSP. In particular, we examine how to exploit various graph, and hypergraph, acyclicity measures from the literature to find classes of relational structures for which our computational problems become efficiently solvable. Among the problems studied are, such where, in addition to the constraints of a CSP, lists of allowed domain values for each variable are specified ($\text{LHOM}$). We also study variants of the CSP where the objective is changed to: counting the number of possible assignments of domain values to the variables given the constraints of a CSP ($\#\text{CSP}$), minimising or maximising the cost of an assignment satisfying all constraints given various different ways of assigning costs to assignments ($\text{MINHOM}$, $\text{MAX SOL}$, and $\text{VCSP}$), or maximising the number of satisfied constraints ($\text{MAX CSP}$). In several cases, our investigations uncover the largest known (or possible) classes of relational structures for which our problems are efficiently solvable. Moreover, we take a different view on our optimisation problems $\text{MINHOM}$ and $\text{VCSP}$; instead of considering fixed arbitrary values for some (hyper)graph acyclicity measure associated with the underlying CSP, we consider the problems parameterised by such measures in combination with other basic parameters such as domain size and maximum arity of constraints. In this way, we identify numerous combinations of the considered parameters which make these optimisation problems admit fixed-parameter algorithms.

In the second part of the thesis, we explore the approximability properties of the (weighted) $\text{MAX CSP}$ problem for graphs. This is a problem which is known to be approximable within some constant ratio, but not believed to be approximable within an arbitrarily small constant ratio. Thus it is of interest to determine the best ratio within which the problem can be approximated, or at least give some bound on this constant. We introduce a novel method for studying approximation ratios which, in the context of $\text{MAX CSP}$ for graphs, takes the form of a new binary parameter on the space of all graphs. This parameter may, informally, be thought of as a sort of distance between two graphs; knowing the distance between two graphs, we can bound the approximation ratio of one of them, given a bound for the other.

This work has been supported by the Theoretical Computer Science Laboratory, Department of Computer and Information Science, Linköping University, and in part by CUGS (the National Graduate School in Computer Science, Sweden).
Populärvetenskaplig sammanfattning

Den här avhandlingen lämnar bidrag till den gren av teoretisk datalogi som kallas komplexitetsteori. Trots att komplexitetsteori är en formell vetenskap, vilket betyder att dess resultat underbyggs med hjälp av matematik och strikt logiska resonemang, har denna vetenskapsgren ytliga likheter med många grenar av naturvetenskaperna. I likhet med forskare inom vissa av naturvetenskaperna bygger komplexitetsteoretiker upp taxonomier, men i stället för till exempel arter, raser och organismer undersöks och klassificeras komplexitetsklasser, problem och probleminstanser. Så många komplexitetsklasser har nu definierats, eller upptäckts, att de faktiskt huserar i ett eget komplexitets-zoo!

I komplexitetsteori studeras beräkningsproblem, alltså problem som på något sätt låter sig lösas med hjälp av en dator, och de resurser som krävs för att lösa dessa problem. Man kan tänka sig att studera olika typer av resurser, så som tid, minnesutrymme eller hur mycket information olika parallella enheter inblandade i en beräkning behöver utbyta. I den här avhandlingen nöjer vi oss dock med att studera den fundamentalta resursen tid. Närmare bestämt är vi intresserade av att avgöra om vissa beräkningsproblem kan lösas effektivt, med endast hanterbar tidsåtgång, eller om de verkar vara så svåra att lösa exakt att vi måste ta till mer indirekta metoder, som heuristiska eller approximativa metoder.

Den klass av beräkningsproblem vi studerar i den här avhandlingen är hämtade från ramverket av villkorsproblem (på engelska, Constraint Satisfication Problems (CSP)). En instans av ett sådant villkorsproblem består av en mängd \( V \) av variabler, en mängd \( D \) av värden, kallad probleminstansens domän samt en uppsättning villkor \( C \). Varje villkor i \( C \) specificeras av en uppsättning variabler tillsammans med en begränsning av vilka värden de in-

Generaliteten i definitionen av villkorsproblem för dock med sig att de i allmänhet är mycket svåra att lösa och ett grundläggande resultat inom området är att ett så allmänt problem antagligen saknar lösningsmetoder, algoritmer, som kan betraktas som effektiva med avseende på tidsåtgång. Med detta i åtanke är det ingen överaskning att mycket tid har lagts ned på att försöka hitta begränsningar av det allmänna villkorsproblemet som låter sig lösa effektivt och att utforska hur stora sådana begränsade klasser av problem kan göras. I den första delen av den här avhandlingen studeras så kallade strukturella begränsningar av villkorsproblem. Där begränsas hur den struktur villkoren inducerar över variablerna får se ut, till exempel på vilket sätt variablerna i olika villkor tillåts överlappa varandra. I avhandlingen undersöker vi varianter av villkorsproblemet där variablerna, förutom att styrs av villkoren själva, endast tillåts hämta sina värden från särskilda listor av domänvärden. Dessutom studerar vi varianter där målet har ändrats till att istället vara att räkna antalet möjliga tilldelningar av värden till variabler som samtidigt uppfyller alla villkor, samt varianter där målet är att minimalera eller maximalera någon typ av kostnad som vidhäftats olika möjliga tilldelningar. I flera fall leder våra efterforskningar till att vi hittar den största kända (eller möjliga) klassen av strukturella restriktioner vilka leder till effektivt lösbara problemvarianter.

I avhandlingens andra del studerar vi approximerbarhet — hur vi ska hantera beräkningsproblem som är så svåra att lösa att de antagligen inte ens tillåter approximativa lösningar med rimlig tidsåtgång. Vi inför här en generell metod för att föra över resultat om approximationsegenskaper hos ett beräkningsproblem till ett annat, likartat, beräkningsproblem. I avhandlingen visar vi hur metoden kan appliceras på en särskild familj av villkorsproblem, där målet är att hitta tilldelningar som uppfyller så många villkor som möjligt, och att metoden på så vis kan ge upphov till nya, större, klasser av approximationsresultat. I det här sammanhanget kan vi också visa att vår metod har starka och intressanta kopplingar till tidigare forskning inom det matematiska forskningsområdet grafteori.
I want to thank my supervisor Prof. Peter Jonsson for the opportunity to do my thesis work in his group, and for his tireless efforts to install in me some of his wisdom about how our particular brand of science is performed.

I also want to thank:

- my co-authors: Peter Jonsson, Johan Thapper, and Robert Engström;
- Johan Thapper for travelling three continents with me to spread the word about our research;
- Prof. Ulf Nilsson for trusting me to take charge of his course in data structures and algorithms;
- Fredrik Heintz for interesting and stimulating programming contest-related events, activities, and projects;
- past and present members at TCSlab for the creative work atmosphere;
- the administrative staff at IDA for making such issues as transparent as they can be.

I also extend much gratitude to my friends and family for their encouragement. Finally, I thank Therese for giving me love and support. Always.

Tommy Färnqvist
Linköping, December 2012
List of Papers

Parts of this thesis have previously been published in the following refereed publications:


- Tommy Färnqvist. Counting homomorphisms via hypergraph-based structural restrictions. In A. Ridha Mahjoub and Ioannis Millis, editors, *...*


Additionally, the results from the following paper are also included:

• Robert Engström, Tommy Färnqvist, Peter Jonsson, and Johan Thapper. An approximability-related parameter on graphs — properties and applications. Submitted for publication. [EFJT12]
Contents

Abstract v
Populärvetenskaplig sammanfattning vii
Acknowledgments ix
List of Papers xi
Contents xiii

I Introduction 1

1 Introduction 3

1.1 Computational Complexity Theory 5

1.1.1 Graph Theory and Computational Problems 9

1.2 Constraint Satisfaction Problems 12

1.3 Optimisation 13

1.3.1 MinHom 13

1.3.2 Max CSP 14

1.3.3 V CSP 15

1.4 Counting 15

1.5 Approximability 15

1.6 Parameterised Complexity 16

1.7 Main Contributions and Thesis Outline 17

2 Preliminaries 21
## 2.1 Constraint Satisfaction

2.1.1 The Constraint Satisfaction Problem .................................. 21
2.1.2 CSP-related Problems .................................................. 23
2.1.3 Relational Structures and Homomorphisms ......................... 24
2.1.4 Structural restrictions ................................................. 25
2.1.5 Hypergraphs and Tree Decompositions ............................... 35

2.2 Approximability ............................................................. 38
2.3 Parameterised Complexity ................................................ 39

## II Structural Restrictions

### 3 Bounded Arity

3.1 Introduction ............................................................... 43
3.2 Homomorphism and Constraint Satisfaction Problems ............... 44
3.3 Main Results .............................................................. 46
  3.3.1 The Structure $B$ ..................................................... 46
  3.3.2 Hardness Results for Decision Problems ......................... 49
  3.3.3 Hardness Results for Optimisation Problems ..................... 53
3.4 Conclusions and Open Questions ..................................... 58

### 4 Unbounded Arity

4.1 Introduction ............................................................... 61
4.2 \#CSPs and Bounded Width Measures ................................. 63
  4.2.1 Bounded Hypertree Width ......................................... 64
  4.2.2 Bounded Fractional Hypertree Width ............................. 67
4.3 $\text{MINHOM}$, $\text{VCSP}$, and Fractional Hypertree Width ............ 71
4.4 Conclusions and Outlook .............................................. 74

### 5 Parameterised Complexity

5.1 Introduction ............................................................... 77
5.2 Parameterised Constraint Optimisation Problems .................... 78
5.3 Tractability Results ...................................................... 80
5.4 The Domination Lattice and Hardness Results ....................... 86
5.5 Conclusions .............................................................. 91

## III Separation, Graphs, and Approximability

### 6 Separation and Approximability

6.1 Introduction ............................................................... 95
  6.1.1 The Maximum $H$-Colourable Subgraph Problem ................. 98
  6.1.2 Chapter Outline ..................................................... 99
6.2 Basic Properties of the Separation Parameter ......................... 100
  6.2.1 Exact Values and Bounds ........................................... 100
  6.2.2 Exploiting Symmetries (Proof of Theorem 6.9) .................. 103
  6.2.3 A Linear Programming Formulation .............................. 106
PART I

Introduction
1 Introduction

The main purpose of this thesis is to make a contribution to the branch of theoretical computer science called computational complexity theory. Although it is a formal science, a brief look at modern computational complexity theory reveals superficial similarities with the natural sciences. Specifically, computational complexity theorists are building taxonomies; but instead of species, races, and organisms, they investigate, and classify, complexity classes, problems, and problem instances. In fact, so many complexity classes have now been defined, or discovered, that they reside in a Complexity Zoo [Aar12], keeping 495 classes and counting!

In this first chapter of the thesis, we will give a brief introduction to computational complexity theory before familiarising ourselves with the constraint satisfaction framework, whence the computational problems which will be our main concern all originate. But before dealing with these abstract concepts, we give a motivating example.

**Example 1.1.** A Sudoku puzzle takes the form of a $9 \times 9$ grid of cells, some of which are filled in with numbers in the range from 1 to 9, such as the one in Figure 1.1 below. To solve the puzzle, the task is to fill in the remaining cells so that the numbers within every row, column, or $3 \times 3$ block of cells (bounded by the thicker border lines) are all distinct. The Sudoku in Figure 1.1 is considered rather difficult [Epp12], but we still encourage the reader to spend at least a few moments finding a solution to this puzzle.

Imagine now that we work as constructors of Sudoku puzzles, and that we handle so many requests for new puzzles that we want to automate parts of the puzzle construction procedure. In particular, we would like to have
1. INTRODUCTION

![Unsolved Sudoku](image)

Figure 1.1: Unsolved Sudoku.

a systematic way of telling whether a proposed Sudoku puzzle does indeed have a solution or not. Sometimes we might get requests for Sudoku variants where the puzzle grid is smaller or larger than the original 9 × 9 grid. With this in mind, we decide to device a procedure solving the following, more general, problem.

**Sudoku Instance:** An (n × n) Sudoku puzzle.

**Output:** “yes” if the puzzle admits a solution, “no” if the puzzle does not have any solution.

The systematic procedure we now seek is called an algorithm for the Sudoku problem. An algorithm consists of a sequence of precise instructions, intended to be executed by either a human or by a computer, which produces a solution to any instance of the problem. This means that, without prior knowledge, even about the grid size, our algorithm should be able to decide whether the puzzle in Figure 1.1 has a solution or not. Fortunately, it is easy to come up with such an algorithm; just enumerate all possible ways of filling in a blank Sudoku grid of appropriate size — if we encounter a grid that is compatible with the current puzzle instance, we stop and answer “yes”, otherwise we answer “no”. Not long after inventing our algorithm we are a bit dismayed by the fact that already the 9 × 9 Sudoku puzzle has 6,670,903,752,021,072,936,960 valid grids [FJ05]. Assuming we have programmed a computer to execute our algorithm, and that it can generate and check a billion (10^9) valid Sudoku grids per second it would take over 200,000 years before we got our answer in the worst case (i.e., when the answer is “no”) for 9 × 9 puzzle instances. This gets us thinking about whether there is a better algorithm for Sudoku or not and, more generally, about the efficiency of algorithms in general. In particular, our algorithm necessarily contains a subroutine for checking that a proposed solution really is a solution to the current problem instance. This part of the algorithm only needs to check a proposed solution against a few simple rules and only contributes a negligible part of the huge running time of our algorithm. Is it really the
1.1 Computational Complexity Theory

Before beginning our brief introduction to computational complexity theory we want to give the reader interested in delving deeper into the subject a few pointers. The textbooks of Sipser [Sip12], Papadimitriou [Pap94], Moore and Mertens [MM11], Goldreich [Gol08], and Arora and Barak [AB09], and the survey articles by Wigderson [Wig06], Fortnow and Homer [FH03], and Stockmeyer [Sto87] are all excellent starting points. For the more philosophically inclined reader, we wholeheartedly recommend the thought-provoking essay by Aaronson [Aar11].

Computational complexity theory studies problems (such as Sudoku) and the resources required to solve these problems. In this thesis we will restrict ourselves to studying the resource time. The usage of the term “time” is slightly misleading. Since the same algorithm executes in different amounts of time on different computational devices we do not count the number of clock cycles it takes to perform a task, but rather the number of instructions, or computational steps, carried out by the algorithm. For this reason, we settle for some reasonable model of computation and disregard additive and multiplicative constant differences between such models.

What we would like to do now is to be able to study a function $t$ describing the number of computational steps a particular algorithm needs to take, given an instance of the problem we are investigating. This function would probably be very complicated and would probably not tell us very much about the problem under study. As we gather more information and build ever faster computers we are often interested in solving larger and larger instances of the same problem. It would therefore make sense to study the number of computational steps needed by an algorithm as a function of the size of the instance. In this thesis, we will settle for a function measuring the size of an instance that grows asymptotically as the actual size in bits needed to encode the instance. The function $t : \mathbb{N} \rightarrow \mathbb{N}$ is then a function from natural numbers, representing the size of the instance, to natural numbers, representing the maximal number of computational steps required to solve an instance of a specific size. Analysing algorithms this way is called worst case analysis, since in the general case the algorithm can not do better than $t(n)$ steps, but there may be instances of a given size $n$ that the algorithm is able to solve much more quickly. Note also that there may be no instances at all for particular values of $n$, meaning that $t$ may be a partial function. As we already mentioned, we are not interested in the exact values of $t$, but rather in
how \( t \) behaves for large \( n \). This intention is captured by using the following asymptotic notation. For partial functions \( t \) and \( g \), we write

\[ t(n) \in O(g(n)), \]

if there are constants \( c \) and \( n_0 \) such that

\[ t(n) \leq c \cdot g(n) \text{ for all } n > n_0. \]

Using this notation takes care of approximations in the choice of instance size and uninteresting constants inherent in the computational model. If \( t(n) \in O(g(n)) \), we say that the time complexity of the algorithm using \( t \) computational steps is \( O(g(n)) \).

We have now encountered both instances and problems, in the sense computational complexity theorists talk about them. But what about the “complexity classes” alluded to in the very first paragraph of this chapter? We have already seen the use of the word “complexity” in this sense, i.e., meaning the resource consumption of a particular algorithm for some problem. Complexity theorists are interested in figuring out which computational problems can be solved with a particular set of resources. Often more importantly, we try to discern which problems cannot be solved with a certain efficiency, meaning we have to rule out every thinkable algorithm for a problem using a certain amount of resources. Often more importantly, we try to discern which problems cannot be solved with a certain efficiency, meaning we have to rule out every thinkable algorithm for a problem using a certain amount of resources. If we manage this task, we say that we have decided the complexity of the problem, i.e., we have decided that a certain resource set is not sufficient to solve a problem regardless of the way we put these resources to use.

A typical complexity class has a definition of the form “the set of problems that can be solved by an abstract computational machine \( M \) using \( O(g(n)) \) of resource \( R \), where \( n \) is the size of the input”. Hence, the inclusion of a problem in a complexity class can be concluded by constructing an algorithm for the problem that is efficient in the appropriate sense. The SUDOKU problem is an example of a decision problem. This class of problems is the problems where the set of solutions is taken to be \{“yes”, “no”\}. The most basic complexity class is called \( \text{P} \), which stands for polynomial time. This class contains all decision problems for which we can construct an algorithm that executes a number of computational steps bounded by a polynomial in the size of the input instance, in the worst case.

**Example 1.2.** The problem of checking whether a proposed solution to an \((n \times n)\) instance of SUDOKU is valid is in \( \text{P} \). We have to check for each row, that all numbers \( \{1, 2, \ldots, n\} \) occur exactly once. By checking off each number when we find it, this can be accomplished in time that is linear in \( n \) for each row, and in time \( O(n^2) \) for all rows. The same type of procedure can be used to check all columns and blocks. Finally, we have to check that each cell agrees with the values in the input instance. To sum up, we have constructed an algorithm for checking proposed solutions to SUDOKU with time
1.1. Computational Complexity Theory

The complexity $O(n^2)$. In figure 1.2 is a solution to the Sudoku puzzle in figure 1.1. The reader is encouraged to check that the procedure we have just sketched actually is sound.

Many problems are more naturally stated as computing a result belonging to a domain larger than only \{"yes", "no"\}. For example, computing the sum of two numbers can be computed in polynomial time using the school book algorithm. Problems of this kind formally belong to the complexity class $\text{FP}$. It has turned out to be a very robust notion to consider problems that cannot be solved in a polynomial number of computational steps to be infeasible computationally (except on small instances), even taking into account the computers of tomorrow. We note that quantum computing [NC00] may come to poke a hole in this “truth”, but this is very much a question of debate. In any case, making a distinction between problems inside and outside of $\text{P}$, and trying to trace this boundary, has proved surprisingly fruitful for both practical and theoretical purposes.

Another very important complexity class is called $\text{NP}$, which stands for non-deterministic polynomial time. This class contains all decision problems for which a correct solution can be checked in a number of steps bounded by a polynomial in the size of the solution. (The size of the solution itself is supposed to be bounded by a polynomial in the input size.) The class $\text{NP}$ obviously contains all problems in $\text{P}$. By the algorithm we gave in Example 1.2, the $\text{SUDOKU}$ problem can also be placed in $\text{NP}$. Actually, $\text{SUDOKU}$, together with thousands of other problems in $\text{NP}$, are part of a class of problems which are not believed to be in $\text{P}$. Returning to our question of whether it is easier to find a solution to a Sudoku puzzle than checking that a proposed solution is correct, our intuition might say that yes, this is the case. This same question, in more general form, i.e., the question of whether is “easier” to verify a correct solution to a problem than to find one can now

\begin{center}
\begin{tabular}{ccccccccc}
3 & 8 & 9 & 5 & 2 & 6 & 1 & 7 & 4 \\
4 & 6 & 5 & 3 & 1 & 7 & 9 & 8 & 2 \\
1 & 7 & 2 & 8 & 4 & 9 & 3 & 6 & 5 \\
6 & 5 & 1 & 4 & 7 & 3 & 8 & 2 & 9 \\
7 & 9 & 8 & 2 & 6 & 5 & 4 & 3 & 1 \\
2 & 3 & 4 & 1 & 9 & 8 & 6 & 5 & 7 \\
5 & 4 & 6 & 7 & 8 & 1 & 2 & 9 & 3 \\
8 & 2 & 3 & 9 & 5 & 4 & 7 & 1 & 6 \\
9 & 1 & 7 & 6 & 3 & 2 & 5 & 4 & 8 \\
\end{tabular}
\end{center}

Figure 1.2: Solved Sudoku.

\footnote{There is now a multitude of computer programs available which enable solving this Sudoku puzzle in fractions of a second. Note that these programs rely on the fact that $n = 9$.}
be compactly stated in the language of computational complexity theory as ‘P ≠ NP?’.

Despite our intuition, and despite the fact that it has been 40 years since the conception of the complexity class NP [Coo71, Lev73], we really have no clue how to decide whether or not P ≠ NP. The quest for the resolution of this question is not only central to theoretical computer science, but considered very important for mathematics as a whole; so important that the Clay Mathematics Institute included it in its list of seven “Millennium Problems”, awarding one million dollars for a solution that holds up to peer review [Cla00]. Of course, most researchers believe that P ≠ NP, and it is not a controversial hypothesis to build one’s work upon. Indeed, many of the results in this thesis depends on this hypothesis being true. If it were to turn out that P = NP, these results, along with much of modern complexity theory, would be rendered completely useless.

We have already touched upon a reason why it seems so hard to resolve the ‘P ≠ NP?’ question: A proof that these two complexity classes are different seems to require an explicit problem in NP which is not a member of P. Proving such a property must necessarily consider all possible known, as well as yet unheard of, algorithms for solving this particular problem. We do, however, have a technique for identifying problems which are “at least as hard” as other problems. The basis of this technique is a tool called a reduction or a reducibility. While we will meet many kinds of reduction in this thesis, we start by mentioning the Karp reduction also known as a polynomial-time reduction.

**Definition 1.3.** A polynomial time reduction from a problem Π₁ to a problem Π₂ is a polynomial time computable function which takes as input an instance I₁ of Π₁ and produces as output an instance I₂ of Π₂ such that the solution to I₁ is “yes” if and only if the solution to I₂ is “yes”.

It is not hard to see that Karp reductions preserve membership in both P and NP, in the sense that if Π₂ is in P (NP), then Π₁ is in P (NP). If there is a polynomial-time reduction from Π₁ to Π₂, we think of Π₂ as at least as hard as Π₁ to solve. This is because to any algorithm for Π₂, we can add the reduction as a first step to get an algorithm for Π₁, which is at most a polynomial factor slower than the original algorithm for Π₂. Hence, if Π₁ is not in P, then neither is Π₂.

One of the true gems of complexity theory is the fact that there are problems Π in NP that are harder than any other problem in NP in the sense that every problem in NP is polynomial-time reducible to Π. Such problems are called NP-complete, and thousands of them have been identified since the seminal papers of Cook [Coo71], Levin [Lev73], and Karp [Kar72].

At this point in the thesis, it might come as no surprise to the reader to know that SUDOKU is NP-complete [YS03]. We will not give a proof of this statement but we will try to put SUDOKU in the context of other NP-complete problems to perhaps strengthen the reasons to believe this result. To do this, we need to introduce the combinatorial structures known as graphs.
1.1.1 Graph Theory and Computational Problems

Graphs are truly ubiquitous in computer science, but can also be used in any science where we need to model for example pairwise relations or some kind of process dynamics.

Definition 1.4. An (undirected) graph $G$ is a tuple $(V(G), E(G))$, where $V$ is a set of vertices or nodes, and $E$ is a set of edges. Each edge has a set of one or two vertices associated to it, which are called its endpoints.

When illustrating graphs, the vertices are usually drawn as points, and the edges as lines connecting the vertices of their respective endpoints. If two vertices of a graph share an edge, they are said to be adjacent or neighbours. The neighbourhood, $N_G(v) = N(v)$, of the vertex $v \in V$ in the graph $G$, is defined as the set $\{u \in v \mid \{u,v\} \in E(G)\}$. The degree, $\deg_G(v) = \deg(v)$, of a vertex $v \in V(G)$, can be defined as $|N_G(v)|$. A graph $G = (V,E)$ is called $d$-regular if $\deg(v) = d$ for all $v \in V$.

Let $G = (V,E)$ be a graph. If $E' \subseteq E$, we say that the graph $G' = (V,E')$ is a subgraph of $G$. If $V' \subseteq V$ is a subset of vertices of $G$, then the graph $G[V'] = (V', E \cap V' \times V')$ is called the subgraph induced by $V'$.

At times, we will allow edges, called loops, which go from a vertex $v$ to itself. A simple graph is a loop-free graph where any two vertices has at most one edge joining them. A graph whose vertices can be divided into two disjoint sets $V_1$ and $V_2$ such that every edge connects a vertex in $V_1$ to one in $V_2$ is called a bipartite graph. In a directed graph, the edges have their endpoints ordered.

Example 1.5. The complete graph, $K_n$, is a simple, undirected graph, on vertex set $\{1, \ldots, n\}$, where every pair of distinct vertices is connected by a unique edge. It is $(n-1)$-regular, and contains $\binom{n}{2} = n(n-1)/2$ edges. The cycle graph, $C_n$, is a simple undirected graph, on vertex set $\{1, \ldots, n\}$, with edge set $E(C_n) = \{\{a,b\} \mid |a-b| \equiv 1 \mod n\}$. It is 2-regular, and contains $n$ edges. $C_n$ is a subgraph of $K_n$, but not an induced subgraph. The complete bipartite graph, $K_{m,n} = (V_1 + V_2, E)$ is a simple, undirected graph, on a vertex set partitioned into two parts, with $V_1 = \{1, \ldots, m\}$ and $V_2 = \{m+1, \ldots, m+n\}$, and edge set $E(K_{m,n}) = \{\{v_1,v_2\} \mid v_1 \in V_1 \text{ and } v_2 \in V_2\}$. It has $mn$ edges. The graphs $K_5$, $C_7$, and $K_{3,3}$ are depicted in Figure 1.3.

The literature on graph theory is massive, but one concept that stands out is that of a graph colouring: A vertex colouring of a graph $G$ is a function $f : V(G) \to C$, from the vertices of $G$ to a set of colours, $C$. If $f$ is such that for any edge $\{u,v\} \in E(G)$, we have $f(u) \neq f(v)$, then $f$ is called a proper colouring.

Example 1.6. SUDOKU can be formulated as a graph-based problem. The graph associated to an instance of SUDOKU has vertex set $\{1, \ldots, n^2\}$, one vertex for each cell of the Sudoku grid. Two vertices are connected by an edge if the cells that they correspond to are in the same row, column or $(n \times n \times n)$.
1. **Introduction**

Figure 1.3: The graphs $K_5$, $C_7$, and $K_{3,3}$.

$n$) block. Furthermore, we assign the numbers from 1 to $n$ a colour, and then colour the vertices corresponding to cells that contain a given number in the colour of the number. A valid solution to a SUDOKU instance now corresponds to a proper $n$-colouring of the corresponding, already partially coloured, graph.

By the example given above, solving the Sudoku puzzle in Figure 1.1 corresponds to colouring a particular, pre-coloured, graph with 9 colours. Since SUDOKU is NP-complete, we know that certain types of graph colouring problems are considered computationally intractable. What about general graphs? It turns out that already the problem of deciding if a proper 3-colouring of a given graph $G$ exists was on Karp’s original list of 21 NP-complete problems [Kar72]. Interestingly enough, the proof of the following theorem shows that there is even a polynomial time algorithm for *finding* a proper 2-colouring in a given graph $G$ if such a colouring exists.

**Theorem 1.7.** The problem of deciding if a given graph $G$ admits a proper 2-colouring is in $P$.

**Proof.** To prove the theorem we need to exhibit an algorithm solving the problem, which runs in time polynomial in the size of $G$. Notice that the existence of a proper 2-coloring of $G$ implies that we can divide $V(G)$ into two distinct sets; $V_1$, consisting of “white” vertices, and $V_2$, consisting of “black” vertices. Furthermore, $G[V_1]$ and $G[V_2]$ both have empty edge sets, since the only edges that can exist are those with one vertex in $V_1$ and one vertex in $V_2$. In other words, for $G$ to have a proper 2-colouring it has to be a bipartite graph. To test if $G$ is bipartite we proceed in a greedy fashion: Pick any vertex of $G$ and colour it “white”. Then continue by colouring all its neighbours “black”. For every newly coloured vertex, colour its neighbours with the opposite colour. If the need arises to colour an already coloured vertex with a different colour than it already has, we can stop and conclude that the graph cannot be 2-coloured and hence is not bipartite (it must contain an odd cycle). Otherwise, we have found a proper 2-colouring of $G$. 

10
In the worst case, when $G$ is 2-colourable, we look at each edge twice; once in each “direction”. We also do work which take time proportional to the sum of the degrees of all vertices in $G$, and since $\sum_{v \in V(G)} \deg_G(v) = 2|E(G)|$ always holds\(^2\), we can conclude that our algorithm has time complexity $O(|V(G)| + |E(G)|)$.

By Theorem 1.7 we can decide in polynomial time if a given graph is bipartite. But what if the graph is not bipartite and we still want to try to colour it with 2 colours in such a way that as few edges as possible have their endpoints coloured with the same colour? It turns out that this is equivalent to solving a problem called MAX CUT: Let $G = (V, E)$ be a graph and let $S$ be a subset of $V$. We say that a cut in $G$ with respect to $S$ is the set of edges with one endpoint in $S$ and one endpoint in the complement $V(G) \setminus S$. Observe that any partition of $V(G)$ into two disjoint sets gives rise to a cut in $G$. The MAX CUT problem asks for the size of a largest cut in $G$. It is possible to find a polynomial-time reduction from an NP-complete problem to MAX CUT. We often express this by saying that MAX CUT is NP-hard. In fact, MAX CUT also was on Karp’s original list of NP-hard problems [Kar72].

Searching for graph colourings as well as cuts in graphs have strong ties to the concept of looking for homomorphism between graphs. A graph homomorphism from a graph $G$ to a graph $H$ is a vertex map which carries the edges in $G$ to edges in $H$. More formally, we have the following definition.

**Definition 1.8.** Let $G$ and $H$ be graphs. A graph homomorphism from $G$ to $H$ is a function $f : V(G) \to V(H)$ such that

$$\{u, v\} \in E(G) \Rightarrow \{f(u), f(v)\} \in E(H).$$

The existence of a graph homomorphism from $G$ to $H$ is denoted by $G \rightarrow H$. In this case, we say that $G$ is homomorphic to $H$. For an excellent introduction to and survey of this topic, we refer to [HN04b].

**Example 1.9.** The graph $K_2$ is just two vertices with an edge between them. From the definition of a graph homomorphism, we see that for a graph $G$ to be homomorphic to $K_2$ we need to be able to partition the vertex set of $G$ into two disjoint sets, such that all edges of $G$ have their endpoints in different partitions. Hence, deciding if a given graph $G$ is 2-colourable is equivalent to deciding if $G$ is homomorphic to $K_2$.

Extrapolating from Example 1.9, we see that deciding 3-colourability of graphs is equivalent to deciding if they are homomorphic to $K_3$, and, more generally, that deciding $k$-colourability is the same as deciding if there exists a graph homomorphism from $G$ to $K_k$.

\(^2\)This is called the degree sum formula, and was proved already by Leonhard Euler in 1736, in the paper that started the study of graph theory. Euler counts the number of incident pairs $(v, e)$, where $e$ is an edge and the vertex $v$ is one of its endpoints, in two different ways. Any vertex $v$ belongs to $\deg(v)$ pairs. It follows that the number of incident pairs is the sum of the degrees. However, it is also the case that each edge in the graph belongs to exactly two incident pairs, one for each of its endpoints; therefore, the number of incident pairs is $2|E|$. 

11
1. Introduction

A homomorphism to $K_k$. But, what happens if we turn the question around and ask if there is a homomorphism from $K_k$ to a given graph $G$? Since $K_k$ is a complete graph on $k$ vertices, for $K_k$ to be homomorphic to $G$, it is necessary for $G$ to have a complete subgraph on $k$ vertices, also known as a $k$-clique (or a clique of size $k$). Deciding if a given graph has a clique of size $k$ is another fundamental graph problem, and yet another problem on Karp’s original list of 21 NP-complete problems [Kar72].

1.2 Constraint Satisfaction Problems

In this thesis we consider a particular class of problems called constraint satisfaction problems, or CSPs. The formal definition of a CSP is given in Chapter 2.1.1. Here, we settle for an informal definition: An instance of a CSP has three components, a set $V$ of variables, a set $D$ of values, called the domain, and a set of constraints $C$. The constraints specify a scope, i.e., a set of variables and associated local conditions on the domain values allowed for each variable, and the objective of a CSP is to assign domain values to the variables, subject to these constraints. Obviously, this is a very general type of problem, and consequently, a large class of problems in different areas of computer science can be viewed as constraint satisfaction problems. This includes problems in artificial intelligence, database theory, scheduling, frequency assignment, graph theory, and satisfiability [Bul11, FH06, FV98, Gro07, Jea98, KV00].

Example 1.10. The problem of deciding if a given graph $G$ is 3-colourable can be cast as a CSP: Each vertex of $G$ has a corresponding variable in the CSP instance, the domain is the set of colours, say, {"white", "black", "grey"}, and for each edge $\{u, v\} \in E(G)$ there is a constraint saying that $u$ and $v$ cannot be assigned the same domain value.

Example 1.11. An instance $(G, k)$ of the $k$-CLIQUE problem, whose objective is to decide if $G$ has a clique of size $k$, corresponds to the following CSP-instance: The variables are $x_1, \ldots, x_k$, the domain is the vertex set of $G$, and for each pair $(x_i, x_j)$ of variables, with $1 \leq i < j \leq k$, there is a constraint allowing the assignment $x_i = u, x_j = v$ if and only if $\{u, v\} \in E(G)$.

For a given CSP instance, and a proposed assignment $f : V \rightarrow D$ of domain values to the variables, it is easy to check, in polynomial time, whether or not all the constraints hold. If this is the case, we say that $f$ is a satisfying assignment or a solution to the CSP instance. Since the checking procedure is polynomial-time, we conclude that CSP is in the complexity class NP, and that, by the examples above, deciding if a CSP instance has a satisfying assignment is NP-complete.

A lot of research effort has been made to understand the computational complexity of CSPs, with the goal of finding restrictions of the problem that are computationally tractable, and, ultimately, determining the boundary between tractable and intractable CSPs. The two main types of restrictions that have been studied are constraint language restrictions, which are concerned
with the types of constraints that occur, and *structural restrictions*, which are concerned with the structure induced by the constraints on the variables, for example with the way the constraints overlap.

As we have seen, both finding \( k \)-cliques and deciding if a graph is 3-colourable can be formulated as CSPs, but also as graph homomorphism problems. Thus it comes as no great surprise that it is a well-known observation (going back to Fever and Vardi [FV98]) that CSPs in general can be described as homomorphism problems. This correspondence requires more general relational structures than graphs, and is worked out in Chapter 2.1.3. Here we are content with noting that the homomorphism formulation of graph 3-colouring corresponds to a CSP with restricted constraint language, and that the homomorphism formulation of finding cliques of size \( k \) corresponds to a CSP with restricted structure.

### 1.3 Optimisation

In many cases merely knowing the existence of a solution to a given computational problem is insufficient. Think, for example, of the MAX CUT problem, where every partition of the vertex set of the input graph gives rise to a cut, but where different partitions may yield wildly different cut sizes. So, there might be need for finding the “best” solution according to some specific criteria. There are several viable approaches for introducing optimisation criteria on CSP instances. We will primarily focus on the following two types of criteria.

- Assigning weights to the variables and domain elements and introducing an objective function based on these weights.
- Allowing constraints to be unsatisfied in a solution, and basing the objective function on the number of satisfied constraints.

Needless to say, introducing such optimisation criteria might yield a problem that is strictly harder (provided $P \neq NP$) than the original problem. We continue this section by introducing those optimisation variants of the constraint satisfaction problem which will be our main concern in this thesis. The formal definitions of these problem can be found in Chapter 2.1.2.

#### 1.3.1 MinHOM

The *minimum cost homomorphism problem*, MinHOM, was introduced by Gutin, Rafiey, Yeo, and Tso [GRYT06], where it was motivated by a real-world problem in defence logistics. MinHOM is a natural optimisation version of the CSP, of the first type mentioned above, where assigning a domain value to a variable is afflicted with costs and the objective is to find the minimum cost of a satisfying assignment. This problem includes as special cases the list homomorphism problem and the general optimum cost chromatic partition problem [LGKR97]. MinHOM also has applications in supervised
machine learning [Tak10] and combinatorial auctions, where it has been used in order to model and solve the winner determination problem of determining the allocation of the items among the bidders that maximises the sum of the accepted bid prices [CG07]. The MINHOM problem will feature prominently in Part II of the thesis.

Example 1.12. A subset \( S \) of vertices of a graph is called an independent set if there is no edge between any pair of vertices \( u, v \in S \). The objective of the problem MAXIMUM INDEPENDENT SET is to find an independent set of maximum size in a given input graph \( G \). It is well-known that finding an optimal solution to an instance of this problem is NP-hard. However, it is easy to find a solution; the empty set is a subset of the vertices of any graph, and it is always an independent set.

To formulate the MAXIMUM INDEPENDENT SET as a CSP, we first work out a correspondence to a graph homomorphism problem. Let \( H \) be a graph on vertex set \( \{0, 1\} \), and edge set \( \{(0,0),(0,1)\} \), i.e., \( H \) has an edge between vertex 0 and vertex 1, as well as a loop on vertex 0. Now, a homomorphism from an input graph \( G \) to \( H \) corresponds to an independent set as follows: If a vertex \( v \in V(G) \) is mapped to vertex 1 in \( H \), then \( v \in S \), and if it is mapped to vertex 0, then \( v \notin S \). A maximum independent set is found by searching for a homomorphism from \( G \) to \( H \) maximising the number of vertices of \( G \) mapped to 1. To obtain a CSP-formulation, we can again take the vertices of the input graph as variables, \( V \), and let the domain \( D \) be \( \{0, 1\} \). For each edge \( \{u,v\} \) we impose the constraint that the pair of variables \( (u,v) \) may only be assigned the values \( (0,0) \), \( (0,1) \), or \( (1,0) \). If we furthermore say that assigning the value 0 to a variable has the cost 2, and that assigning the value 1 to a variable has the cost 1, the size of a maximum independent set is precisely the cost of a minimum cost satisfying assignment to a corresponding instance of MINHOM.

1.3.2 MAX CSP

The maximum constraint satisfaction problem, MAX CSP, is an optimisation version of the CSP of the second kind mentioned above, where we have added a weight function to the constraints of a plain CSP instance. The objective is to maximise the weighted sum of satisfied constraints [FW92]. In part III of the thesis, a special case of MAX CSP will be the main problem under study.

Example 1.13. The MAX CUT problem can be formulated as a case of MAX CSP: Similar to our previous CSP-formulations, we let the set of variables be the vertex set of the input graph and let the domain be \( \{b,w\} \). For each edge \( \{u,v\} \) in the input graph, we enforce the constraint that the variable pair \( (u,v) \) may take the values \( (b,b) \) and \( (w,w) \). This way, the corresponding CSP instance has a satisfying assignment precisely when the input graph is bipartite and the maximum cut is given by the entire edge set of the input graph. The MAX CUT problem is now obtained by viewing the generated
1.4. Counting

A natural extension of the decision problems we have encountered in the thesis is to consider counting problems. For example, we might ask, for any problem in \( \text{NP} \), not only if it has a solution, but how many solutions there are. The corresponding complexity class is called \( \#P \) and has its own set of reductions and complete problems. We note that a \( \#P \)-complete problem is considered to be extremely hard computationally, and even less likely than an \( \text{NP} \)-complete problem to admit a polynomial-time algorithm.

Part II of the thesis, in particular Chapter 4, features the study of structurally restricted counting constraint satisfaction problems, \( \# \text{CSPs} \). Our ability to solve this problem has several applications in artificial intelligence \[ \text{Orp90; Rot96} \], statistical physics \[ \text{BS94; LG71} \], and more recently in guiding backtrack search heuristics to find solutions to CSPs \[ \text{Pes05} \]. Of course, the \( \# \text{CSP} \) problem is \( \#P \)-complete in general. We postpone its formal definition to Chapter 2.1.2.

1.5. Approximability

In Section 1.3, we encountered optimisation problems that are \( \text{NP} \)-hard to solve to optimality. By considering approximation algorithms it turns out that it might be possible to differentiate the hardness of such problems. This is because some \( \text{NP} \)-hard optimisation problems actually are easy to solve if we are willing to sacrifice the requirement that we always must find an optimal solution, while others still remain hard to solve, even without this requirement. Approximation algorithms is one of the main concerns in Part III of the thesis. We introduce this framework and its definitions more formally in Chapter 2.2.
Example 1.14. There is a simple randomized approximation algorithm for the MAX CUT problem. Let \(G = (V, E)\) be a graph. Place each vertex \(v \in V\) into \(S\) or \(V \setminus S\) with equal probability. By linearity of expectation, half the edges of \(G\) are cut. It is also easy to derandomize this algorithm by the method of conditional probabilities. One such resulting algorithm is the following: Given a graph \(G = (V, E)\), start with an arbitrary partition \(V\) and move a vertex from one side of the partition to the other if it improves the solution until no such vertex exists. The number of iterations of this algorithm is bounded by \(O(|E|)\), since the algorithm improves the cut value by at least 1 at each step, and the maximum cut is at most \(|E|\). When the algorithm terminates, each vertex \(v \in V\) has at least half its edges cut (otherwise moving \(v\) to the other side of the partition improves the solution). Hence, the cut is at least \(0.5 \cdot |E|\).

We say that an algorithm approximates a maximisation problem within a constant \(r\), if it always returns a solution \(f\), such that the measure of \(f\) divided by the measure of an optimal solution is bounded from below by \(r\). We call \(r\) the approximation ratio of the algorithm. There are two complexity classes for optimisation problems that are natural analogues to the classes \(P\) and \(NP\) for decision problems. These are called \(PO\) and \(NPO\). The complexity class \(APX\) is the class of optimisation problems in \(NPO\) which can be approximated within some constant \(r > 0\). In Example 1.14 we demonstrated algorithms which approximate MAX CUT within 1/2. Therefore, we have shown that the MAX CUT problem is in \(APX\). Furthermore, it can be shown that the MAXIMUM INDEPENDENT SET problem is not in \(APX\), but instead resides in a complexity class called \(poly-APX\), which strictly contains \(APX\) (if \(P \neq NP\)). This means we have managed to differentiate the hardness of these two problems. Both \(APX\) and \(poly-APX\) have complete problems under appropriate reductions (see Chapter 2.2). In fact, MAX CUT is \(APX\)-complete, while MAXIMUM INDEPENDENT SET is \(poly-APX\)-complete.

1.6 Parameterised Complexity

In the previous section, we saw a common way of handling computational hardness of optimisation problems; using algorithms that approximate the optimal solution in polynomial time, with some acceptable guarantee on the worst case performance of the algorithm relative to the optimum solution. In this section, we introduce another way of trying to cope with \(NP\)-hardness, this time geared towards decision problems.

In parameterised complexity, we relax the classical notion of tractability, polynomial-time computability, by admitting algorithms whose running time may be exponential (or worse) in some parameter that can be expected to be small in the typical application. We say that a problem is fixed-parameter tractable, if there is an \(O(f(k) \cdot N^c)\) algorithm for the problem, where \(f\) is an arbitrary (could be exponential or worse) function of the parameter \(k\), \(N\) is the size of the input, and \(c\) is a constant independent of \(k\). A thorough
1.7. Main Contributions and Thesis Outline

An introduction to this fascinating subject can be gained from the monographs by Downey and Fellow [DF99] and Flum and Grohe [FG06]. In the thesis, we give further details and definitions of this framework in Chapter 2. Parameterised complexity plays an important role in Chapter 3 and is the main theme of Chapter 5.

Example 1.15. To get a bit of a feel for the odds and ends of parameterised complexity, we will demonstrate that \textbf{MAX CUT} is fixed parameter tractable when parameterised by the cut size. An instance of this problem will now be a pair \((G,k)\), where \(G\) is a graph, and we are asked to decide whether \(G\) has a cut of size at least \(k\). If \(k \leq \lceil |E(G)|/2 \rceil\), then the required cut can be found in \(O(|V| + |E|)\) time using the deterministic algorithm sketched in Example 1.14. Otherwise, we do the following: Cycle through all subsets \(S \subseteq E(G)\) of size exactly \(k\). For each \(S\), check if the graph \((V(G), S)\) is bipartite. If any such subset is found, stop and return “yes”, otherwise return “no”. There are at most \(\binom{|E(G)|}{k}\) subsets to check, and processing each subset requires \(O(|E|) \in O(2^k)\) time. Since \(\binom{|E|}{k} \leq 2^{|E|} < 2^{2^k}\), this algorithm has time complexity \(O(|V| + |E| + k2^k)\). In conclusion, we have a fixed parameter algorithm for \textbf{MAX CUT} parameterised by the cut size.

The astute reader has now undoubtedly surmised that the fixed-parameter tractability result given above is not actually that exiting. Since our problem always has a “yes” answer when \(k \leq \lceil |E(G)|/2 \rceil\), non-trivial situations arise only for large parameter values, in which range the fixed-parameter algorithm described above is not really feasible to utilise. However, by a result of Mahajan and Raman [MR99], also \textbf{MAX CUT} parameterised by \(\lceil |E(G)|/2 \rceil + k\) is fixed-parameter tractable.

1.7 Main Contributions and Thesis Outline

We conclude this introductory chapter of the thesis with a “road map”. The main body of the thesis has been divided into two parts, roughly separating considerations of computational complexity from those of approximability.

Part I

This part contains the introduction, as well as some preliminaries. In Chapter 2, we begin by (re-)introducing the constraint satisfaction framework and the main optimisation and counting problems considered in the thesis. We then proceed with formal treatments of approximability and parameterised complexity, and their related reductions. This chapter also serves to establish conventions and notation which will be assumed throughout the thesis.
Part II

In the first main part of the thesis, we are mainly concerned with the computational complexity of structurally restricted CSP-related problems.

- In Chapter 3, we deal with homomorphism and optimisation problems, where the arity of constraints is bounded: For every class of relational structures $\mathcal{A}$, let $\text{LHOM}(\mathcal{A}, \_\_)$ be the problem of deciding whether a structure $A \in \mathcal{A}$ has a homomorphism to a given arbitrary structure $B$, when each element in $A$ is only allowed a certain subset of elements of $B$ as its image. We prove, under a certain complexity-theoretic assumption, that this list homomorphism problem is solvable in polynomial time if and only if all structures in $\mathcal{A}$ have bounded tree-width. The result is extended to the connected list homomorphism, edge list homomorphism, minimum cost homomorphism, valued constraint satisfaction, and maximum solution problems. We also show an inapproximability result for the $\text{MinHOM}$ problem.

- In Chapter 4, we investigate structurally restricted $\#\text{CSP}$, $\text{MinHOM}$, and $\text{VCSP}$ problems, where there are no bounds on the constraint arities. The way in which the graph structure of the constraints influences the computational complexity of counting constraint satisfaction problems is well understood for constraints of bounded arity. The situation is less clear if there is no bound on the arities. In this chapter, we initiate the systematic study of these problems and identify new classes of polynomial time solvable instances based on dynamic programming over tree decompositions, in a way generalizing well-known approaches to combinatorial optimization problems on bounded tree-width graphs, but basing the decompositions on various hypergraph width measures from the literature on plain CSPs. We also use these hypergraph-based with measures to enlarge the tractability landscape for $\text{MinHOM}$ and $\text{VCSP}$ problems.

- In Chapter 5, we take a different view on our optimization problems $\text{MinHOM}$ and $\text{VCSP}$; instead of considering fixed arbitrary values for some structural invariant of the (hyper)graph structure of the constraints, we consider the problems parameterised by the tree-width of primal, dual, and incidence graphs, combined with several other basic parameters such as domain size and arity. Such parameterisations of plain CSPs have been studied by Samer and Szeider. Here, we extend their framework to encompass our optimization problems, by coupling it with further non-trivial machinery and new reductions. By doing so, we are able to determine numerous combinations of the considered parameters that make our optimization problems admit fixed-parameter algorithms.
Part III

In the second main part of the thesis, we introduce a novel method, designed to extend known approximation ratios for one problem to bounds on the ratio for other problems.

- In Chapter 6 we introduce a binary parameter on optimisation problems called separation. The parameter is used to relate the approximation ratios of different optimisation problems; in other words, we can convert approximability (and non-approximability) result for one problem into (non)-approximability results for other problems. Our main application is the problem (weighted) maximum H-colourable subgraph (MAX H-COL), which is a restriction of the general MAX CSP problem to a single, binary, and symmetric relation. In this context the separation parameter turns out to depend heavily on the symmetries of the involved graphs. Using known approximation ratios for MAX k-CUT, we obtain general asymptotic approximability results for MAX H-COL for an arbitrary graph H. For several classes of graphs, we provide near-optimal results under the unique games conjecture.

- In Chapter 7 we investigate separation as a graph parameter. In this vein, we study its properties on circular complete graphs. Furthermore, we establish a close connection to work by Šámal on cubical colourings of graphs. This connection shows that our parameter is closely related to a special type of chromatic number. We believe that this insight may turn out to be crucial for understanding the behaviour of the parameter, and in the longer term, for understanding the approximability of optimisation problems such as MAX H-COL.

Part IV

In the final part of the thesis, Chapter 8 concludes by giving a broader view on the thesis topics. We take the opportunity, while giving a short conceptual summary of the thesis, to expand a bit on both different ways of representing constraint relations as well as another area, namely combinatorial auctions, where the hypergraph invariants used in the thesis have nice implications. The thesis is finished by relating a few interesting future research directions concerning the separation parameter introduced in Part III of the thesis.
In this chapter we formally define the CSP-related problems and associated notions which will be our main concern in the remainder of the thesis. We also give the relevant theoretical background to approximability, parameterised complexity, and their related reductions. This chapter also serves to establish conventions and notation which will be assumed throughout the thesis.

2.1 Constraint Satisfaction

2.1.1 The Constraint Satisfaction Problem

Definition 2.1. An instance $I$ of the constraint satisfaction problem (CSP) is a triple $(V, D, C)$, where

- $V$ is a finite set of variables,
- $D$ is a finite set of domain values, and
- $C$ is a finite set of constraints.
Each constraint in $C$ is a pair $(S, R)$, where the constraint scope $S$ is a nonempty sequence of distinct variables from $V$, and the constraint relation $R$ is a relation over $D$ whose arity matches the length of $S$. The constraint language of $\mathcal{I}$ is the set of all relations that occur in the constraints in $C$.

We will assume that every variable occurs in at least one constraint scope and that every domain element occurs in at least one constraint relation. We write $\text{var}(C)$ for the set of variables that occur in the scope of constraint $C$, $\text{rel}(C)$ for the relation of $C$, and $\text{con}(x)$ for the set of constraints that contain variable $x$ in their scopes. Moreover, for a set $C$ of constraints, we set $\text{var}(C) = \bigcup_{\sigma \in C} \text{var}(\sigma)$.

An assignment is a mapping $\sigma : X \rightarrow D$ defined on some set $X$ of variables. Let $C = ((x_1, \ldots, x_n), R)$ be a constraint and $\sigma : X \rightarrow D$. We define

$$C[\sigma] = \{(d_1, \ldots, d_n) \in R : x_i \not\in R \text{ or } \sigma(x_i) = d_i, 1 \leq i \leq n\}.$$ 

Thus, $C[\sigma]$ contains those tuples of $R$ that do not disagree with $\sigma$ at some position. An assignment $\sigma : X \rightarrow D$ is consistent with a constraint $C$ if $C[\sigma] \neq \emptyset$. An assignment $\sigma : X \rightarrow D$ satisfies a constraint $C$ if $\text{var}(C) \subseteq X$ and $\sigma$ is consistent with $C$. An assignment satisfies a CSP instance $I$ if it satisfies all constraints of $I$. The instance $I$ is consistent (or satisfiable) if it is satisfied by some assignment. The constraint satisfaction problem is the problem of deciding whether a given CSP instance is consistent (resp. satisfiable).

A constraint $C = ((x_1, \ldots, x_n), R)$ is the projection of a constraint $C' = (S', R')$ to $X \subseteq \text{var}(C')$ if $X = \{x_1, \ldots, x_n\}$ and $R$ consists of all tuples $(\sigma(x_1), \ldots, \sigma(x_n))$ for assignments $\sigma$ that are consistent with $C'$. If $C$ is a projection of $C'$, we say that $C$ is obtained from $C'$ by projecting out all variables in $\text{var}(C') \setminus \text{var}(C)$. A constraint $C = ((x_1, \ldots, x_n), R)$ is the join of constraints $C_1, \ldots, C_r$ if $\text{var}(C) = \bigcup_{i=1}^r \text{var}(C_i)$ and if $R$ consists of all tuples $(\sigma(x_1), \ldots, \sigma(x_n))$ for assignments that are consistent with $C_i$ for all $1 \leq i \leq r$.

Let $\mathcal{I} = (V, D, C)$ be a CSP instance and $V' \subseteq V$ be a nonempty subset of variables. The CSP instance $\mathcal{I}[V']$ induced by $V'$ is $\mathcal{I}' = (V', D, C')$, where $C'$ is defined in the following way: For each constraint $C = ((x_1, \ldots, x_k), R)$ having at least one variable in $V'$, there is a corresponding constraint $C'$ in $C'$. Suppose that $x_{i_1}, \ldots, x_{i_l}$ are the variables among $x_1, \ldots, x_k$ that are in $V'$. Then the constraint $C'$ is defined as $((x_{i_1}, \ldots, x_{i_l}), R')$, where the relation $R'$ is the projection of $R$ to the components $i_1, \ldots, i_l$, that is, $R'$ contains an $l$-tuple $(d'_{i_1}, \ldots, d'_{i_l}) \in D'$ if and only if there is a $k$-tuple $(d_1, \ldots, d_k) \in R$ such that $d'_i = d_i$ for $1 \leq i \leq l$. This means that an assignment $\sigma : V' \rightarrow D$ satisfies $\mathcal{I}[V']$ if for each constraint $C$ of $\mathcal{I}$, there is an assignment extending $\sigma$ that satisfies $C$. Note that it is not necessarily true that there is an assignment extending $\sigma$ that satisfies every constraint of $\mathcal{I}$ simultaneously.

Constraints are specified by explicitly enumerating all possible combinations of values for the variables, that is, all tuples in the relation $R$. Consequently, we define the size of a constraint $C = ((x_1, \ldots, x_k), R) \in C$ to be the...
number $|\mathcal{C}| = k + k \cdot R$. The size of an instance $\mathcal{I} = (V, D, \mathcal{C})$ is the number $|\mathcal{I}| = |V| + |D| + \sum_{C \in \mathcal{C}} |C|$.

### 2.1.2 CSP-related Problems

Denote by $\mathbb{Q}_{\geq 0}$ the non-negative rational numbers, i.e., $\mathbb{Q}^+ \cup \{0\}$.

**Definition 2.2.** An instance $\mathcal{I}$ of the maximum constraint satisfaction problem (MAX CSP) is a quadruple $(V, D, \mathcal{C}, w)$, where we have added a weight function, $w : \mathcal{C} \rightarrow \mathbb{Q}_{\geq 0}$ to a plain CSP instance. The objective is to find an assignment to the variables that maximises the weighted sum of satisfied constraints.

**Definition 2.3.** An instance $\mathcal{I}$ of the minimum cost homomorphism problem (MINHOM) is a quadruple $(V, D, \mathcal{C}, \{c_d\}_{d \in D})$, where we have added cost functions, $c_d : V \rightarrow \mathbb{Q}_{\geq 0}$ for each $d \in D$, to a plain CSP instance. The cost of a satisfying assignment $\sigma$ to a MINHOM instance $\mathcal{I}$ is $\sum_{x \in V} c_{\sigma(x)}(x)$, and the objective is to decide whether $\mathcal{I}$ is satisfiable, and if so, determine the minimum cost of a satisfying assignment.

**Definition 2.4.** An instance $\mathcal{I} = (V, D, \mathcal{C}, \Omega)$ of a valued constraint satisfaction problem (VCSP) is a quadruple, where $V$ is a set of variables, $D$ is the domain, $\mathcal{C}$ is a set of constraints of the form $\rho^{x_1 \ldots x_r}$, where $r \geq 1, x_1, \ldots, x_r \in V$, and $\rho$ is an $r$-ary cost function from $D^r$ to $\Omega$, and $\Omega$ is the valuation structure. The valuation structure $\Omega$, is ordered, with a $0$ and $\infty$, and an associative, commutative aggregation operator $\oplus$, where $\alpha \geq \beta, \gamma \in \Omega$ and $\alpha \oplus \gamma \geq \beta \oplus \gamma$. An assignment is a mapping $\sigma : V \rightarrow D$, with associated cost

$$\text{Cost}_\mathcal{I} = \bigoplus_{((x_1, \ldots, x_r), \rho) \in \mathcal{C}} \rho(\sigma(x_1), \ldots, \sigma(x_r)).$$

A solution for $\mathcal{I}$ is an assignment with minimal cost.

Here we will consider the valuation structure $\overline{\mathbb{Q}} = \mathbb{Q} \cup \infty$ with addition. If the range of $\rho$ lies entirely within $\mathbb{Q}$, then $\rho$ is called a finite-valued cost function. If the range of a cost function $\rho$ is $\{0, \infty\}$, then $\rho$ is called a crisp cost function. If the range of a cost function $\rho$ includes non-zero finite costs and infinity, we call $\rho$ a general-valued cost function.

**Definition 2.5.** An instance of a counting constraint satisfaction problem ($\#\text{CSP}$) is a CSP triple $(V, D, \mathcal{C})$. The objective is to count the number of assignments $\sigma : V \rightarrow D$ such that all constraints in $\mathcal{C}$ are satisfied.
2. Preliminaries

2.1.3 Relational Structures and Homomorphisms

A vocabulary $\tau$ is a finite set of relation symbols of specified arities, denoted $ar(\cdot)$. The arity of $\tau$ is $\max\{ar(R) \mid R \in \tau\}$. A $\tau$-structure $A$ consists of a finite set $A$ (called the universe of $A$) and for each relation symbol $R \in \tau$, a relation $R^A \subseteq A^{ar(R)}$. Here, we require both vocabularies and structures to be finite. We say that a class $\mathcal{A}$ of structures is of bounded arity if there is an $r$ such that every structure in $\mathcal{A}$ is at most $r$-ary.

For vocabularies $\sigma \subseteq \tau$, the $\sigma$-reduct of a $\tau$-structure $B$ is the $\sigma$-structure $A$ with universe $A = B$ and $R^A = R^B$ for all $R \in \sigma$. We denote the $\sigma$-reduct of $B$ by $B|_\sigma$. A $\tau$-structure $A$ is an expansion of a $\sigma$-structure $B$ if $B|_\sigma = A$. A substructure of a $\tau$-structure $A$ is a $\tau$-structure $B$ with universe $B \subseteq A$ and relations $R^B \subseteq R^A$ for all $R \in \tau$. A substructure $B$ is induced if for all $R \in \tau$, say, of arity $r$, $R^B = R^A \cap B^r$. We define the size $\|A\|$ of the structure $A$ as $\|A\| = |\tau| + |A| + \sum_{R \in \tau} |R^A| \cdot |ar(R)|$. $\|A\|$ is roughly the size of a reasonable encoding of $A$ [FPG02]. Let $A$ and $B$ be $\tau$-structures. We define $A \cup B$ to be the $\tau$-structure with universe $A \cup B$ and such that for all $R \in \tau$, $R^{A \cup B} = R^A \cup R^B$.

A homomorphism from a $\tau$-structure $A$ to a $\tau$-structure $B$ is a mapping $h : A \rightarrow B$ from the universe of $A$ to the universe of $B$ that preserves all relations, that is, for all $R \in \tau$, say, of arity $k$, and all tuples $(a_1,\ldots,a_k) \in R^A$ it holds that $(h(a_1),\ldots,h(a_k)) \in R^B$. As noted in Chapter 1, homomorphism and constraint satisfaction problems are two sides of the same coin. Feder and Vardi [FV98] observed that a solution to a CSP instance corresponds exactly to a homomorphism from the relational structure of the variables to the relational structure of the possible values for the variables. With every instance $\mathcal{I} = (V,D,C)$ of a CSP we associate two structures $A(\mathcal{I})$ and $B(\mathcal{I})$ as follows: The vocabulary $\tau(\mathcal{I})$ of both structures contains an $r$-ary relation symbol $R$ for every $r$-ary relation symbol $R^\mathcal{I}$ in the constraint language of $\mathcal{I}$. The universe of $B(\mathcal{I})$ is $D$ and the relations of $B(\mathcal{I})$ are those appearing in the constraint language, i.e., for every $R \in \tau(\mathcal{I})$ we let $R^{B(\mathcal{I})} = R^\mathcal{I}$. The universe of $A(\mathcal{I})$ is $V$, and for each $r$-ary relation symbol $R \in \tau(\mathcal{I})$ we let $R^{A(\mathcal{I})} = \{(x_1,\ldots,x_r) \mid ((x_1,\ldots,x_r), R^\mathcal{I}) \in C\}$. Then a function $h : V \rightarrow D$ is a solution for $\mathcal{I}$ if and only if it is a homomorphism from $A(\mathcal{I})$ to $B(\mathcal{I})$. Conversely, if a pair of structures $A$ and $B$ have the same vocabulary, we can construct a corresponding CSP instance $\mathcal{I}$ such that $A(\mathcal{I}) = A$ and $B(\mathcal{I}) = B$.

Now we are in a position to define the restricted CSP-related problems that will be our main concern in Part II of this thesis.

---

**Problem 1.** $\text{MINHOM}(A,B)$, for classes $A,B$ of relational structures, is the restricted $\text{MINHOM}$ problem with:

**Instance:** $A \in A$, $B \in B$, non-negative rational costs $c_b(a)$, where $a \in A$ and $b \in B$.

**Output:** The cost of a minimum cost homomorphism from $A$ to $B$, “no” if no homomorphism from $A$ to $B$ exists.
2.1. Constraint Satisfaction

To simplify the notation, if either $A$ or $B$ is the class of all structures, we just use the placeholder '\_'. We see that restrictions on the class $A$ corresponds precisely to restrictions on the structure induced by the constraints on the variables in the underlying problem.

### Problem 2. VCSP($A, \_\_\_$), for classes $A$ of relational structures, is the structurally restricted VCSP problem with:

**Instance:** VCSP instance with $A(I) \in A$, arbitrary domain $B$, for all $k$, all $k$-ary $R \in \tau(A)$, and all tuples $(a_1, \ldots, a_k) \in R^A$, a cost function $\rho : B^k \rightarrow \bar{Q}$.

**Output:** The minimum cost of an assignment $\sigma : A \rightarrow B$, where $A$ is the universe of $A(I)$.

### Problem 3. #CSP($A, B$), for classes $A, B$ of relational structures, is the restricted #CSP problem with:

**Instance:** #CSP instance with $A(I) \in A$ and $B(I) \in B$.

**Output:** The number of homomorphisms from $A(I)$ to $B(I)$.

It is important to stop here, and think about what we mean when we say that for some class $A$ a problem, such as $\text{MinHom}(A, \_\_\_$), is in polynomial time. If $A$ is not polynomial-time decidable, we view $\text{MinHom}(A, \_\_\_$) as a promise problem. This means that we are only interested in algorithms that work correctly on instances with $A(I) \in A$, and we assume that we are only given such instances. We say that $\text{MinHom}(A, \_\_\_$) is tractable if it is solvable in polynomial time (viewed as a promise problem).

2.1.4 Structural restrictions

In Part II of the thesis, we focus on the so called structural restrictions of our CSP-related problems, i.e., the question of how to restrict the way variables and constraints may interact in the instance, so that the problem is polynomial-time solvable. The usual way to formulate structural restrictions is in terms of certain graphs and hypergraphs that are associated with a problem instance as follows. The primal graph (or Gaifman graph) has the variables as its vertices; two variables are joined by an edge if they occur together in the scope of a constraint. The dual graph instead has the constraints as its vertices; two constraints are joined by an edge if their scopes have variables in common. The incidence graph is a bipartite graph and has both the variables and the constraints as its vertices; a variable and a constraint are joined by an edge if the variable occurs in the scope of the constraint. Finally the constraint hypergraph is a hypergraph\(^1\) whose vertices are variables and whose

---

\(^1\)A hypergraph is just like a graph, except that edges may contain more than two vertices. A formal definition is given in Chapter 2.1.5.
Figure 2.1: Various graphs associated with a constraint satisfaction instance.

edges are the constraint scopes. Figure 2.1(a) gives an example constraint hypergraph and the rest of Figure 2.1 illustrates the three other graph types mentioned above relative to this hypergraph.

The research on structurally restricted CSPs have identified fundamental classes of tractable instances by considering various notions of acyclicity of the associated (hyper)graphs. By using (hyper)graph decomposition techniques acyclicity can be generalized and gives rise to “width” parameters that measure how far an instance is from being acyclic. Freuder [Fre85] and Dechter and Pearl [DP89] observed that the CSP is solvable in polynomial time if

- the tree-width of primal graphs,
2.1. Constraint Satisfaction

is bounded by a constant. In subsequent years several further structural parameters have been considered, such as

- the tree-width of dual graphs,
- the tree-width of incidence graphs,

and various width parameters on constraint hypergraphs, including

- the (generalized) hypertree-width [GLS02],
- the spread-cut-width [CJG08],
- the fractional hypertree-width [GM06], and
- the submodular width [Mar10b].

The remainder of this section will be devoted to exploring tree decompositions and tree-width.

Tree Decompositions

As mentioned above, many hard computational problems become easy when some associated graph has particularly convenient structure. For example, many problems are solvable in polynomial time on trees, i.e., connected undirected graphs without cycles. The reason why this is the case is that a tree is very easy to separate into parts. If we pick any vertex with at least two neighbours, we can delete this vertex from the graph, and in this way create several new, smaller, instances of the problem, which we can solve independently (since we know that there are no edges between the parts left in the graph). This simple idea can be generalised from trees to structures that are, in some sense, similar to trees.

**Definition 2.6.** A tree decomposition of a graph $G$ is a tuple $(T, (B_t)_{t \in V(T)})$, where $T$ is a tree and $(B_t)_{t \in V(T)}$ is a family of subsets of $V(G)$, such that

- **(TD1)** $\bigcup_{t \in V(T)} B_t = V(G)$;
- **(TD2)** for each edge $(u, v) \in E(G)$ there is a node $t \in T$, such that both $u, v \in B_t$; and
- **(TD3)** for each $v \in V(G)$ the set $\{ t \in V(T) \mid v \in B_t \}$ forms a connected subtree of $T$.

The sets $B_t$ are called the bags of the decomposition. The width of a tree decomposition $(T, (B_t)_{t \in V(T)})$ is $\max\{|B_t| \mid t \in V(T)\} - 1$ and the tree-width $tw(G)$ of a graph $G$ is the minimum width over all tree decompositions of $G$.

Deciding whether there exists a tree decomposition of width at most $k$ of a given graph is NP-complete if $k$ is part of the input [ACP87]. If $k$ can be considered a constant, the algorithm by Bodlaender [Bod96] checks in linear time whether or not a tree decomposition of width at most $k$ exists, and if so,
finds such a tree decomposition. However, the asymptotic notation hides a huge constant coefficient that obstructs the practical value of this algorithm. We note that Reed’s algorithm \cite{Ree92} runs in time $O(|V| \log(|V|))$ for any fixed $k$, with a much nicer constant, and decides either that the tree-width of a given graph $G = (V, E)$ exceeds $k$, or outputs a tree decomposition of width at most $4k$ with $O(|V|)$ many nodes.

Figure 2.2 shows a graph and a tree decomposition of the graph. The property (TD3) is sometimes called the interpolation property. It says that if $B_{t_1}$ and $B_{t_2}$ both contain a vertex $v$, then all bags of nodes $t \in V(T)$ in the (unique) path between $t_1$ and $t_2$ contain $v$ as well, and hence can be stated equivalently as:

(TD3) If a node $t_2$ of $T$ lies on the unique path between nodes $t_1$ and $t_3$, then $B_{t_1} \cap B_{t_3} \subseteq B_{t_2}$.

We also make the observation that (TD1) is superfluous if $G$ has no isolated vertices. From an algorithmic point of view, perhaps the most important feature of a tree decomposition is that it transfers the separation properties of its tree to the graph being decomposed.

**Lemma 2.7.** Let $G$ be a connected graph and $(T, (B_t)_{t \in V(T)})$ a tree decomposition of $G$. Furthermore, let $(t_1, t_2)$ be an edge of $T$ and let $T_1, T_2$ be the components of $T - (t_1, t_2)$, with $t_1 \in T_1$ and $t_2 \in T_2$. Then $B_{t_1} \cap B_{t_2}$ separates $G_1 = \bigcup_{t \in V(T_1)} B_t$ from $G_2 = \bigcup_{t \in V(T_2)} B_t$ in $G$.

**Proof.** Since $T$ is a tree, every path from $t$ to $t'$ in $T$ with $t \in T_1$ and $t' \in T_2$ contains both $t_1$ and $t_2$. Hence, by (TD3), $G_1 \cap G_2 \subseteq B_{t_1} \cap B_{t_2}$, so all we need to show is that there is no edge $\{u, v\}$ with $u \in G_1 \setminus G_2$ and $v \in G_2 \setminus G_1$ in $G$. Assume that $\{u, v\}$ is such an edge, then there is a $t \in V(T)$ with $u, v \in B_t$. 

Figure 2.2: A graph and a tree decomposition of width 3.
2.1. Constraint Satisfaction

(by (TD2)). By the choice of $u$ and $v$ we have neither $t \in V(T_1)$ nor $t \in V(T_2)$, a contradiction. Figure 2.3 illustrates this situation.

We will also make frequent (implicit) use of the following basic property of tree decompositions.

Lemma 2.8. Let $G$ be a connected graph and $(T, (B_t)_{t \in V(T)})$ a tree decomposition of $G$. For any clique $G[W]$, with $W \subseteq V(G)$, there is a $t \in V(T)$ such that $W \subseteq B_t$.

Proof. We proceed by induction on the size of the clique. Cliques of size 1 and 2 follow by (TD1) and (TD2). Now assume that $G[W]$ is a clique of size $k + 1$, with $k \geq 2$. There are at least tree nodes in $W$, call them $u, v,$ and $w$. From the induction hypothesis, we know there are nodes $t_1, t_2,$ and $t_3$ in $T$ such that $B_{t_1} \supseteq W \setminus \{u\}$, $B_{t_2} \supseteq W \setminus \{v\}$, and $B_{t_3} \supseteq W \setminus \{w\}$ holds.

Since $T$ is a tree, the three paths between $u, v,$ and $w$ have non-empty intersection. Let $x \in V(T)$ be node in this intersection. By (TD3) the following holds:

$$B_x \supseteq B_{t_1} \cap B_{t_2} \supseteq W \setminus \{u, v\},$$
$$B_x \supseteq B_{t_2} \cap B_{t_3} \supseteq W \setminus \{v, w\},$$
$$B_x \supseteq B_{t_1} \cap B_{t_3} \supseteq W \setminus \{u, w\}.$$

This implies that $B_x \supseteq W$. □

Corollary 2.9. $\text{tw}(K_n) = n - 1$.

Dynamic programming algorithms solving NP-hard problems on graphs of bounded tree-width are often presented on nice tree decompositions. These are rooted tree decompositions in which each node is of one of the following types:

1. Leaf node: a leaf of $T$.

2. Introduce node: an internal node $t$ with one child node $c$ for which $B_t = B_c \cup \{v\}$. This node is said to introduce the vertex $v$.

3. Forget node: an internal node $t$ with one child node $c$ for which $B_t = B_c \setminus \{v\}$. This node is said to forget the vertex $v$. 

Figure 2.3: Illustration for Lemma 2.7.
4. Join node: an internal node $t$ with two child nodes $l$, $r$ with $B_l = B_l = B_r$.

Lemma 2.10 (Klo94). A tree decomposition $(T, (B_i)_{i \in V(T)})$ of a graph $G$ with $n$ nodes can be transformed in time $O(n)$ into a nice tree decomposition $(T', (B'_i)_{i \in V(T')})$ of $G$ which has the same width as $(T, (B_i)_{i \in V(T)})$ and where $T'$ has $O(n)$ nodes.

It is a well-known result that $\text{MAX CUT}$ can be solved in polynomial time on graphs with bounded tree-width [Wim87]. We feel, however, that this result illustrates well the principles in action behind a result of this type and choose to illustrate it briefly. In the following we will assume that we have a nice tree decomposition $(T, (B_i)_{i \in V(T)})$ of width $k$ of the input graph $G$. Furthermore, we assume that $t_r$ is the root of $T$. We associate to each node $i \in T$ the set $\beta_i$ of all vertices in a set $B_i$ with $i = j$ or $j$ a descendant of $i$ in the rooted tree $T$. The algorithm works by computing a table $mc_i$ for every node $t \in T$. For every subset $S$ of $B_i$, there is an entry in the table $mc_i$ satisfying

$$mc_i(S) = \max_{S' \subseteq \beta_i, S' \cap B_i = S} | \{ \{u, v\} \in E(G) \mid u \in S', v \in \beta_i \setminus S'\} |.$$

Stated in words, for $S \subseteq B_i$, $mc_i(S)$ denotes the maximum number of cut edges for a partition of $\beta_i$, such that all vertices in $S$ are in one set of the partition, and all vertices in $B_i \setminus S$ are in the other set of the partition.

The tables are computed in a bottom-up manner; starting with the leaf nodes, and always computing the table for an internal node of the tree later than the tables of its children are computed. The following lemma lets us compute the tables efficiently.

Lemma 2.11. 1. Let $t$ be a leaf node in $T$. Then for all $S \subseteq B_i$, $mc_i(S) = | \{ \{u, v\} \in E(G) \mid u \in S, v \in B_i \setminus S\} |$.

2. Let $t$ be an introduce node with child node $c$. Suppose $B_l = B_c \cup \{v\}, v \notin B_c$. For all $S \subseteq B_l$, if $v \in S$, then $mc_i(S) = mc_c(S \setminus \{v\}) + | \{ \{s, v\} \mid v \in B_l \setminus S\} |$, and if $v \notin S$, then $mc_i(S) = mc_c(S) + | \{ \{s, v\} \mid v \in S\} |$.

3. Let $t$ be a forget node with child node $c$. For all $S \subseteq B_l$, $mc_i(S) = \max_{S' \subseteq B_c, S' \cap B_i = S} mc_c(S')$.

4. Let $t$ be a join node with child nodes $l, r$. For all $S \subseteq B_l$, $mc_i(S) = mc_l(S) + mc_r(S) - | \{ \{u, v\} \in E(G) \mid u \in S, v \in B_l \setminus S\} |$.

Proof. It is easy to see that these conditions hold from the definitions. The subtraction in case (4) is needed because we would double count these edges otherwise. \qed

From Lemma 2.11 we see that computing a table $mc_i$ can be done in constant time. This means we can compute the table of the root $t_r$ in time $O(n)$. The size of the maximum cut in $G$ is $\max_{S \subseteq B_r} mc_r(S)$. 

30
2.1. Constraint Satisfaction

Figure 2.4: A graph and a nice tree decomposition of width 3.

Figure 2.5: The dynamic programming tables computed for the leaf nodes.

**Theorem 2.12.** \( \text{MAX CUT} \) can be solved in polynomial time on graphs with tree-width bounded by a constant.

**Example 2.13.** Figure 2.4 shows a graph and a nice tree decomposition of this graph. We will now illustrate how the algorithm behind Theorem 2.12 puts this tree decomposition to use. The decomposition has three leaf nodes. Starting with the leaf having \( B_t = \{7, 8, 9\} \), what we want to do is simply to compute, for each subset of vertices from the bag, the number of edges that are cut in the graph if this subset is in one partition of the cut and all other vertices of this leaf bag is in the other partition of the cut. So if, for example, the subset \( S \) is the empty set or the while bag, the cut size is obviously zero, while having the vertice 8 in one partition and 7 and 9 in the other partition gives the cut size two. Figure 2.5 shows the tables computed for all three leaf nodes.

The next step of the computation involves three forget nodes. If we look at the node with \( B_t = \{7, 9\} \), we want to compute a table that, for each subset of vertices of \( B_t \), indicates the size of a maximum cut in the subgraph induced by all vertices belonging to this bag or any descendant of it in the decomposition tree, when this subset is in one partition of the cut and all other vertices in the bag are in the other partition of the cut. Fortunately, Lemma 2.11 lets us do just that, as well as take care of any remaining intro-
duce and join nodes. Figure 2.6 shows the resulting tables computed in this manner for the whole decomposition tree. The maximum cut in the graph cuts 16 out of 19 edges. It is possible to modify the algorithm, so that it also yields a partition with the maximum number of cut edges.

For more information on tree decompositions and dynamic programming over tree decompositions, see [BK08]. In Chapter 3, we need good grasp of some common property of graphs with large tree-width. To this end, we turn to the grid graphs.

Grid Graphs and Graph Minors

For \( k, \ell \geq 1 \), the \((k \times \ell)\)-grid is the graph with vertex set \{1, \ldots, k\} \times \{1, \ldots, \ell\} and an edge between \((i, j)\) and \((i', j')\) if and only if \(|i - i'| + |j - j'| = 1\). It is a folklore result that the \((n \times n)\)-grid has tree-width \( n \). To prove this result we would first need a tree decomposition of suitable width. Consider the path on \( B_{n(i-1)+j} = \{(i, k) \mid j \leq k \leq n\} \cup \{(i+1, k) \mid 1 \leq k \leq j\} \), for \( 1 \leq i \leq n-1 \) and \( 1 \leq j \leq n \). It is not hard to see that this construction is a tree decomposition of the \((n \times n)\)-grid of width \( n \). Figure 2.7 illustrates the construction for the \((4 \times 4)\)-grid.

We now turn our attention to a method to prove lower bounds on tree-width, following the terminology of Reed [Ree97]. Two subsets \( W_1, W_2 \subseteq V(G) \) are said to touch if they have a vertex in common or \( E(G) \) contains an edge between them (i.e., \( W_1 \cap W_2 \neq \emptyset \) or there is an edge \((u, v) \in E(G)\) with \( u \in W_1 \) and \( v \in W_2 \)). A set \( B \) of mutually touching connected vertex sets is called a bramble. A subset of \( V(G) \) is said to cover \( B \) if it is a hitting set for \( B \) (i.e., a set which intersects every element of \( B \)). The order of a bramble \( B \) is the minimum size of a hitting set for \( B \). The following theorem is called the Tree-width Minimax Theorem.

**Theorem 2.14** (Seymour and Thomas [ST93]). For any \( k \geq 1 \), \( G \) contains a bramble of order at least \( k \) if and only of its tree-width is at least \( k - 1 \).

With Theorem 2.14 given, all we have left to do is demonstrate a bramble of order \( n + 1 \) for the \((n \times n)\)-grid. Figure 2.8 illustrates such a bramble: We have one set that contains all vertices on the bottom row of the grid, one set that contains all vertices in the last column except the last one, and \((n-1)^2\) crosses, each comprising the first \( n - 1 \) vertices of one of the first \( n - 1 \) rows of the grid and the first \( n - 1 \) vertices of one of the first \( n - 1 \) columns. A set covering the bramble must contain at least \( n - 1 \) vertices to cover the crosses, and one vertex in each of the other two sets.

In a very precise sense, grid graphs are the only example of graphs with large tree-width. To relate to this fact, we have to introduce the concept of a minor of a graph. A graph \( H \) is a minor of a graph \( G \) if \( H \) can be obtained from a subgraph of \( G \) by a sequence of edge contractions, whereby the contraction of \( e = (u, v) \in E(G) \) results in the graph \( G \circ e \) with vertex set \( V(G \circ e) = V(G) \setminus \{v\} \) and edges \( E(G \circ e) = E(G) \setminus \{e \mid v \in e\} \cup \{(u, w) \mid (v, w) \in E(G)\} \).
2.1. Constraint Satisfaction

<table>
<thead>
<tr>
<th>Node</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$mc(\emptyset)$</td>
<td>0</td>
</tr>
<tr>
<td>$mc(7)$</td>
<td>1</td>
</tr>
<tr>
<td>$mc(8)$</td>
<td>2</td>
</tr>
<tr>
<td>$mc(9)$</td>
<td>1</td>
</tr>
<tr>
<td>$mc(12)$</td>
<td>2</td>
</tr>
<tr>
<td>$mc(5,10)$</td>
<td>2</td>
</tr>
<tr>
<td>$mc(10,12)$</td>
<td>2</td>
</tr>
<tr>
<td>$mc(5,10,12)$</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 2.6: The dynamic programming tables computed for all nodes.
2. Preliminaries

Figure 2.7: A tree decomposition of the $4 \times 4$ grid.

Figure 2.8: A bramble of a grid.
A proper minor of a graph $G$ is any minor other than $G$ itself. Figure 2.9 exemplifies the edge contraction operation and finding a minor of a graph.

**Proposition 2.15.** If $H$ is a minor of $G$, then $\text{tw}(H) \leq \text{tw}(G)$.

**Proof.** If $H$ is a subgraph of $G$, then $\text{tw}(H) \leq \text{tw}(G)$, because if we delete all vertices in $G$ outside $H$ in a tree decomposition of minimal width of $G$, we get a tree decomposition of $H$ of width at most $\text{tw}(G)$.

This means the only non-trivial part of the proof corresponds to edge contraction. Assume inductively that $H$ is a graph that came out of $G$ after contracting an edge $(u, v)$ into the new vertex $w$. By replacing every occurrence of $u$ or $v$ by $w$ in the minimal tree decomposition of $G$, we get a tree decomposition of $H$. 

The theory concerning graph minors contains many wonderful results. We choose the following result by Wagner to illustrate the power inherent in this theory. A **planar** graph is a graph that can be drawn on the plane in such a way that its edges intersects only at their endpoints.

**Theorem 2.16** (Wagner [Wag37]). A graph $G$ is planar if and only if $K_5$ and $K_{3,3}$ are not minors of $G$.

The work of Wagner started a long line of efforts to expand on graph minor theory, and in 1984 Neil Robertson and Paul Seymour announced a proof of one of the deepest theorems in combinatorics [RS84]. The details of their proof took two decades, and over five hundred pages in 21 articles, to publish.

**The Graph Minor Theorem** (Robertson and Seymour [RS04]). In any infinite set of graphs, at least one graph is a proper minor of another.

As part of the proof of the Graph Minor Theorem, Robertson and Seymour proved the following theorem which is known as the Excluded Grid Theorem:

**Theorem 2.17** (Robertson and Seymour [RS86]). For every $k$ there exists a $w(k)$ such that the $(k \times k)$-grid is a minor of every graph of tree-width at least $w(k)$.

The best known upper bound for $w(k)$ is currently $20^{2k^5}$ [RST94], although the current conjecture with best support claims that the bound is $\Omega(k^3)$ [DHK09].

### 2.1.5 Hypergraphs and Tree Decompositions

A **hypergraph** is a pair $H = (V(H), E(H))$, consisting of a set $V(H)$ of vertices and a set $E(H)$ of subsets of $V(H)$, the hyperedges of $H$. We always assume that hypergraphs have no isolated vertices, that is, for every $v \in V(H)$ there exists at least one $e \in E(H)$ such that $v \in e$. 


2. Preliminaries

(a) An edge contraction.

(b) An example graph.

(c) Identifying a subgraph.

(d) Identifying edges to contract.

(e) The resulting graph.

(f) $K_5$ is a minor of the example graph.

Figure 2.9: The edge contraction operation, and finding $K_5$ as a minor.
With each relational structure \( A \) we associate a hypergraph \( H(A) \) as follows: The vertex set of \( H(A) \) is the universe of \( A \), and for all \( k \), all \( k \)-ary \( R \in \tau \), and all tuples \((a_1, \ldots, a_k) \in R^A\), the set \( \{a_1, \ldots, a_k\} \) is an edge of \( H(A) \). For a \( \text{MINHOM} \) instance \( \mathcal{I} \), we let \( H(\mathcal{I}) = H(A(\mathcal{I})) \). (Hypergraphs of instances of our various other CSP-related problems are defined analogously.) Note that the vertices of \( H(\mathcal{I}) \) are the variables of \( \mathcal{I} \) and the edges of \( H(\mathcal{I}) \) are the scopes of the constraints of \( \mathcal{I} \).

For a hypergraph \( H \) and a set \( X \subseteq V(H) \), the subhypergraph of \( H \) induced by \( X \) is the hypergraph \( H[X] = (X, \{e \cap X \mid e \in E(H)\}) \). We let \( H \setminus X = H[V(H) \setminus X] \). The Gaifman graph (or primal graph) of a hypergraph \( H \) is the graph \( \mathcal{G} = (V(H), \{v, w\} \mid v \neq w, \text{there exists an } e \in E(H) \text{ such that } \{v, w\} \subseteq e \}) \). A hypergraph is connected if \( \mathcal{G} \) is connected.

A set \( C \subseteq V(H) \) is connected \((\text{in } \mathcal{G})\) if the induced subhypergraph \( H[C] \) is connected. We say that a set \( B \subseteq A \) is connected in a relational structure \( A \) if it is connected in \( H(A) \).

**Definition 2.18.** A tree decomposition of a hypergraph \( H \) is a tuple \((T, (B_t)_{t \in V(T)})\), where \( T \) is a tree and \((B_t)_{t \in V(T)}\) is a family of subsets of \( V(H) \), such that

1. **(TDH1)** for each \( e \in E(H) \) there is a node \( t \in T \) such that \( e \subseteq B_t \) and
2. **(TDH2)** for each \( v \in V(H) \) the set \( \{t \in V(T) \mid v \in B_t\} \) is connected in \( T \).

The sets \( B_t \) are called the bags of the decomposition. The width of a tree decomposition \((T, (B_t)_{t \in V(T)})\) is \( \max \{|B_t| \mid t \in V(T)\} - 1 \). The tree-width \( \text{tw}(H) \) of a hypergraph \( H \) is the minimum of the widths of all tree decompositions of \( H \). It is not hard to see that \( \text{tw}(H) = \text{tw}(\mathcal{G}) \) for all \( H \).

In many of our proofs we need to root the tree in a tree decomposition and view it as directed away from the root. Accordingly, for a node \( t \) in a (rooted) tree \( T \), we define \( T_t \) be the subtree rooted at \( t \), i.e., the induced subtree of \( T \) whose vertices are all those vertices reachable from \( t \).

We remark that by only looking at the primal graph of a hypergraph we might not catch many situations where it would be natural to consider the hypergraph itself to be nearly acyclic in some sense. Indeed, we will see many examples of this situation in Chapter 3. However, if we are in the situation that the hypergraph of an instance \( \mathcal{I} \) of some CSP-related problem has a tree decomposition \((T, (B_t)_{t \in V(T)})\) such that the bags of the decomposition are hyperedges of the hypergraph of \( \mathcal{I} \), then, because of the connectedness condition of tree decompositions \((T, (B_t)_{t \in V(T)})\) is a join tree of the hypergraph of \( \mathcal{I} \). These are two of several equivalent definitions of hypergraph acyclicity \( \text{BFMY83} \). The notion of acyclicity used in this thesis is the most general one occurring in the literature and coincides with \( a \)-acyclicity according to Fagin \( \text{Fag83} \).

Finally, we can now let the tree-width \( \text{tw}(A) \) of a relational structure \( A \) be defined to be the tree-width of its hypergraph \( H(A) \). We say that a class \( \mathcal{A} \) of structures has bounded tree-width if there is a \( k \) such that \( \text{tw}(A) \leq k \) for all
A ∈ A. We shall use a similar terminology for other invariants defined later in the thesis, such as bounded hypertree width, without explicitly defining it.

### 2.2 Approximability

In this section, we review the basic definitions concerning optimisation problems, the basic classes of such problems, and the related reductions.

**Definition 2.19.** A combinatorial optimisation problem Π is defined over a set of instances I_Π; each instance I ∈ I_Π has an associated finite set Sol_Π(I) of feasible solutions. The objective is, given an instance I, to find a feasible solution of optimum value, with respect to some measure (objective function) m_Π(I, f), where f ∈ Sol_Π(I). The optimum of I is denoted by Opt_Π(I), and is defined as the largest measure of any solution to I for maximisation problems and the smallest one for minimisation problems.

All optimisation problems that we consider in this thesis belong to the complexity class NPO.

**Definition 2.20.** The class NPO contains the combinatorial optimisation problems for which the instances and solutions can be recognised in polynomial time, the solutions are polynomially bounded in the input size, and the objective function can be computed in polynomial time. An NPO problem belongs to the class PO if it can be solved to optimality in polynomial time.

Sometimes we will say that an NPO problem is NP-hard. By this we mean that we can reduce an NP-complete problem to it in polynomial time:

**Definition 2.21.** An NPO problem Π_1 is said to be NP-hard if there exists an NP-complete problem Π_2 and a polynomial-time algorithm A which solves Π_2 using queries to an oracle for Π_1.

**Definition 2.22.** A solution f ∈ Sol_Π(I) to an instance I of an NPO problem Π is called r-approximate if it satisfies

\[
\min \left\{ \frac{m_Π(I, f)}{\text{Opt}_Π(I)}, \frac{\text{Opt}_Π(I)}{m_Π(I, f)} \right\} \geq r.
\]

An approximation algorithm for Π has approximation ratio r(n) if, given any instance I of Π, it outputs an r(|I|)-approximate solution.

We say that Π can be approximated within r(n) if there exists a polynomial-time algorithm for Π with approximation ratio r(n).

**Definition 2.23.** An NPO problem is in the class APX (poly-APX) if it can be approximated within a constant factor (a factor polynomial in the input size). If, in addition, there exists an algorithm which, given an instance and a
rational value \( r < 1 \), returns an \( r \)-approximate solution in time polynomial in the size of the instance, then we say that the problem admits a *polynomial-time approximation scheme* (PTAS).

A reduction from an \( \text{NPO} \)-problem \( \Pi_1 \) to an \( \text{NPO} \)-problem \( \Pi_2 \) is specified by two polynomial-time computable functions; \( \phi \) which maps instances of \( \Pi_1 \) to instances of \( \Pi_2 \), and \( \gamma \) which takes an instance \( I \in \Pi_1 \) and a solution \( f \in \text{Sol}_{\Pi_2}(\phi(I)) \) and returns a solution to \( I \). Completeness in \( \text{APX} \) and \( \text{poly-APX} \) can be defined using appropriate reductions, called \( \text{AP} \)-reductions and \( \text{A} \)-reductions, respectively. It is known that an \( \text{APX} \)-complete problem does not admit a PTAS, unless \( \text{P} = \text{NP} \). Every \( \text{AP} \)-reduction is also an \( \text{A} \)-reduction, and every \( \text{A} \)-reduction preserves membership in \( \text{APX} \) [ACGK99].

For a more comprehensive introduction to these classes and their accompanying reductions, see Ausiello, Crescenzi, Gambosi, and Kann [ACGK99] and Crescenzi [Cre97].

### 2.3 Parameterised Complexity

In this section, we review the basic definitions of parameterised problems, the basic classes of such problems, and the related reductions.

**Definition 2.24.** A *parameterised problem* is a set \( \Pi \subseteq \Sigma^* \times \mathbb{N} \), where \( \Sigma \) is a fixed finite alphabet. If \( (x, k) \in \Sigma^* \times \mathbb{N} \) is an instance of a parameterised problem, we call \( x \) the input and \( k \) the parameter.

For example, the parameterised clique problem \( p\text{-CLIQUE} \), is the following problem:

**INPUT:** graph \( G \).

**PARAMETER:** \( k \in \mathbb{N} \).

**OUTPUT:** “Yes” if \( G \) has a clique of size \( k \), “no” otherwise.

Similarly, \( p\text{-INDEPENDENT SET} \) is the parameterised independent set problem with instance \( (G, k) \), where \( G \) is a graph, \( k \in \mathbb{N} \), and we are asked to decide if \( G \) has an independent set of size \( k \).

**Definition 2.25.** A parameterised problem \( \Pi \) over \( \Sigma \) is *(uniformly)* fixed-parameter tractable if there is a computable function \( f : \mathbb{N} \rightarrow \mathbb{N} \), a constant \( c \in \mathbb{N} \) and an algorithm that given \( (x, k) \in \Sigma^* \times \mathbb{N} \) computes the solution in time \( f(k) \cdot |x|^c \). \( \text{FPT} \) denotes the class of all fixed-parameter tractable parameterised decision problems.

**Definition 2.26.** An *fpt-reduction* from a parameterised problem \( \Pi \) over \( \Sigma \) to a parameterised problem \( \Pi' \) over \( \Sigma' \) is a mapping \( \Phi : \Sigma^* \times \mathbb{N} \rightarrow \Sigma'^* \times \mathbb{N} \) which transforms an instance \( (x, k) \) of \( \Pi \) to an instance \( (x', k') \) of \( \Pi' \), such that \( (x, k) \in \Pi \iff (x', k') \in \Pi' \), \( \Phi \) is computable in time \( f(k) \cdot |x|^c \) and \( x' \leq g(x) \) (for computable functions \( f, g : \mathbb{N} \rightarrow \mathbb{N} \) and a constant \( c \)).

There is also a natural notion of a parameterised Turing reduction:
Definition 2.27. A parameterised Turing reduction from a parameterised problem \( \Pi \) over \( \Sigma \) to a parameterised problem \( \Pi' \) over \( \Sigma' \). Such a reduction is an algorithm \( A \) with an oracle for \( \Pi' \) such that \( A \) is an fpt-algorithm (That is, there is a computable function \( f : \mathbb{N} \to \mathbb{N} \) such that the running time of \( A \) on input \((x, k)\) is bounded by \( f(k) \cdot |x|^c \)) and there is a computable function \( g : \mathbb{N} \to \mathbb{N} \) such that all inputs \((x', k')\) to the oracle satisfies \( k' \leq g(k) \) (when \((x, k)\) is the \( \Pi \) instance).

We note that \( \text{FPT} \) is closed under both fpt-reductions and fpt Turing reductions \([\text{FG06}]\). This is complemented by the work of Downey and Fellows \([\text{DF95a}]\) who defined a hierarchy \( \text{W}[1] \subseteq \text{W}[2] \subseteq \cdots \subseteq \text{W}[\text{P}] \) of parameterised complexity classes called the \emph{weft hierarchy}. Each class is the equivalence class of certain parameterised satisfiability problems under fpt-reductions. For instance, asking whether a given 3SAT instance can be satisfied by setting at most \( k \) variables to \emph{true}, is the canonical \( \text{W}[1] \)-complete problem. This way, parameterised complexity offers a completeness theory similar to the theory of \( \text{NP} \)-completeness.

The class \( \text{XP} \) consists of parameterised problems which can be solved in polynomial time if the parameter is considered as a constant. The parameterised classes above form the chain

\[
\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \cdots \subseteq \text{W}[\text{P}] \subseteq \text{XP}
\]

where all inclusions are believed to be proper. It is known that \( \text{FPT} \neq \text{XP} \) \([\text{DF99}]\), and a parameterised analogue of Cook’s Theorem \([\text{DF99}]\) as well as the Exponential Time Hypothesis \([\text{FG06}; \text{IPZ01}]\) give strong evidence that the assumption \( \text{FPT} \neq \text{W}[1] \) is true. In particular, we know that \( \text{FPT} \neq \text{W}[1] \) implies \( \text{P} \neq \text{NP} \) and that \( \text{FPT} = \text{W}[1] \) would imply that 3SAT is solvable in time \( 2^{o(n)} \). Downey and Fellows \([\text{DF95b}]\) have shown that \( \text{p-CLIQUE} \) and the parameterised independent set problem, \( \text{p-INDEPENDENT SET} \), are \( \text{W}[1] \)-complete under fpt-reductions. These theorems are used in our hardness proofs in Chapter 3 and Chapter 5. For a formal exposition of parameterised complexity, see \([\text{DF99}; \text{FG06}]\).
PART II

Structural Restrictions
In this chapter we start exploring the computational complexity of various CSP-related problems. In particular, we look at structurally restricted problems, with the additional constraint that the arities of all involved classes of relational structures are bounded by constants.

3.1 Introduction

As mentioned in Chapter 1 of the thesis, Feder and Vardi [FV98] observed that constraint satisfaction problems can be described as homomorphism problems for relational structures. Recall that, for every two classes of relational structures \( \mathcal{A} \) and \( \mathcal{B} \), \( \text{HOM}(\mathcal{A}, \mathcal{B}) \) is the problem of deciding whether a structure \( A \in \mathcal{A} \) has a homomorphism to a given arbitrary structure \( B \in \mathcal{B} \).

Grohe [Gro07] has studied structural restrictions, i.e. the question of how to restrict \( \mathcal{A} \), so that \( \text{HOM}(\mathcal{A}, \_ ) \) is polynomial-time solvable. He proves the following:

Assume \( \text{FPT} \neq \text{W[1]} \). Then for every recursively enumerable class \( \mathcal{A} \) of structures of bounded arity, \( \text{HOM}(\mathcal{A}, \_ ) \) is polynomial-time solvable if and only if the core of every structure in \( \mathcal{A} \) has tree-width at most \( w \) (for some fixed \( w \)).

A core of a relational structure \( \mathcal{A} \) is a substructure \( \mathcal{A}' \subseteq \mathcal{A} \) such that there is a homomorphism from \( \mathcal{A} \) to \( \mathcal{A}' \), but no homomorphism from \( \mathcal{A}' \) to a proper substructure of \( \mathcal{A}' \). All cores of a structure \( \mathcal{A} \) are isomorphic, so it is reasonable to speak of the core of \( \mathcal{A} \). If we let \( \mathcal{A} \) be a triangle and \( \mathcal{B} \) is the disjoint union of an arbitrary graph \( G \) with \( \mathcal{A} \), we see that \( \mathcal{A} \) is a core of \( \mathcal{B} \) if and only if \( G \) is 3-colourable. This shows that it is NP-hard to decide, given structures.
3. Bounded Arity

$A \subseteq B$, whether $A$ is isomorphic to the core of $B$. Hell and Nešetřil [HN92] have shown that it is co-NP-complete to decide if a graph is a core.

In the list homomorphism problem $\text{LHOM}(A, B)$, the goal is to decide whether there is a homomorphism from a structure $A \in A$ to a given structure $B \in B$, when each element in $A$ is only allowed a certain subset of elements in the universe of $B$ as its image. Such list homomorphisms generalise e.g. list colourings and have many natural applications. We show the following:

Assume that $\text{FPT} \neq \text{W[1]}$. Then for every recursively enumerable class $A$ of structures of bounded arity, $\text{LHOM}(A, \_)$ is polynomial-time solvable if and only if every structure in $A$ has tree-width at most $w$ (for some fixed $w$).

Incidentally, this complexity-theoretic classification coincides with that of Dalmau and Jonsson’s in [DJ04], where they study the problem of counting homomorphisms (i.e., in essence, $\text{#CSP}$). Our result is then extended to the connected list homomorphism problem [FH98], where every list has to induce a connected substructure of the right hand side input structure and to the edge list homomorphism problem [FH07], where the lists contain tuples from the relations of the right hand side input structure that the tuples on the left hand side have to map to.

Furthermore, we are able to show that our hardness result extends to three optimisation problems. Two of these are $\text{MINHOM}$ and $\text{VCSP}$, and the hard instances of the minimum cost homomorphism problem are also shown to be inapproximable as well. The third optimisation problem is the maximum solution problem [JKN08; JN08]. In this problem, the right hand side elements are assumed to be a finite subset of the natural numbers and the objective is to find a homomorphism that has maximum possible total weight. In some sense, see [JN08], this is a generalisation of integer programming and captures e.g. the $\text{INDEPENDENT SET}$ problem. When the right hand side is restricted to $\{0, 1\}$ this is the well-studied $\text{MAXONES}$ problem.

The rest of this chapter is organised as follows. Section 3.2 introduces the requisite background material and problem definitions for several variants of the homomorphism problem. Section 3.3 contains proofs of our (in)tractability and inapproximability results. Finally, Section 3.4 concludes the chapter and presents possible future work.

3.2 Homomorphism and Constraint Satisfaction Problems

In the following, we formally define those restricted homomorphism problems which will be our main concern in Section 3.3 and that were not formally defined in Chapter 2.

In the list homomorphism problem, $\text{LHOM}(A, B)$, each element of the left hand side input structure is given together with a set, called a list, of possible images in the right hand side input structure. This problem is well studied with regard to restrictions to the right hand side input structure, see e.g. [Bul11; FH98; FH06; FHH99; FHH03; FHH07; FHHR12; HR11].
3.2. Homomorphism and Constraint Satisfaction Problems

**Problem 4.** LHOM\((A, B)\) is the problem with:

- **Instance:** \(A \in A, B \in B, L_a \subseteq B\) for each \(a \in A\).
- **Output:** “yes” if a homomorphism \(h\) from \(A\) to \(B\) such that \(h(a) \in L_a\) for each \(a \in A\) exists, “no” otherwise.

By restricting LHOM\((A, B)\) to those inputs in which each list \(L_a\) induces a connected subgraph of the Gaifman graph \(G(B)\) of \(B\), we get the connected list homomorphism problem, introduced for graphs by Feder and Hell [FH98].

**Problem 5.** CLHOM\((A, B)\) is the problem with:

- **Instance:** \(A \in A, B \in B, L_a \subseteq B\) for each \(a \in A\), such that each \(L_a\) induces a connected substructure in \(B\).
- **Output:** “yes” if a homomorphism \(h\) from \(A\) to \(B\) such that \(h(a) \in L_a\) for each \(a \in A\) exists, “no” otherwise.

Feder and Hell [FH07] introduce the edge list homomorphism problem for undirected graphs. Here we generalise this problem to arbitrary relational structures.

**Problem 6.** ELHOM\((A, B)\) is the problem with:

- **Instance:** \(A \in A, B \in B\), lists of tuples from the relations of \(B\) for each tuple of the relations in \(A\).
- **Output:** “yes” if a homomorphism \(h\) from \(A\) to \(B\) such that each tuple in the relations of \(A\) maps to a tuple in the corresponding list of tuples from \(B\) exists, “no” otherwise.

If we let the universes \(B\) of the right hand side input structures of HOM\((A, B)\) be finite subsets of the natural numbers equipped with the usual total order \(<\), the maximum solution problem [JKN08; JN08] is the following problem.

**Problem 7.** MAX SOL\((A, B)\) is the problem with:

- **Instance:** \(A \in A, B \in B\), weight function \(w : A \rightarrow \mathbb{N}\)
- **Output:** The maximum of \(\sum_{a \in A} w(a) \cdot h(a)\) for any homomorphism \(h\) from \(A\) to \(B\), “no” if no homomorphism from \(A\) to \(B\) exists.

We note that MAX SOL is an extension of the well studied MAX ONES problem [KSTW06], and, as such, captures e.g. the INDEPENDENT SET problem. As in [KSTW06], where Khanna, Sudan, Trevisan, and Williamson classify the approximability of MAX ONES with respect to restrictions to the right hand side input structure, we restrict our attention to instances of MAX SOL.
3. **Bounded Arity**

satisfying the following restriction: if \( a, a' \) occur in the same tuple \((a_1, \ldots, a_r)\) in some relation in \( A \), then \( a \neq a' \) must hold. We say that a structure having this property is **replication free**.

In the forthcoming hardness proofs, we use the tool box of parameterised complexity theory. Therefore we need to define appropriate parameterised versions of our homomorphism problems. The parameterisations we are interested in equip our homomorphism problems with a parameter defined as the size of the left hand side input structure. E.g. we have the following definition of the parameterised list homomorphism problem, \( p\text{-LHOM}(A, B) \):

**INPUT:** \( A \in A, B \in B, L_a \subseteq B \) for each \( a \in A \).

**PARAMETER:** \(||A|||.

**OUTPUT:** “yes” if a homomorphism \( h \) from \( A \) to \( B \) such that \( h(a) \in L_a \) for each \( a \in A \) exists, “no” otherwise.

The parameterised versions of the other homomorphism problems addressed in this chapter are defined analogously and with the same parameter.

### 3.3 Main Results

We are now ready to prove the main results. First, we make the observation that when our homomorphism problems are restricted to classes of structures that have bounded tree-width, standard techniques using tree-decompositions, cf. [Hel03; BK08], may be deployed to solve these problems in polynomial time. Essentially, these techniques use the same methodology as in the \textsc{Max Cut} example in Chapter 2.1.4 with appropriate modifications to suit the problem type. Then we see that what is left to do to get a classification of our problems, with regard to structural restrictions, is to prove hardness for classes of structures with unbounded tree-width. The proofs need a bit of preparation, which is taken care of in Section 3.3.1, Section 3.3.2 and Section 3.3.3 contain the actual proofs for our decision problems and optimisation problems, respectively.

#### 3.3.1 The Structure \( B \)

In what follows, we will need an alternative definition of the notion of a graph minor.

**Definition 3.1.** A **minor map** from a graph \( H \) to a graph \( G \) is a mapping \( \mu : V(H) \to 2^{V(G)} \) having the following properties:

1. for all \( v \in V(H) \), the graph \( G[\mu(v)] \) is non-empty and connected;
2. for all \( v, w \in V(H) \) with \( v \neq w \), the sets \( \mu(v) \) and \( \mu(w) \) are disjoint; and
3. for all edges \((u, v) \in E(H)\), there are \( u' \in \mu(u) \) and \( v' \in \mu(v) \) such that \((u', v') \in E(G)\).
We call a minor map \( \mu \) from \( H \) to \( G \) onto if \( \bigcup_{v \in V(H)} \mu(v) = G \).

**Lemma 3.2.** There is a minor map from a graph \( H \) to a graph \( G \) if and only if \( H \) is a minor of \( G \).

**Proof.** To get the mapping \( \mu \): Assume that a vertex \( w \) of \( V(H) \) is the result of contracting edges \( e_1, \ldots, e_k \) of \( G \). Then those edges form a connected subgraph of \( G \). These subgraphs are vertex-disjoint for different vertices \( w \). Define \( \mu(w) \) as the set of vertices \( e_1, \ldots, e_k \). We prove the third property by contradiction: If no edge \( (u', v') \) exists, then it is impossible to obtain the edge \( (u, v) \) by a series of contractions.

In the other direction: We have the mapping \( \mu \); first we take the subgraph in \( G \) induced by \( \bigcup_{u \in V(H)} \mu(u) \). Now each \( \mu(u) \) can be contracted into a single vertex, and analogously, when \( \mu(u) \) and \( \mu(v) \) are disjoint, the contraction of \( \mu(u) \) yields a different vertex than \( \mu(v) \). Hence, after these two steps we get a graph which is a minor of an induced subgraph of \( G \) which can be further reduced, by removing edges, to a minor isomorphic to \( H \). \( \Box \)

It is easy to see that, if \( H \) is a minor of a connected graph \( G \), then we can always find a minor map from \( H \) onto \( G \).

Let \( A \) be a connected \( \tau \)-structure. Let \( k \geq 2 \), \( K = \binom{\frac{k}{2}}{2} \), and \( \mu : \{(1, \ldots, k) \times \{1, \ldots, K\}\} \rightarrow 2^A \) a minor map from the \( (k \times K) \)-grid onto \( H(A) \) (the Gaifman graph of \( A \)). Let us assume that we have fixed some bijection \( \rho \) between \( \{1, \ldots, k\} \) and the set of all unordered pairs of elements of \( \{1, \ldots, k\} \).

Let \( G \) be a graph. We now concentrate on the \( \tau \)-structure \( B = B(A, \mu, G) \), as defined by Grohe \([\text{Gro07}]\). The universe \( B \) of \( B \) is given by:

\[
\{(v, e, i, p, a) \mid v \in V(G), e \in E(G), 1 \leq i \leq k, 1 \leq p \leq K \text{ s.t. } (v \in e \iff i \in \rho(p)), a \in \mu(i, p)\}
\]

We define the projection function \( \Pi : B \rightarrow A \) by letting \( \Pi(v, e, i, p, a) = a \). As usual, we extend \( \Pi \) and \( \Pi^{-1} \) to tuples by defining it component-wise.

The relations we will add to \( B \) might seem totally ad hoc, but their purpose are to equate the presence of a \( k \)-clique in \( G \) with the existence of a homomorphism from \( A \) to \( B \) having certain desirable properties, as we will see shortly. For every relation \( R \in \tau \), say, of arity \( r \), and for all tuples \( (a_1, \ldots, a_r) \in R^A \), we add to \( R^B \) all tuples \( (b_1, \ldots, b_r) \in \Pi^{-1}(a_1, \ldots, a_r) \) satisfying the following two constraints for all \( b, b' \in \{b_1, \ldots, b_r\} \):

(C1) if \( b = (v, e, i, p, a) \) and \( b' = (v', e', i, p', a') \), then \( v = v' \); and

(C2) if \( b = (v, e, i, p, a) \) and \( b' = (v', e', i', p, a') \), then \( e = e' \).

In the remainder of this chapter, we will focus on homomorphisms from \( A \) to \( B \) such that each \( a \in A \) is mapped to an element \( b \in B \) that was "generated" by \( a \), i.e. \( b \in \Pi^{-1}(a) \). We will denote this by saying that for a homomorphism \( h : A \rightarrow B \), \( h(a) = (\ldots, \ldots, a) \) for each \( a \in A \), where the
placeholders ‘_’ are used to indicate that the values in question are arbitrary, as long as the element is a member of $B$. To proceed we need the following fact:

**Lemma 3.3.** The graph $G$ contains a $k$-clique if and only if there exists a homomorphism $h$ from $A$ to $B$ such that $h(a) = (\_ , \_ , \_ , a)$ for all $a \in A$.

*Proof.* In the proof of Lemma 3.1 in [DJ04] it is shown that the graph $G$ contains a $k$-clique if and only if there exists a homomorphism $h$ from $A$ to $B$ satisfying $\Pi \circ h = \text{id}$, where id is the identity function on the set $A$. Now, if $h$ is a homomorphism from $A$ to $B$ such that $h(a) = (\_ , \_ , \_ , a)$ for all $a \in A$, $h$ obviously satisfies $\Pi \circ h = \text{id}$ and vice versa. \qed

In our hardness proofs we heavily use the structure $B$ and sometimes also modify it. Therefore, in order to have a good intuition for the properties of the structure $B$, we give an example.

**Example 3.4.** Let $k = 3$ and $\mu$ a minor map from the $(3 \times 3)$-grid onto $A$. Assume that, with labels as in Figure 3.1, $A$ is isomorphic to the $(3 \times 3)$-grid (i.e. $\mu(i, p) = (\mu(i, p))$ in the figure) and that the graph $G$ is $K_3$. Fix $\rho = \{1 \mapsto \{1, 2\}, 2 \mapsto \{1, 3\}, 3 \mapsto \{2, 3\}\}$.

Take a look at Figure 3.2 where $B$ is depicted; what does the universe of $B$ look like? Well, take the element of $A$ pointed out by $\mu(1, 1)$, with label $\mu(1, 1)$, for example. By definition, the 5-tuples in $B$ “generated” by this element have $i = 1$ and $p = 1$, i.e. $i \in \rho(p)$ and thus all pairs $v \in V(G), e \in E(G)$ in $G$, where $v \in e$, followed by $i$, $p$ and $\mu(1, 1)$ form a part of $B$. $\mu(1, 3)$, on the other hand, yields elements of $B$ with $i = 1$ and $p = 3$, i.e. $i \notin \rho(p)$. This means that $\{(a, bc, 1, 3, \mu(1, 3)), (b, ac, 1, 3, \mu(1, 3)), (c, ab, 1, 3, \mu(1, 3))\}$ is the subset of $B$ generated by $\mu(1, 3)$. We see that one property of the bijection $\rho$ is that exactly one element in each row and column of $A$ generates three elements in $B$, while the other two elements in each row and column generate six elements each in $B$.

What tuples are members of the relations of $B$? In our example, $A$ only has binary relations. Let us focus on just one element of $B$, namely $b^* = (a, ab, 1, 1, \mu(1, 1))$. The possible members of $B$, that $b^*$ may be related to, are those looking something like $(v, e, i, p, \mu(1, 2))$ or $(v, e, i, p, \mu(2, 1))$ and
out of the twelve possibles, only four actually take part in a tuple of a relation of $B$ together with $b^*$, namely $(a, ab, 1, 2, \mu(1, 2)), (a, ac, 1, 2, \mu(1, 2)), (a, ab, 2, 1, \mu(2, 1))$ and $(b, ab, 2, 1, \mu(2, 1))$. In fact, each element in $B$ will have four neighbours in the Gaifman graph of $B$, except those elements generated by $\mu(2, 2)$, that will have eight neighbours each.

In Figure 3.2 we have also highlighted the co-domain of a homomorphism $h$ from $A$ to $B$, with $h(a) = (_{-}, _{-}, _{-}, _{-}, a)$ for all $a \in A$, indicating the presence of a 3-clique in $G$ according to Lemma 3.3, by making the border of those elements in $B$ in the co-domain of $h$ have rounded corners. The interested reader might want to verify that there are indeed five more homomorphisms of this kind from $A$ to $B$.

### 3.3.2 Hardness Results for Decision Problems

The problem $p$-LHOM$(A, \_)$ is trivially in $\text{FPT}$ when LHOM$(A, \_)$ is in $\text{P}$, and we know that LHOM$(A, \_)$ is solvable in polynomial time if the structures in $A$ have bounded tree-width. What is left to prove, to achieve the result announced in Section 3.1, is that if $p$-LHOM$(A, \_)$ is in $\text{FPT}$, then the structures in $A$ have bounded tree-width. We do this by assuming that $p$-LHOM$(A, \_)$ is in $\text{FPT}$ even when $A$ has unbounded tree-width and showing that this implies $p$-CLIQUE is in $\text{FPT}$, in contradiction with the fact that it is $\text{W}[1]$-complete. This is accomplished by exhibiting an fpt-reduction from $p$-CLIQUE to $p$-LHOM$(A, \_)$, where the structure $B$ and Lemma 3.3 from the previous subsection play essential roles. As the same line of reasoning applies to the other problems under study, this proof is then adapted and extended to fit our different problem variations.

**LHOM**

**Lemma 3.5.** Let $A$ be a recursively enumerable class of structures of bounded arity that does not have bounded tree-width. If $p$-LHOM$(A, \_)$ is in $\text{FPT}$, then $\text{FPT} = \text{W}[1]$.

**Proof.** Let $(G, k)$ be an instance of $p$-CLIQUE, with $k \geq 1$ and $K = \binom{k}{2}$. By the Excluded Grid Theorem, there is some structure $A$ in $A$ such that the $(k \times K)$-grid is a minor of the Gaifman graph of $A$. We enumerate the recursively enumerable class $A$ until we find such an $A = A(k)$. Then we compute a minor map $\mu$ from the $(k \times K)$-grid to the Gaifman graph of $A$. Let $A_1, \ldots, A_m$ be a decomposition of $A$ into its connected components. We can assume, without loss of generality, that the $(k \times K)$-grid is a minor of (the Gaifman graph of) $A_1$ and that the minor map $\mu$ is onto $A_1$.

Let $B = (A, \mu, G)$ be constructed as above. By Lemma 3.3 we know that in order to decide if there exists a $k$-clique in $G$ we only need to check if there is a homomorphism $h$ from $A_1$ to $B$ such that $h$ maps every $a \in A_1$ to some $(\_, \_, \_, a) \in B$, since such an $h$ exists if and only if $G$ has a $k$-clique. We would like to differentiate $B$, so that only homomorphisms mapping $a \in A_1$
Figure 3.2: An example of the structure $B$. 

3. Bounded Arity
to \((\ldots, \ldots, a) \in B\) are allowed. Fortunately, the list homomorphism problem lets us enforce precisely such a differentiation of \(B\).

To do this construct \(B'\) as \(B \cup A_2 \cup \cdots \cup A_m\) and lists \(L_a \subseteq B', a \in A\) defined by:

\[
L_a = \begin{cases} 
\{ b \mid b \in B \text{ and } b = (\ldots, \ldots, a) \} & \text{if } a \in A_1 \\
\{ b \mid b \in B' \setminus B, b = a \} & \text{otherwise}
\end{cases}
\]

This way, we will always be able to find a homomorphism from \(A \setminus A_1\) to \(B' \setminus B\); it is just a matter of selecting the only element \(b\) available in \(L_a\) for each \(a \in A \setminus A_1\). Since \(b = a\) in each case this obviously results in a homomorphism from \(A \setminus A_1\) to \(B' \setminus B\).

It is also clear that the only possible homomorphisms \(h\) from \(A_1\) to \(B\) (and hence also the only possible homomorphisms from \(A\) to \(B'\)), under our lists, are the ones obeying the condition that \(h\) maps each \(a \in A_1\) to some \((\ldots, \ldots, a) \in B\), due to the definition of the lists for elements \(a \in A_1\).

Thus, the conclusion is that if \(G\) contains a \(k\)-clique, then we will be able to find a homomorphism from \(A\) to \(B'\), since then a homomorphism \(h\) from \(A_1\) to \(B\), obeying \(h(a) = (\ldots, \ldots, a)\) for each \(a \in A_1\), exists (by Lemma 3.3). If \(G\) has no \(k\)-clique, then we will not be able to find any homomorphism from \(A\) to \(B'\).

The construction of \(A\) only depends on \(k\) and is effective because \(\mathcal{A}\) is recursively enumerable. Computing the minor map \(\mu\) may require exponential time in the size of \(A\), but this still bounded in terms of \(k\). The size of an \(r\)-ary relation \(R^B\) is at most \(|\Pi^{-1}(A^r)| \leq (|V(G)| \cdot |E(G)| \cdot |A|)^r\). This is polynomial in \(|A|\) and \(|G|\) since the arity of \(\mathcal{A}\) is bounded. It follows that the size of \(B\) and \(B'\) is polynomially bounded in terms of \(|A|\) and \(|G|\) and so, \(B'\) can be computed in polynomial time. The lists \(L_a\) for \(a \in A \setminus A_1\) are easy to compute and only hold one element each. While generating \(B\) it is easy to construct the lists \(L_a\) for \(a \in A_1\) and the size of these lists are linear in the size of \(B\). This shows that the reduction from \((G, k)\) to \(A, B', L_a\) is an fpt-reduction.

An immediate consequence of the above is that the problem of counting list homomorphisms [HN04a] is \(\text{W[1]}\)-hard when \(\mathcal{A}\) does not have bounded tree-width. If we let \(\text{#LHOM}(\mathcal{A}, \_\_)\) be the problem with:

\[\text{INSTANCE: } A \in \mathcal{A}, B \in B, L_a \subseteq B, \text{ for each } a \in A.\]

\[\text{OUTPUT: } \text{The number of homomorphisms } h \text{ from } A \text{ to } B \text{ such that } h(a) \in L_a \text{ holds for each } a \in A.\]

we also have the following result.

**Lemma 3.6.** Let \(\mathcal{A}\) be a class of relational structures of bounded tree-width. Then \(\text{#LHOM}(\mathcal{A}, \_\_)\) is solvable in polynomial time.

**Proof.** In Theorem 4.4 we will prove that if \(\mathcal{A}\) is a class of relational structures of bounded \textit{generalised hypertree width}, then the problem of counting homomorphisms from a structure in \(\mathcal{A}\) to a given arbitrary structure is solvable.
in polynomial time. Inspecting the algorithm behind this result, we see that, by making trivial modifications to it, we can extend this algorithm to be able to take into account the lists of #LHOM. In Chapter 4 we will also see that if \( A \) has bounded generalised hypertree width, then \( A \) necessarily also has bounded tree-width.

Putting everything together, we can now conclude that for every recursively enumerable class \( A \) of structures of bounded arity, \#LHOM(\( A, _- \)) is polynomial-time solvable if and only if \( A \) has bounded tree-width (assuming FPT \( \neq \) W[1]).

The fpt-reduction from p-CLIQUE to p-LHOM(\( A, _- \)) utilised in the proof of Lemma 3.5 is actually called a uniform fpt-reduction and an inspection of the proof shows that we need the recursive enumerability of the class \( A \) only to guarantee the uniformity of this reduction. If we strengthen the complexity theoretic assumption to nonuniform-FPT \( \neq \) nonuniform-W[1], we can prove the theorem for arbitrary classes of bounded arity. For more about nonuniform fixed-parameter tractability, cf. [DF99].

Before we continue dealing with our hardness results, a remark about our chosen proof method is in place. Why do we need to use the structure \( B \) at all, could we not just reduce HOM(\( A', _- \)) to LHOM(\( A, _- \)) for some suitable class \( A' \)? Indeed, this is the case [Dal08]. However, we feel that the preceding proof clearly and cleanly illustrates the way we use Grohe’s structure in the forthcoming hardness proofs, in particular the proofs for the optimisation problems. For completeness, we sketch the alternative hardness proof for LHOM(\( A, _- \)). Let \( A \) be a class of structures as in Lemma 3.5. For every structure \( A \) we let \( A' \) be the structure obtained by adding a new unary relation \( R_a \) for each \( a \in A \). Furthermore this new relation contains in \( A' \) exactly \( \{(a)\} \). (In other words, we add a constant for each element of \( A \).) Then one can prove that HOM(\( A', _- \)) reduces to LHOM(\( A, _- \)), where \( A' \) is the class containing \( A' \) for every \( A' \in A \). Since every \( A' \) is a core, HOM(\( A', _- \)) is W[1]-hard. Let \( (A', B) \) be an instance of HOM(\( A', _- \)). An equivalent instance of LHOM(\( A, _- \)) can be constructed in the following way: The left hand structure of the instance is \( A \) and the right hand side structure of the instance is precisely the reduct of \( B \) obtained by removing all relations \( R_a \). Finally we assign to every element \( a \in A \) the list \( R^B_a \).

**CLHOM**

To be able to prove hardness for CLHOM we will have to modify the structure \( B \) a bit; by adding some dummy elements to \( B \) we will make our lists of elements in \( B \) induce connected substructures of \( B \) to achieve the following:

**Lemma 3.7.** Let \( A \) be a recursively enumerable class of structures of bounded arity that does not have bounded tree-width. If p-CLHOM(\( A, _- \)) is in FPT, then FPT = W[1].
3.3. Main Results

Proof. We start out as in the proof of Lemma 3.5 but instead of using 
\( B = (A, \mu, G) \), we use a new structure \( B_+ \). \( B_+ \) is \( B \) augmented with a 
dummy element \( b_a \) for each \( a \in A_1 \). Furthermore, we want the property 
that each \( b_a \) is related to every \( b \in \Pi^{-1}(a) \). For each \( b_a \), this is 
accomplished by adding to every relation of \( B \), say of arity \( r \), for every 
\( b \in \Pi^{-1}(a) \) the tuple \((b_a, b_{(1)}, \ldots, b_{(r-1)})\), where each \( b_{(i)} = b \). As usual, we let \( B'_+ = B_+ \cup (A \setminus A_1) \).

Now we can construct our lists; 
\[ L_a = \begin{cases} 
\{b \mid b \in B \text{ and } b = (\ldots, a, \ldots)\} \cup b_a & \text{if } a \in A_1 \\
\{b \mid b \in B'_+ \setminus B, b = a\} & \text{otherwise}
\end{cases} \]

This way, each \( L_a \) induces a connected subgraph of the Gaifman graph of 
\( B'_+ \), thus ensuring that we indeed have an instance of CLHOM at hand. (For 
\( a \in A \setminus A_1, L_a \) just holds one element of \( B'_+ \) and is trivially connected in the 
Gaifman graph of \( B'_+ \).)

Now, since \( A_1 \) is connected, we know that each \( a \in A_1 \) has at least one 
neighbour in the Gaifman graph of \( A_1 \). This, coupled with the fact that each 
\( b_a \) only is related to a certain set of \( b \in B_+ \) (our preferred choices when 
it comes to images in a potential homomorphism) and that these sets are 
disjoint leads us to the conclusion that \( B'_+ \) neither adds nor subtracts any 
potential homomorphisms when compared to potential homomorphisms from 
\( A \) to \( B' \) in the hardness proof for LHOM. Furthermore, we can add the \( b_a \) and 
their related tuples to \( B'_+ \) in time polynomial in terms of \( |A| \) and \( |G| \). \( \blacksquare \)

**ELHOM**

Lemma 3.8. Let \( A \) be a recursively enumerable class of structures of bounded arity 
that does not have bounded tree-width. If \( p\text{-ELHOM}(A, \_\) is in \( \text{FPT} \), then \( \text{FPT} = \text{W}[1] \).

Proof. We proceed exactly as in the proof of Lemma 3.5. By letting the edge 
lists for the tuples \((a_1, \ldots, a_r)\) in the relations of \( A_1 \) contain tuples \((b_1, \ldots, b_r)\) 
from the relations of \( B \), where \((b_1, \ldots, b_r) = (\ldots, a_1, \ldots, \ldots, a_r)\) 
we have restricted the potential homomorphisms in the same way as in the 
proof of Lemma 3.5. \( \blacksquare \)

3.3.3 Hardness Results for Optimisation Problems

In this section, we take care of the \( \text{W}[1] \)-hardness proofs for our optimisation 
problems. The proof outline is the same as the one used for the decision 
problems in Section 3.3.2.

**MINHOM**

Lemma 3.9. Let \( A \) be a recursively enumerable class of structures of bounded arity 
that does not have bounded tree-width. If \( p\text{-MINHOM}(A, \_\) is fixed-parameter 
tractable, then \( \text{FPT} = \text{W}[1] \).
3. Bounded Arity

Proof. We start out exactly as in the proof of Lemma 3.5 but transform our LHOM instance to an instance of MINHOM by assigning \( c_b(a) = 1 \) if \( b \in L_a \) and \( c_b(a) = 2 \) otherwise. This transformation does not change the Gaifman graph \( \mathcal{H}(A) \) of \( A \).

We end up with costs \( c_b(a), a \in A, b \in B' \) defined by:

\[
c_b(a) = \begin{cases} 
1 & \text{if } a \in A_1, b \in B \text{ and } b = (\_\_\_\_\_a) \\
1 & \text{if } a \notin A_1 \text{ and } a = b \\
2 & \text{otherwise}
\end{cases}
\]

Clearly, we will always be able to find a homomorphism from \( A \setminus A_1 \) to \( B' \setminus B \). Specifically it is always possible to find such a homomorphism of cost \(|A \setminus A_1|\), which is the minimum possible cost.

Due to the definition of \( c_b(a) \) above, it is clear that any homomorphism from \( A \to B \) has cost \( c \geq |A| \). Now, by Lemma 3.3 if and only if \( G \) contains a \( k \)-clique there exists a homomorphism \( h \) from \( A \to B \) such that each \( a \in A_1 \) maps to some \( b = (\_\_\_\_\_a) \in B \). It is easy to see that such an \( h \) has minimum cost \(|A|\).

Suppose that the cost, \( c \), of a minimum cost homomorphism from \( A \) to \( B \) is strictly greater than \(|A|\). Since we can always find a homomorphism from \( A \setminus A_1 \) to \( B' \setminus B \) of cost \(|A \setminus A_1|\), it must be the case that \( c - |A \setminus A_1| > |A_1| \). If so, there exists \( a \in A_1, b \in B \), such that \( a \) is mapped to \( b \) in the homomorphism, but \( b = (\_\_\_\_\_a') \), where \( a' \neq a \). We can thus conclude that there exists no homomorphism \( h \) from \( A \to B \) such that each \( a \in A_1 \) maps to some \( b = (\_\_\_\_\_a) \in B \) and, hence, by Lemma 3.3 \( G \) contains no \( k \)-clique. \( \square \)

In the proof of Lemma 3.9 we see that there is a small gap between the positive and negative instances of MINHOM. This gap can be amplified to yield an inapproximability result for structurally restricted MINHOM problems.

Proposition 3.10. Let \( \mathcal{A} \) be a recursively enumerable class of structures that does not have bounded tree-width. If MINHOM(\( \mathcal{A}_\_ \)) is approximable within \( 1/2^{p(|A|)} \), (where \( p \) is a fixed polynomial), for every structure \( A \in \mathcal{A} \), then FPT = W[1].

Proof. Assume that we have an algorithm that approximates MINHOM(\( \mathcal{C}_\_ \)) within \( 1/2^{p(|A|)} \), i.e., for every instance \( I \) of MINHOM(\( \mathcal{A}_\_ \)) it returns a value \( m(s) \) for a solution \( s \) obeying \( m(s) \leq c \cdot 2^{p(|A|)} \cdot OPT(I) \), for some constant \( c \geq 1 \). We will show that we can use this algorithm to decide \( p \)-CLIQUE in polynomial time.

To produce a large enough gap we redefine the costs \( c_b(a), a \in A, b \in B' \) used in the proof of Lemma 3.9 in the following way:

\[
c_b(a) = \begin{cases} 
1 & \text{if } a \in A_1, b \in B \text{ and } b = (v,e,i,p,a) \\
1 & \text{if } a \notin A_1 \text{ and } a = b \\
|A| \cdot c \cdot 2^{p(|A|)} & \text{otherwise}
\end{cases}
\]
3.3. Main Results

Now the positive instances of \( p\text{-CLIQUE} \) still have optimal measure \(|A|\) and the cost returned from our approximation algorithm, for a positive instance, is at most \(|A| \cdot c \cdot 2^p(|A|)\), while the negative instances have measure at least \(|A| - 1 + |A| \cdot c \cdot 2^p(|A|)\). So, in case our approximation algorithm for \( \text{MINHOM}(A,_) \) returns a value not greater than \(|A| \cdot c \cdot 2^p(|A|)\) we answer "yes" and "no" otherwise.

It is important to note that the number \(|A| \cdot c \cdot 2^p(|A|)\) is polynomially bounded in terms of \(|A|\) and since we have to write it down \(|A| \cdot |B'|\) times when defining the cost function, our instance of \( \text{MINHOM} \) is still polynomial in size of \( A \) and the input graph \( G \). (Because the size of \( B' \) is polynomially bounded in terms of \( ||A|| \) and \( |G| \)). Hence, we have a polynomial time algorithm for deciding \( p\text{-CLIQUE} \) and \( \text{FPT}=W[1] \).

\[ \square \]

VCSP

It is well-known that \( \text{VCSP}(A,_) \) is tractable if \( A \) is a class of structures of bounded tree-width \( \text{GSV06; NJT08; SW04; TJ03} \).

**Lemma 3.11.** Let \( A \) be a recursively enumerable class of structures of bounded arity that does not have bounded tree-width. If \( p\text{-VCSP}(A,_) \) is fixed-parameter tractable, then \( \text{FPT}=W[1] \).

**Proof.** It should be clear that we can perfectly emulate the \( W[1] \)-hard instances of \( p\text{-MINHOM}(A,_) \) computed in Lemma 3.9 with the help of corresponding instances of \( p\text{-VCSP} \). In essence, we start out exactly as in the proof of Lemma 3.9 but transform our \( \text{MINHOM} \) instance to an instance of \( \text{VCSP} \) by assigning each allowed tuple of values a cost which is equal to the sum of individual costs indicated by the \( \text{MINHOM} \) instance. More concretely, if \((A, B)\) is the computed \( \text{MINHOM} \) instance, for each tuple \((a_1, \ldots, a_r)\), in each relation \( R^A \) we add a new constraint \( \rho^R(a_1, \ldots, a_r) \) to our instance of \( \text{VCSP} \) with the following costs:

\[
\rho^R(a_1, \ldots, a_r)(b_1, \ldots, b_r) = \begin{cases} 
\sum_{i=1}^{r} c_b(a_i) & \text{if } (b_1, \ldots, b_r) \in R^B \\
\infty & \text{otherwise}
\end{cases}
\]

Since the arity of \( A \) is bounded we can specify these cost functions in polynomial time and, taken altogether, we have an fpt-reduction from \( p\text{-CLIQUE} \) to \( p\text{-VCSP}(A,_) \).

\[ \square \]

We remark that also the inapproximability result for \( \text{MINHOM} \) in Proposition 3.10 carries over to \( \text{VCSP} \) by using the same reduction as in Lemma 3.11.

MAX SOL

The final problem left is \( \text{MAX SOL} \). As we have seen, both \( \text{MINHOM} \) and \( \text{VCSP} \) give us great freedom in assigning costs in different ways. The \( \text{MAX SOL} \) problem is a bit more restrictive in this regard, and it seems hard to find
a structure preserving fpt-reduction from either MinHOM or VCSP to Max SOL. We have not been able to come up with such a reduction directly from LHOM either. Instead we take a more direct approach. The basic idea is to exploit the fact that a Max SOL instance has to impose a total order on the elements in the right hand side structure $B$; by letting elements of the form $(\_, \_, \_, a)$, for some $a \in A$, have essentially the same values and inter-spacing these clusters with large gaps, we are able to separate the positive and negative instances of $p$-CLIQUE.

**Lemma 3.12.** Let $\mathcal{A}$ be a recursively enumerable class of replication free structures of bounded arity that does not have bounded tree-width. If $p$-Max SOL($\mathcal{A}_{\_\_\_}$) is fixed-parameter tractable, then $\text{FPT} = \text{W}[1]$.

**Proof.** We start out as in the proof of Lemma 3.5 and construct $B'$ as $B \cup A_2 \cup \cdots \cup A_m$. To proceed, we have to impose some total order on the elements in $B'$. Fix the natural order $< \text{on } N$. The intuition is to let elements in $B$ on the form $(\_, \_, \_, a)$, for some $a \in A_1$, have essentially the same values in $B'$. If these small intervals where the $(\_, \_, \_, a) \in B$ reside, for each $a$, are inter-spaced by large gaps and the weights assigned to $a \in A_1$ are chosen accordingly we might be able to separate the positive and negative instances of $p$-CLIQUE.

Let $\sigma = \max_{a \in A_1} |\Pi^{-1}(a)|$, the maximum number of elements in $B$ “generated” by an element in $A_1$. Clearly, $\sigma$ is bounded in terms of $k$ and the size of $G$.

Let $B' \setminus B = \{1, \ldots, d\}$. Also, let $w(a) = 0$ when $a \in A \setminus A_1$. Furthermore, take an $a \in A_1$; let $w(a) = d + 1$ and let each $b \in \Pi^{-1}(a)$ have a distinct value in $[d + 1, d + \sigma]$. The next $a \in A_1$ gets $w(a) = d + \Delta + 1$ while the associated $b \in \Pi^{-1}(a)$ get distinct values in $[d + \Delta + 1, d + \Delta + \sigma]$. We continue this process until $A_1$ is exhausted and end up with the arrangement in Figure 3.3.

We are interested in homomorphisms $h$ between $A_1$ and $B$, such that each $a \in A_1$ maps to some $(\_, \_, \_, a) \in B$, i.e. where the $a \in A_1$ with highest weight get mapped to some $(\_, \_, \_, a) \in B$ in the highest interval of values, the $a \in A_1$ with second highest weight get mapped to some $(\_, \_, \_, a) \in B$ in the second highest interval of values and so on. Such an $h$ will receive a measure $m_{id}$ with

$$(d + 1)^2 + (d + \Delta + 1)^2 + \ldots + (d + (|A_1| - 1)\Delta + 1)^2 \leq m_{id} \leq (d + 1)(d + \sigma) + (d + \Delta + 1)(d + \Delta + \sigma) + \ldots + (d + (|A_1| - 1)\Delta + 1)(d + (|A_1| - 1)\Delta + \sigma).$$

It is easy to extend $h$ to a homomorphism $h'$ from $A$ to $B'$ (by mapping each $a \in A \setminus A_1$ to the $b \in B' \setminus B$ with $b = a$) and the measure for $h'$ will still be $m_{id}$.

What false positives could we get? Recall that for each relation symbol $R \in \tau$ and for all tuples $(a_1, \ldots, a_r) \in R^{A_1}$, we add tuples $(b_1, \ldots, b_r) \in \Pi^{-1}(a_1, \ldots, a_r)$ satisfying certain conditions to $R^B$ and that, in this case, $A_1$ is
replication free. This means that $B$ is constructed so that any homomorphism $h$ from $A_1$ to $B$ must have the property that the image of $h$ contains at most one element from each interval $[d + n\Delta + 1, d + n\Delta + \sigma]$, where $0 \leq n < |A_1|$, of values in $B$.

That leaves the possibility that some intervals of values have been permuted in some way, i.e. at least a pair of elements in $A_1$ have been mapped to somewhere in “each others” intervals. If $|A_1| = 2$ the maximum possible measure for such a mapping is $(d + 1)(d + \Delta + \sigma) + (d + \Delta + 1)(d + \sigma)$. When $|A_1| = 3$, the highest measure for a false positive occurs when we have $h(a_{11}) = (\_, \_, \_, a_{12})$, $h(a_{12}) = (\_, \_, \_, a_{11})$, and $h(a_{13}) = (\_, \_, \_, a_{13})$ and the maximum value of each interval is picked as image. We see that, in the general case, we can not gain a higher false positive measure by permuting more than the intervals of the bottom two elements of $|A_1|$. Hence the maximum measure of such a homomorphism occurs when the two elements in $A_1$ that have lowest weight have swapped intervals, i.e. we have $h(a_{11}) = (\_, \_, \_, a_{12})$ and $h(a_{12}) = (\_, \_, \_, a_{11})$ in Figure 3.3 and the maximum value of each interval is picked as image. This measure matches the maximum possible $m_{id}$ except for the two first summands.

The difference, denoted $\delta$, between the lowest possible $m_{id}$ and the measure of such a homomorphism is

$$\delta = \sum_{n=1}^{|A_1|} (d + (n - 1)\Delta + 1)^2 - (d + 1)(d + \Delta + \sigma)$$
3. **Bounded Arity**

\[-(d + \Delta + 1)(d + \sigma) - \sum_{n=3}^{[A_1]} (d + (n - 1)\Delta + 1)(d + (n - 1)\Delta + \sigma),\]

which is the same as (omitting the calculations) \(\delta\) being equal to

\[\Delta^2 + \left(|A_1| - |A_1| \cdot \sigma|A_1|^2 + \sigma|A_1|\right) \Delta/2 +
\]

\[+ |A_1| + d|A_1| - d\sigma|A_1| - \sigma|A_1|.\]

If we choose \(\Delta\) large enough, for example \(\Delta = d^2\sigma^2|A_1|\), the difference \(\delta\) will be positive and hence, we can say that if we find a homomorphism with measure \(m_{id}\), \(G\) has a \(k\)-clique and that if the maximum measure of any homomorphism from \(A\) to \(B'\) is strictly less than the smallest possible \(m_{id}\), \(G\) contains no \(k\)-clique.

### 3.4 Conclusions and Open Questions

We have utilised the structure \(B\) defined by Grohe to classify a number of homomorphism problems by computational complexity with regard to structural restrictions, under the assumption that \(\text{FPT} \neq \text{W}[1]\). It is interesting to note that while the variants of the homomorphism problem we have treated have their boundary between tractability and intractability at bounded tree-width of the left hand side input structure, the original \(\text{HOM}(A, _\cdot)\) problem exhibits the same boundary at bounded tree-width for the core of the structures in \(A\). It would be interesting to characterise exactly what properties make the computational complexity of our problems different from that of the “regular” homomorphism problem.

Of course it would be nice to classify further homomorphism problems. For example, the one-or-all list homomorphism problem would be an interesting subject. In this problem, inputs of the list homomorphism problem are restricted to each list containing only a single element or the entire universe of the right hand side input structure [FH98]. Here we let \(\text{OALHOM}(A, B)\) be the following problem:

**INSTANCE:** \(A \in A, B \in B, L_a \subseteq B\), such that \(|L_a| = 1\) or \(|L_a| = |B|\), for each \(a \in A\).

**OUTPUT:** “yes” if a homomorphism \(h\) from \(A\) to \(B\) such that \(h(a) \in L_a\) for each \(a \in A\) exists, “no” otherwise.

We remark that, for every graph \(H\), the problem \(\text{OALHOM}(\_\cdot \{H\})\) is linear time equivalent to the retraction problem [FH98]. Retractions and the retraction problem have been intensively studied in graph theory for a long time, cf. [HN04b]. Obviously, \(\text{OALHOM}(A, _\cdot)\) is at least as hard as \(\text{HOM}(A, _\cdot)\), and we know that \(\text{OALHOM}(A, _\cdot)\) is tractable if \(A\) has bounded tree-width by standard techniques. Furthermore, there exists classes of structures \(A\) with unbounded tree-width such that \(\text{OALHOM}(A, _\cdot)\) is solvable in polynomial time.

58
Example 3.13. Recall that $K_{k,k}$ is the complete bipartite graph with $k$ nodes in each set of the bipartition. It is easy to see that any graph $G$ with treewidth at most $k$ must have a node with degree at most $k$. Hence $K_{k,k}$ cannot have treewidth $k-1$ and hence the set $A$ containing $K_{k,k}$ for every $k$ has unbounded treewidth. However, there is an easy, consistency-style, algorithm for deciding whether an instance $(K_{k,k}, G)$ with lists $L$ is satisfiable in polynomial time.

This good news, however, comes paired with bad news of equal importance. Let $SCSP(A, \_)$ be the variant of $CSP(A, \_)$ in which, besides, deciding the existence of a solution, the algorithm is required to output a solution (if it exists). It is not difficult to see, by a self-reducibility argument, that if $OALHOM(A, \_)$ is solvable in polynomial time then so is $SCSP(A, \_).$ But we know of classes $A$ with bounded tree-width modulo homomorphic equivalence such that $SCSP(A, \_)$ is not solvable in polynomial time (see Lemma 1 of [BDGM12]). Taken together, these facts seem to imply that the computational complexity of $OALHOM(A, \_)$, for bounded arity structures, does not have a classification in simple terms of tree-width, but will require something more complicated.

Hell and Nešetřil [HN90] proved that for every class of undirected simple graphs $B$, $HOM(\_, B)$ is in polynomial time if all graphs in $B$ are bipartite. Otherwise $HOM(\_, B)$ is $NP$-complete. This type of result is known as a dichotomy theorem in the sense that the problems $HOM(\_, B)$ are either in polynomial time or $NP$-complete. Notice that Ladner [Lad75] has shown that if $P \neq NP$ then there are infinitely many pairwise distinct complexity classes between $P$ and $NP$. By following the lines of Ladner’s argument, Chen, Thurley, and Weyer [CTW08] show that the complexities of any parameterised problem in $XP$, not solvable in polynomial time, can be restricted to contain a dense linear order between $P$ and $NP$. This means that none of the structurally restricted homomorphism problems in this chapter exhibits a dichotomy in the classical sense.

We also want to mention that investigating our problems under simultaneous, or so called hybrid restrictions, where both the left and right hand side structures are simultaneously restricted, certainly could be interesting. E.g., if we let $B(2)$ denote the class of all relational structures over a two-element domain, it would seem that suitable restrictions on $A$ could give new larger classes $A$ of tractable $HOM(A, B(2))$ problems.

Recently, Takhanov [Tak10] obtained a complete classification of the computational complexity of $MINHOM(\_, B)$ for any restriction on (the constraint language) $B$. There has also been some initial progress on classifying $MINHOM$ for hybrid restrictions [CZ11]. Together with our results, this makes $MINHOM$ one of the most well understood problems in the CSP framework. In our reduction from $p$-CLIQUE to $p$-$MINHOM(A, \_)$ a gap that can be used to show inapproximability properties of the intractable instances is produced. A gap is also produced in the MAX SOL case, but it is not ex-
ployable in the same way. Is it possible to change the reduction somewhat to achieve a gap large enough for proving inapproximability?

A further observation is that the structure \(B\), so far, only has been applied when classifying homomorphism problems: Is it possible to modify the structure \(B\), or the analysis of it, so that hardness proofs for problems where the solution is not necessarily a homomorphism, e.g. \(\text{Max CSP}\), becomes plausible? It is well-known that \(\text{Max CSP}(\mathcal{A}, \_\_\) is tractable if \(\mathcal{A}\) is a class of structures of bounded tree-width [GSV06; NJT08; SW04; TJ03]. Now assume that \(\mathcal{A}\) is a class of structures such that \(p\text{-Max SOL}(\mathcal{A}, \_\) is \(\text{W[1]}\)-hard. Then we can show that weighted \(p\text{-Max CSP}(\mathcal{A}', \_\) also is \(\text{W[1]}\)-hard, where \(\mathcal{A}'\) consists of the structures in \(\mathcal{A}\) extended with “free use” of unary relations. By “free use” we mean that we can add a set of unary relations to each variable. More specifically, for an instance \(I = (V, D, C, w)\) of \(p\text{-Max SOL}\) we construct the following instance \(I^*\) of \(p\text{-Max CSP}\). We keep each constraint \(((x_1, \ldots, x_r), R^I)\) from \(I\) in \(I^*\) and set its weight to some large constant \(M\). Additionally, for each variable \(x_i \in \{x_1, \ldots, x_r\}\) we add a unary constraint for each domain value \(d \in D\) so that the only allowed tuple is \((d)\), with weight \(w(x_i) \cdot d\). If \(M\) is chosen large enough, a maximum weight assignment for \(I^*\) will be a satisfying assignment for \(I\) if one exists and it is easy to discern the case when \(I\) is unsatisfiable. Furthermore, such an assignment will be a maximum solution of \(I\). It is easy to see that this is an ftp-reduction from \(p\text{-Max SOL}(\mathcal{A}, \_\) to \(p\text{-Max CSP}(\mathcal{A}', \_\).

The structural restrictions studied in this chapter are for relational structures of bounded arity. When there is no bound on the arity, i.e. when the constraint scopes are large, the tree-width of the Gaifman graph may become unbounded rendering the tree decomposition technique useless. The correct way to handle this situation is to instead look at the constraint hypergraph. Gottlob, Greco, and Scarcello [GGS09] have identified tractable cases of \(\text{MINHOM}\) and \(\text{VCSP}\) based on (generalised) hypertree decompositions. Trying to enlarge this class of tractable instances by looking at other hypergraph decomposition measures is a research direction we explore in Chapter 4.
4 Unbounded Arity

In this chapter, we study the computational complexity of structurally restricted #CSP, MINHOM, and VCSP problems, when there is no bound on the arity of the relational structures involved.

4.1 Introduction

As we mentioned in the introduction to Chapter 3, Flum and Grohe [FG02] and Dalmau and Jonsson [DJ04] have studied structurally restricted #HOM problems, i.e., the question of how to restrict $A$, so that $\text{#HOM}(A, \_)$ is polynomial-time solvable. They prove the following:

Assume that $\text{FPT} \neq \text{#W[1]}$. Then for every recursively enumerable class $A$ of structures of bounded arity, $\text{#HOM}(A, \_)$ is polynomial-time solvable if and only if every structure in $A$ has treewidth at most $w$ (for some fixed $w$).

$\text{#W[1]}$ denotes the counting version of $\text{W[1]}$, and the classes $\text{FPT}$ and $\text{#W[1]}$ are conjectured to be non-equal [FG02] (in fact, if $\text{FPT} = \text{#W[1]}$, then $\text{FPT} = \text{W[1]}$).

The situation is a lot less clear in the unbounded arity case, and to our knowledge, no systematic study of this type of restriction for the counting homomorphism problem exists until now. First, the complexity in the unbounded-arity case depends on how the constraints are represented. In the bounded-arity case every reasonable representation of a constraint containing at most $r$ variables over the domain $D$ has size $|D|^{O(r)}$, which means that the size of the different representations only can differ by a polynomial factor. On the other hand, if there is no bound on the arity, then there can
be exponential difference between the size of succinct representations (e.g., formulas \[CG10\]) and verbose representations (e.g., truth tables \[Mar11\]).

The most well-studied representation of constraints is listing all the tuples that satisfy the constraint, and unless otherwise stated, our results will be for this representation. This seems the most reasonable representation, since we do not have any information about the instances, and it is the standard generic representation in artificial intelligence (see, for example \[Dec03\]). An important application where the constraints are always given in explicit form is the conjunctive query containment problem, which has a crucial role in database query optimisation. Kolaitis and Vardi \[KV00\] observed that it can be represented as a constraint satisfaction problem, where the constraint relations are given explicitly as part of one of the input queries. The problem of evaluating conjunctive queries is a related problem from database systems. Here the constraint relations represent the tables of a relational database given in explicit form.

It is known that the classification theorem above does not generalize to classes of structures of unbounded arity (we will demonstrate a simple counterexample in Section 4.2.1). Dalmau and Jonsson \[DJ04\] conjecture that some of the known tractable cases for unbounded arity for the decision version of the problem can be translated to the counting setting – a conjecture which we are able to verify in this chapter. As a byproduct of a result for plain CSPs, we know that a large family of classes of structures of arbitrary arity for which \(#\text{Hom}\) is in polynomial time consists of all classes of bounded \emph{fractional edge cover number} \[GM06\]. This is an entropy-based measure that, in a precise sense, captures the situation that the hypergraph associated with the instance is \emph{tightly constrained}. Homomorphism problems may also become tractable because their associated hypergraphs are \emph{loosely constrained}. Formally this is captured by the notion of bounded \emph{hypertree width} \[GLS02\]. Hypertree width is a hypergraph invariant that generalizes acyclicity \[Fag83\] and is incomparable to the fractional edge cover number. It is a very robust invariant that, up to a constant factor, coincides with a number of other natural invariants that measure the global connectivity of a hypergraph \[AGG07\]. In this chapter we show that \(#\text{Hom}(\mathcal{A}, _{-})\) is polynomial-time solvable if \(\mathcal{A}\) has hypertree width at most \(w\) (for some fixed \(w\)).

Grohe and Marx \[GM06\] proposed a new hypergraph invariant, the \emph{fractional hypertree width}, which generalizes both the hypertree width and fractional edge cover number. Their investigations, together with an algorithm for approximating this new invariant \[Mar10a\], implies that constraint satisfaction on structures of bounded fractional hypertree width is polynomial-time solvable. Here, we are able to transfer this result to the setting of counting problems.

What about our optimisation problems, \(\text{MinHom}\) and \(\text{VCSP}\), when it comes to unbounded arity? Here, the situation is a bit better: We know that optimal solutions to \(\text{MinHom}\) and \(\text{VCSP}\) problems can be found in polynomial time for classes of structures with bounded fractional edge cover number \[GM06\], and furthermore, Gottlob et al. \[GCS09\] have shown that in-
4.2 #CSPs and Bounded Width Measures

Instances of MINHOM are solvable to optimality if their hypertree width is at most \( w \) (for some fixed \( w \)). By using the techniques concerning fractional hypertree decompositions mentioned above as a base, we construct algorithms making bounded fractional hypertree width the most general known hypergraph property allowing polynomial-time solvable MINHOMs and VCSPs.

We close this introduction by making the observation that the algorithms we give in this chapter could easily be extended to also take care of instances of the list homomorphism problems from the previous chapter, LHOM, CLHOM, and ELHOM, as well as the MAXSOL problem. The rest of this chapter is organized as follows. Section 4.2 contains proofs of our tractability results for counting problems, while the corresponding results for optimisation problems can be found in Section 4.3. Finally, Section 4.4 concludes the chapter and presents possible future work.

4.2 #CSPs and Bounded Width Measures

Let \( I = (V, D, C) \) be an instance of #CSP. We denote by \( \text{sol}_I(V') \), for \( V' \subseteq V \), the number of satisfying assignments to \( I[V'] \), the #CSP instance induced by \( V' \). Our tractability results will crucially depend on the following lemma.

**Lemma 4.1.** There is an algorithm that, given an instance \( I \) of #CSP, an integer \( b_{\text{max}} \), a tree decomposition \( (T, (B_t)_{t \in V(T)}) \) of \( H(I) \) such that \( \text{sol}_I(B_t) \leq b_{\text{max}} \), and a procedure to enumerate the solutions to \( I[B_t] \) in time polynomial in \(|I|\) and \( b_{\text{max}} \), for every bag \( B_t \), counts the number of satisfying assignments of \( I \) in time polynomial in \(|I|\) and \( b_{\text{max}} \).

**Proof.** We will transform the instance to a solution equivalent binary #CSP instance, \( I' \), which we then use dynamic programming techniques on to be able to count the number of solutions in polynomial time. The variables of \( I' \) are the nodes of the hypertree decomposition, i.e., \( V(T) \). For \( t \in V(T) \), let \( b_t \) be the number of solutions \( \sigma \) of \( I[B_t] \), and denote by \( \sigma_{t,i} \) the \( i \)-th such solution to \( I[B_t] \) \( (0 \leq i \leq b_t) \). By the requirements of the lemma, we have \( b_t \leq b_{\text{max}} \). (Should \( b_t \) be zero for any \( t \in V(T) \), we know that the original instance \( I \) does not have any solutions.)

The domain of \( I' \) is \( D' = \{1, \ldots, b_{\text{max}}\} \). For each edge \( t_1t_2 \in E(T) \), we add a constraint \( C_{t_1,t_2} = ((t_1, t_2), R_{t_1,t_2}) \) to \( I' \), where \((i, j) \in R_{t_1,t_2} \) if and only if

- \( i \leq b_{t_1} \) and \( j \leq b_{t_2} \), and
- \( \sigma_{t_1,i} \) and \( \sigma_{t_2,j} \) are compatible, i.e., \( \sigma_{t_1,i}(v) = \sigma_{t_2,j}(v) \) for every \( v \in B_{t_1} \cap B_{t_2} \).

The size of \( I' \) is polynomial in \( b_{\text{max}} \) and \(|I|\), and its primal graph is tree shaped. It is easy to see that a solution of \( I' \) corresponds exactly to a solution of \( I \) and vice versa. This follows since there are no conflicts between the partial assignments of different bags (due to the connectedness property
of tree decompositions) and every original constraint is satisfied by one of the partial solutions (due to the property of tree decompositions that every hyperedge is contained in some bag of the decomposition).

Since the primal graph of $\mathcal{I}'$ is a tree, we can now use a second step of dynamic programming to count the number of solutions. Starting from the leaves, for every vertex $t \in V(T)$ of the tree we compute a table that stores, for all domain values $d \in D'$, the number $\sigma^d(t, d)$ of satisfying assignments $\sigma$ to $\mathcal{I}'[T_t]$, the instance induced by the subtree rooted at $t$, with $\sigma(t) = d$. Then the total number of solutions of $\mathcal{I}'$ is $\sum_{d \in D'} \sigma^d(r, d)$, where $r$ is the vertex corresponding to the root of the hypertree decomposition of $\mathcal{I}$.

If $t$ is a leaf, then $\sigma^d(t, d) = 1$ if $d \leq b_t$ and $\sigma^d(t, d) = 0$ otherwise. If $t$ has children $t_1, \ldots, t_l$, then if $d \leq b_t$ we have

$$\sigma^d(t, d) = \prod_{i=1}^{l} \sum_{d' \in D'} \sigma^d(t_i, d').$$

If $d > b_t$, we have $\sigma^d(t, d) = 0$.

By the size arguments given above, the counting step spends $O(||\mathcal{I}|| \cdot b_{\text{max}})$ time at each node of the tree, hence making the total running time polynomial in $||\mathcal{I}||$ and $b_{\text{max}}$ as required. \hfill \Box

### 4.2.1 Bounded Hypertree Width

We continue by showing a simple example of a class $\mathcal{A}$ of structures of unbounded treewidth such that $\#\text{CSP}(\mathcal{A}, \_)$ is tractable.

**Example 4.2.** For $n \geq 1$, let $R_n$ be an $n$-ary relation symbol, and let $A_n$ be the $\{R_n\}$-structure with universe $a_1, \ldots, a_n$ and $R_n^A = \{(a_1, \ldots, a_n)\}$. Let $\mathcal{A} = \{A_n \mid n \geq 1\}$. It is easy to see that the structure $A_n$ has tree-width $n - 1$. Thus $\mathcal{A}$ has unbounded tree-width. But $\#\text{CSP}(\mathcal{A}, \_)$ is tractable. To see this, let $\mathcal{I}$ be an instance of $\#\text{CSP}(\mathcal{A}, \_)$, with $A(\mathcal{I}) = A_n$. Then $\mathcal{I}$ has a single constraint $(\{a_1, \ldots, a_n\}, R_n)$. Thus $\mathcal{I}$ is satisfiable if and only if $R_n$ is nonempty, and clearly we can count the number of satisfying assignments in polynomial time. Note that this also entails $\text{MinHom}(\mathcal{A}, \_)$ and $\text{VCSP}(\mathcal{A}, \_)$ to be solvable to optimality in polynomial time.

Let $H = (V(H), E(H))$ be a hypergraph. An edge cover of $H$ is a set $C \subseteq E(H)$ of edges such that $V(H) = \bigcup C$. Here $\bigcup C = \bigcup_{e \in C} e = \{v \in V(H) \mid \exists e \in C : v \in e\}$. The edge cover number of $H$, denoted by $\rho(H)$, is the minimum cardinality of an edge cover of $H$. The edge cover number of a relational structure is defined to be the edge cover number of its hypergraph. Note that the structure $A_n$ of Example 4.2 has edge cover number 1 and tree-width $n - 1$.

**Example 4.3.** Let $\mathcal{A}$ be a class of relational structures of bounded edge cover number. Then $\#\text{CSP}(\mathcal{A}, \_)$ is tractable. This follows since if the hypergraph
of an instance $I$ has edge cover number $w$, then there are at most $|I|^w$ satisfying assignments. Clearly, also this result carries over to $\text{MINHOM}(\mathcal{A},_\cdot)$ and $\text{VCSP}(\mathcal{A},_\cdot)$.

We can now combine the observation in the previous example with the ideas used for structures of bounded tree-width. Let $H = (V(H), E(H))$ be a hypergraph. A generalised hypertree decomposition \cite{GLS02} of $H$ is a triple $(T, (B_t)_{t \in V(T)}, (C_t)_{t \in V(T)})$, where $(T, (B_t)_{t \in V(T)})$ is a tree decomposition of $H$ and $(C_t)_{t \in V(T)}$ is a family of subsets of $E(H)$ such that for every $t \in V(T)$ we have $B_t \subseteq \bigcup C_t$. The sets $C_t$ are called the guards of the decomposition and the width of $(T, (B_t)_{t \in V(T)}, (C_t)_{t \in V(T)})$ is $\max\{|C_t| \mid t \in V(T)\}$. The generalised hypertree width $\text{ghw}(H)$ of $H$ is the minimum of the widths of the generalised hypertree decompositions of $H$.

For completeness, we should mention that the “regular” hypertree decompositions incorporate an additional technical condition on how the guards must be arranged in the tree. However, it has been proven \cite{AGG07} that the two width measures are the same up to a constant factor, which makes them equivalent for our purposes. For simplicity, we will only work with generalised hypertree width.

Note that $\text{ghw}(H) \leq \text{tw}(H) + 1$ holds for every hypergraph $H$ \cite{Adl04}, and that, if $H$ is a hypergraph with $V(H) \in E(H)$, we have $\text{ghw}(H) = 1$, while $\text{tw}(H) = |V(H)| - 1$. Figure 4.1 gives an example hypergraph and a generalised hypertree decomposition of it.

Gottlob et al. \cite{GLS02} proved that $\text{CSP}(\mathcal{A},_\cdot)$ is tractable for all classes $\mathcal{A}$ of bounded generalised hypertree width. Here, we show that tractability also holds for the corresponding $\#\text{CSP}(\mathcal{A},_\cdot)$ problems.
Theorem 4.4. Let $\mathcal{A}$ be a class of relational structures of bounded generalised hypertree width. Then $\#\text{CSP}(\mathcal{A},\_\_)$ is solvable in polynomial time.

Proof. Let $\mathcal{I}$ be an instance of $\#\text{CSP}(\mathcal{A},\_\_)$ and $(T, (B_t)_{t \in V(T)}, (C_t)_{t \in V(T)})$ a generalised hypertree decomposition of $H(\mathcal{I})$ of width $r$. Unfortunately, it has been shown that deciding whether $\text{ghw}(H) \leq k$ is $\text{NP}$-complete (for any fixed $k \geq 3$) [GMS09]. However, the regular hypertree decomposition is polynomial time computable [GLS02], so we settle for such a decomposition, which makes $r$ become $3 \cdot \text{ghw}(H(\mathcal{I})) + 1$ [AGG07].

The key observation that we will use is that every bag in the hypertree decomposition has edge cover number $r$, meaning each bag can be covered by using at most $r$ hyperedges (or $r$ constraint scopes, if we so wish). Since we also know which hyperedges constitute these covers we can, in accordance with Example 4.3, conclude that there are at most $||\mathcal{I}||^r$ satisfying assignments to $\mathcal{I}[B_t]$, and that it is easy to enumerate these solutions in time $||\mathcal{I}||^{O(r)}$. To finish the proof, all we have to do is apply the algorithm of Lemma 4.1. □

Example 4.5. Consider the following $\#\text{CSP}$ instance $\mathcal{I}$: The set of variable is given by $V = \{x_1, x_2, x_3, x_5\}$, the domain is $D = \{0, 1\}$, and the constraints are $C_1 = \{(x_1, x_3, x_5), \{(0,0,0), (0,0,1), (0,1,0), (1,0,0)\}\}$, $C_2 = \{(x_1, x_3), \{(0,0), (0,1), (1,0)\}\}$, $C_3 = \{(x_1, x_2, x_4), \{(0,0,0), (0,0,1), (0,1,0), (1,0,0)\}\}$, $C_4 = \{(x_2, x_4), \{(0,0), (0,1), (1,0)\}\}$, $C_5 = \{(x_3, x_4), \{(0,0), (0,1), (1,0)\}\}$. In Figure 4.2, we can see the constraint hypergraph of this instance alongside a hypertree decomposition of the hypergraph. Right away, we see that this instance of $\#\text{CSP}$ admits the all zero solution, but the real question is how many different solution there really is. We will now use Theorem 4.4 to answer this question.

To start, we have to compute the solutions of $\mathcal{I}$ projected to the variables in each bag of the hypertree decomposition. Luckily, the solutions to
4.2. \#CSPs and Bounded Width Measures

\[ \sigma^#(t_1, 1) = 1 \]
\[ \sigma^#(t_1, 2) = 1 \]
\[ \sigma^#(t_1, 3) = 1 \]
\[ \sigma^#(t_1, 4) = 0 \]
\[ \sigma^#(t_1, 5) = 0 \]
\[ \sigma^#(t_1, 6) = 0 \]
\[ \sigma^#(t_1, 7) = 0 \]

\[ \sigma^#(t_2, 1) = \sigma^#(t_1, 1) = 1 \]
\[ \sigma^#(t_2, 2) = \sigma^#(t_1, 2) = 1 \]
\[ \sigma^#(t_2, 3) = \sigma^#(t_1, 3) = 1 \]
\[ \sigma^#(t_2, 4) = \sigma^#(t_1, 4) = 0 \]
\[ \sigma^#(t_2, 5) = \sigma^#(t_1, 5) = 0 \]
\[ \sigma^#(t_2, 6) = \sigma^#(t_1, 6) = 0 \]
\[ \sigma^#(t_2, 7) = \sigma^#(t_1, 7) = 1 \]

\[ \sigma^#(t_r, 1) = (\sigma^#(t_2, 1) + \sigma^#(t_2, 2) + \sigma^#(t_2, 4)) \cdot (\sigma^#(t_3, 1)) = 3 \]
\[ \sigma^#(t_r, 2) = (\sigma^#(t_2, 1) + \sigma^#(t_2, 2) + \sigma^#(t_2, 4)) \cdot (\sigma^#(t_3, 1)) = 3 \]
\[ \sigma^#(t_r, 3) = (\sigma^#(t_2, 3) + \sigma^#(t_2, 5)) \cdot (\sigma^#(t_3, 2)) = 2 \]
\[ \sigma^#(t_r, 4) = (\sigma^#(t_2, 6)) \cdot (\sigma^#(t_3, 3)) = 1 \]
\[ \sigma^#(t_r, 5) = 0 \]
\[ \sigma^#(t_r, 6) = 0 \]
\[ \sigma^#(t_r, 7) = 0 \]

Figure 4.3: The dynamic programming tables computed by the algorithm behind Lemma 4.1

\[ \mathcal{I}[\{x_1, x_3, x_5\}], \mathcal{I}[\{x_1, x_3\}], \text{ and } \mathcal{I}[\{x_2, x_4\}], \] with respect to this particular hypertree decomposition, are precisely the allowed tuples in the relations of \( C_1, C_2, \) and \( C_4 \), respectively. The bag containing \( \{x_1, x_2, x_3, x_4\} \) is guarded by \( C_3 \) and \( C_5 \), which means that the solutions to \( \mathcal{I}[\{x_1, x_2, x_3, x_4\}] \) becomes \( \{(0, 0, 0, 0), (0, 0, 1, 0), (0, 1, 0, 0), (0, 1, 1, 0), (1, 0, 0, 0), (1, 0, 1, 0)\} \).

Lemma 4.1 now tells us to proceed by transforming the instance \( \mathcal{I} \) to an equivalent binary instance \( \mathcal{I}' \). Choosing \( t_r \) as the root of the decomposition, the domain of \( \mathcal{I}' \) becomes \( D' = \{1, 2, 3, 4, 5, 6, 7\} \), and we add the following constraints to \( \mathcal{I}' \):

\[ C_{t_2, t_1} = \{(t_2, t_1), \{(1, 1), (2, 2), (3, 1), (4, 3), (5, 3), (6, 1), (7, 1)\}\}, C_{t_2, t_2} = \{(t_2, t_2), \{(1, 1), (1, 2), (1, 3), (2, 2), (2, 4), (3, 3), (3, 5), (4, 6)\}\}, C_{t_2, t_3} = \{(t_2, t_3), \{(1, 1), (2, 1), (3, 2), (4, 3)\}\} \].

Figure 4.3 now gives the result of dynamic programming counting step for \( \mathcal{I}' \). We conclude that the number of solutions to \( \mathcal{I}' \) (and \( \mathcal{I} \)) is \( \sum_{i=1}^{7} \sigma^#(t_r, d) = 9 \).

4.2.2 Bounded Fractional Hypertree Width

By construction, we have that \( \text{gwh}(H) \) is less than or equal to the edge cover number of \( H \), for every hypergraph \( H \). The problem of finding a minimum edge cover of a hypergraph \( H = (V(H), E(H)) \) has the following integer linear programming (ILP) formulation:

\begin{align}
\text{Minimise} \quad & \sum_{e \in E(H)} x_e \\
\text{subject to} \quad & \sum_{e \in V(H)} x_e \geq 1 \quad \text{for all } v \in V(H), \\
& x_e \in \{0, 1\} \quad \text{for all } e \in E(H).
\end{align}

67
4. Unbounded Arity

If we relax the integrality constraints by replacing them with the inequalities $x_e \geq 0$, we get a linear program whose feasible solutions are called fractional edge covers of $H$. The weight $\sum_{e \in E} x_e$ of an optimal solution is called the fractional edge cover number of $H$, denoted by $\rho^*(H)$. It follows from standard linear programming results that an optimal fractional edge cover and the (rational) fractional edge cover number can be computed in polynomial time. Computing the (integral) edge cover number is NP-complete. Figure 4.4 shows a simple hypergraph $H$, with $\rho(H) = 2$ and $\rho^*(H) = 1.5$.

Example 4.6. For $\ell \geq 1$, let $H_\ell$ be the following hypergraph: $H_\ell$ has a vertex $v_S$ for every subset $S$ of $\{1, \ldots, 2\ell\}$ of cardinality $\ell$. Furthermore, $H_\ell$ has a hyperedge $e_i = \{v_S \mid i \in S\}$ for every $i \in \{1, \ldots, 2\ell\}$. Setting $x_{e_i} = 1/\ell$ for every hyperedge $e_i$ gives a fractional edge cover of weight 2, implying that $\rho^*(H_\ell)$ is at most 2. In comparison, the hypertree width of $H_\ell$ is $\ell$ [GM06].

It is not hard to see that the hypergraph consisting of the disjoint union of $n$ edges of cardinality 1 has hypertree width 1 and fractional edge cover number $n$. This fact, together with the observations in Example 4.6, show that (generalised) hypertree width and fractional edge cover number are incomparable. The preceding two examples show that (generalised) hypertree width and fractional edge cover number are incomparable. As mentioned in the introduction, Grohe and Marx [GM06] have studied fractional edge covers in the context of computational complexity of CSPs. By using a clever argument based on Shearer’s Lemma [CGFS86], which is a combinatorial consequence of the submodularity of the entropy function, they show that a CSP instance $I$ has at most $|I|^{\rho^*(H(I))}$ solutions. This result is then used to bound the running time of a simple algorithm to prove that the solutions of a CSP instance $I$ can be enumerated in time $|I|^{\rho^*(H(I)) + O(1)}$, without the use of an actual fractional edge cover.

**Lemma 4.7 (Shearer’s Entropy Lemma [CGFS86]).** Let $H$ be a hypergraph, and let $A_1, A_2, \ldots, A_p$ be (not necessarily distinct) subsets of $\mathcal{N}(H)$ such that each $v \in \mathcal{N}(H)$ is contained in at least $q$ of the $A_i$’s. Denote by $E_i$ the edge set of the induced hypergraph $H[A_i]$. Then $|E(H)| \leq \prod_{i=1}^p |E_i|^{1/q}$.
Note that we admit empty hyperedges. In particular, if \( e \cap A_i = \emptyset \) for some \( e \in \mathcal{E} \) and \( i \leq p \), then \( \emptyset \in \mathcal{E}_i \).

We remark that using Shearer’s Entropy Lemma for counting purposes is in no way unique; Friedgut and Kahn [FK98] used the lemma, and fractional edge covers, to bound the number of subhypergraphs of a certain isomorphism type in a hypergraph. Furthermore, Radhakrishnan [Rad03] give applications of Lemma 4.7 to several graph covering problems. Friedgut [Fri04] references an extension of the lemma to a weighted setting by Friedgut and Rödl [FR01].

**Lemma 4.8 (Generalised Weighted Entropy Lemma)** [Fri04]. Let \( H = (V,E) \) be a hypergraph, and let \( A_1, A_2, \ldots, A_p \) be (not necessarily distinct subsets) of \( V \). Let \( E_i \) denote the edge set of the induced hypergraph \( H[A_i] \), and let the edge \( e_i = e \cap A_i \) of \( E_i \) be endowed with a non-negative real weight \( w_i(e_i) \). If \( \alpha = (\alpha_1, \ldots, \alpha_p) \) is a vector of weights such that

\[
\sum_{v \in A_i} \alpha_i \geq 1
\]

for each \( v \in V \), then

\[
\sum_{e \in E} \prod_{i=1}^{p} w_i(e_i) \leq \prod_{i=1}^{p} \left( \sum_{e_i \in E_i} w_i(e_i)^{1/\alpha_i} \right)^{\alpha_i}.
\]

We will now use this extension to build an alternative, more direct, proof of Grohe and Marx’ key lemma.

**Lemma 4.9.** A CSP instance \( \mathcal{I} \) admits at most \( ||\mathcal{I}||^{\rho^*(H(\mathcal{I}))} \) different solutions.

**Proof.** Let \( \mathcal{I} = (V,D,C) \) be a CSP instance with \( V = \{v_1, \ldots, v_n\} \). We will now define the same auxiliary hypergraph \( \hat{H} \) over \( V \times D \) as Grohe and Marx does. The intention is that the edges of \( \hat{H} \) should correspond to the solutions of \( \mathcal{I} \). To this end, we let each solution \( \sigma : V \to D \) to the instance correspond to an edge \( \{(v_1,\sigma(v_1)), \ldots, (v_n,\sigma(v_n))\} \) in \( \hat{H} \). We will now use the Generalised Weighted Entropy Lemma to bound the number of edges on the auxiliary hypergraph. Let \( \psi \) be a fractional edge cover of \( H(\mathcal{I}) \) with \( \sum_{e \in E(H(\mathcal{I}))} \psi(e) = \rho^*(H(\mathcal{I})) \). Due to well-known linear programming results, we may assume that the \( \psi(e) \) only takes rational values.

Remember that each edge in \( H(\mathcal{I}) \) corresponds to a constraint \( c_i \in C \). Now define \( \hat{A}_i = c_i \times D \), and let the weight \( \alpha_{c_i} \) of \( \hat{A}_i \) be the same as the weight \( \psi(c_i) \). We let \( \hat{E}_i \) be the edge set of \( H[\hat{A}_i] \), i.e., \( \hat{E}_i = \{e \cap \hat{A}_i | e \in E(\hat{H})\} \), and set the weights \( w_i \) of \( e_i \in \hat{E}_i \) uniformly to one. By Lemma 4.8, we have

\[
\sum_{e \in E(\hat{H})} \prod_{i=1}^{\lvert E(H(\mathcal{I})) \rvert} w_i(e_i) \leq \prod_{i=1}^{\lvert E(H(\mathcal{I})) \rvert} \left( \sum_{e_i \in \hat{E}_i} w_i(e_i)^{1/\alpha_i} \right)^{\alpha_i}.
\]

69
Since the weights $w_i$ are all one, we can conclude that the number of edges of $\hat{H}$ can be bounded by

$$|E(H(I))| \prod_{i=1}^{\gamma_i} |E_i|^{|a_i|}.$$

In exactly the same way as in the proof of Grohe and Marx, the hypergraph $\hat{H}[A_i]$ describes how the solutions look like if we consider only the variables in $c_i$; each edge of $\hat{H}[A_i]$ describes a possible combination of values a solution can have on the variables in the constraint $c_i$. This means that, in the worst case, there can be at most $|I|$ edges in $E_i$. This leads us to the conclusion that the number of edges of $\hat{H}$, and hence the number of solutions to the CSP instance, can be bounded by

$$\prod_{i=1}^{\gamma_i} |I|^{|a_i|} \prod_{i=1}^{\gamma_i} \sum_{e \in E_i} \psi(e) = |I|^{|\rho(H(I))|}.$$

Following Grohe and Marx, we can now combine fractional edge covers and generalised hypertree decompositions as follows: A fractional hypertree decomposition \cite{GM06} of a hypergraph $H = (V(H), E(H))$ is a triple $(T, (B_t)_{t \in V(T)}, (\gamma_t)_{t \in V(T)})$, where $(T, (B_t)_{t \in V(T)})$ is a tree decomposition of $H$ and $(\gamma_t)_{t \in V(T)}$ is a family of mappings from $E(H)$ to $\mathbb{Q}_{\geq 0}$ such that for every $t \in V(T)$ it holds that

$$\sum_{e \in E(t)} \gamma_t(e) \geq 1 \quad \text{for all } v \in B_t.$$

Hence the (fractional) guard $\gamma_t$ is a fractional edge cover of the subhypergraph induced by the bag $B_t$. The width of $(T, (B_t)_{t \in V(T)}, (\gamma_t)_{t \in V(T)})$ is $\max\{\sum_{t \in V(T)} \gamma_t(e) \mid t \in V(T)\}$, and the fractional hypertree width fhw$(H)$ of $H$ is the minimum of the widths of the fractional hypertree decompositions of $H$. Since there are only finitely many tree decompositions (up to the obvious equivalence) of a hypergraph $H$, it follows that this minimum width always exists and is rational for each $H$.

By definition, fhw$(H) \leq \rho^*(H)$ and fhw$(H) \leq \operatorname{ghw}(H)$ holds for every hypergraph $H$, and the examples given above show that the gap between fractional hypertree width and both generalised hypertree width and fractional edge cover number can become unbounded. We also want to mention that for every hypergraph $H$, fhw$(H) = 1 \Leftrightarrow \operatorname{ghw}(H) = 1$ \cite{GM06}, and that ghw$(H) = 1$ if and only if $H$ is acyclic \cite{GLS02}.

Grohe and Marx \cite{GM06} proved that CSPs are polynomial-time solvable if the input contains a bounded-width fractional hypertree decomposition of the associated hypergraph. Recently, Marx \cite{Mar10a} showed that for every fixed $w \geq 1$, there is a polynomial-time algorithm that, given a hypergraph $H$ with fractional hypertree width at most $w$, computes a fractional hypertree decomposition of width $O(w^3)$ for $H$. Therefore, if $A$ is a
class of relational structures with bounded fractional hypertree width, then $\text{CSP}(\mathcal{A}, \_)$ is tractable. Now we prove that this tractability result holds also for $\#\text{CSP}(\mathcal{A}, \_)$.

**Theorem 4.10.** Let $\mathcal{A}$ be a class of relational structures of bounded fractional hypertree width. Then $\#\text{CSP}(\mathcal{A}, \_)$ is solvable in polynomial time.

**Proof.** Let $\mathcal{I}$ be an instance of $\#\text{CSP}(\mathcal{A}, \_)$ and $(T, (B_t)_{t \in V(T)}, (\gamma_t)_{t \in V(T)})$ a fractional hypertree decomposition of $H(\mathcal{I})$ of width $r$. Such a decomposition can be computed by Marx's algorithm mentioned above, which makes $r$ become $O((\text{fhw}(H(\mathcal{I})))^3)$. Define $V_t := \bigcup_{t \in V(T)} B_t$. The corresponding algorithm for CSPs (of Grohe and Marx) constructs, for each $t \in V(T)$, in a bottom-up manner, the list $L_t$ of solutions of $\mathcal{I}[B_t]$ that can be extended to a solution of $\mathcal{I}[V_t]$. Indeed, by inspecting $L_{t_0}$, for the root $t_0$ of the tree decomposition, we can decide whether $\mathcal{I}$ has a solution or not.

Here, we use Lemma 4.9, observing that, by the results of Grohe and Marx mentioned above, for every bag $B_t$, we have $\text{sol}_H(\mathcal{I}[B_t]) \leq ||\mathcal{I}||^r$, and that we can enumerate the solutions of the projection to each bag in time $||\mathcal{I}||^r \cdot O(1)$.

---

**4.3 MINHOM, VCSP, and Fractional Hypertree Width**

Gottlob et al. [GGS09] proved that $\text{MINHOM}(\mathcal{A}, \_)$ is solvable to optimality in polynomial time for all classes $\mathcal{A}$ of bounded generalised hypertree width. We will use this theorem repeatedly in Chapter 5.3 of the thesis. The proof of Theorem 4.11 is founded on Yannakakis' classical algorithm for evaluating acyclic Boolean conjunctive queries [Yan81].

**Theorem 4.11 ([GGS09]).** Let $\mathcal{A}$ be a class of relational structures of bounded generalised hypertree width. Then $\text{MINHOM}(\mathcal{A}, \_)$ is solvable to optimality in polynomial time.

We have shown that if $\mathcal{A}$ is a class of relational structures with bounded fractional hypertree width, then $\#\text{CSP}(\mathcal{A}, \_)$ is tractable. In what follows, we will demonstrate that this holds also for our optimisation problems. In the MINHOM case, this is done by exploiting Lemma 4.9 which lets us conclude that the projection of the instance to every bag has a polynomial number of solutions. This lets us transform the instance to an equivalent binary MINHOM instance, which we then process in a two-phase dynamic programming step to find an optimal solution. The VCSP case is dealt with by demonstrating a structure preserving reduction to the MINHOM case. This reduction is also used as a building block in the proofs of Propositions 5.2 and 5.3.

**Theorem 4.12.** Let $\mathcal{A}$ be a class of relational structures of bounded fractional hypertree width. Then $\text{MINHOM}(\mathcal{A}, \_)$ and $\text{VCSP}(\mathcal{A}, \_)$ are solvable to optimality in polynomial time.
4. Unbounded Arity

Proof. Let $I = (V, D, C, \{c_d\}_{d \in D})$ be an instance of MINHOM($\mathcal{H}$) and $(T, (B_t)_{t \in V(T)}, (\gamma_t)_{t \in V(T)})$ a fractional hypertree decomposition of $H(I)$ of width $r$. Such a decomposition can be computed by Marx’s algorithm mentioned above, which makes $r$ become $O((\text{fhw}(H(I)))^3)$. Define $V_t := \bigcup_{t \in V(T)} B_t$. The corresponding algorithm for CSPs (of Grohe and Marx) constructs, for each $t \in V(T)$, in a bottom-up manner, the list $L_t$ of solutions of $I[B_t]$ that can be extended to a solution of $I[V_t]$. Indeed, by inspecting $L_{t_0}$, for the root $t_0$ of the tree decomposition, we can decide whether $I$ has a solution or not.

Here, we choose to transform the instance to a solution equivalent binary MINHOM instance, $I'$, which we then use dynamic programming techniques on to be able to find an optimal solution in polynomial time. The variables of $I'$ are the bags of the fractional hypertree decomposition, i.e., $V(T)$. For $t \in V(T)$, let $b_t$ be the number of satisfying assignments $\sigma$ of $I[B_t]$, and denote by $\sigma_{i,t}$ the $i$-th such solution to $I[B_t]$ ($0 \leq i \leq b_t$). By the results of Grohe and Marx mentioned above, we have $b_t \leq ||I||^r$ and we can enumerate the $\sigma_{i,t}$ in time $||I||^{r+O(1)}$. Furthermore, we let $n_v$, for each variable $v$ of $I$, be the number of bags $v$ occurs in.

The domain of $I'$ is $D' = \{1, \ldots, b_{\text{max}}\}$, where $b_{\text{max}} = \max_{t \in V(T)} b_t$. (Should $b_t$ be zero for any $t \in V(T)$, we know that the original instance $I$ does not have any solutions.) For each edge $t_1t_2 \in E(T)$, we add a constraint $C_{t_1,t_2} = ((t_1, t_2), R_{t_1,t_2})$ to $I'$, where $(i,j) \in R_{t_1,t_2}$ if and only if

- $i \leq b_{t_1}$ and $j \leq b_{t_2}$, and
- $\sigma_{t_1,i}$ and $\sigma_{t_2,j}$ are compatible, i.e., $\sigma_{t_1,i}(v) = \sigma_{t_2,j}(v)$ for every $v \in B_{t_1} \cap B_{t_2}$.

The size of $I'$ is polynomial in $b_{\text{max}}$, i.e., its size is $||I||^{O(r)}$, and its primal graph is tree shaped. By the arguments above the procedure of generating the instance $I'$, given $I$ and its fractional hypertree decomposition, is polynomial time for each fixed $r$. It is also easy to see that a solution of $I'$ corresponds exactly to a solution of $I$ and vice versa. This follows since there are no conflicts between the partial assignments of different bags (due to the connectedness property of tree decompositions) and every original constraint is satisfied by one of the partial solutions (due to the property of tree decompositions that every hyperedge is contained in some bag of the decomposition). By setting the costs of $I'$ as $c'_t(t) = \sum_{\gamma_t(v) \in B_t} (c_{\gamma_t(v)}(v)/n_v)$, for $t \in V(T)$ and $i \leq b_t$, we have also preserved the costs of all solutions. (Any costs not defined by this procedure can remain undefined, since they can not contribute to any satisfying assignment of $I'$.)

Since the primal graph of $I'$ is a tree, we can now use a second step of dynamic programming to determine an optimal solution. Starting from the leaves, for every vertex $t \in V(T)$ of the tree we compute a table that stores, for all domain values $d \leq b_t$, the cost $c_{\text{opt}}(t,d)$ of an optimal solution $\sigma$ to $I'[V_t]$, the instance induced by the subtree rooted at $t$, with $\sigma(t) = d$. Then
the value of an optimal solution of $I'$ is $\min_{d \leq b_I} c^{opt}(t_0, d)$, where $t_0$ is the vertex corresponding to the root of the hypertree decomposition of $I$.

To start, we set $c^{opt}(t, d) := c'_I(t)$, for each $t \in V(T)$ and $d \leq b_I$. If $t$ has children $t_1, \ldots, t_l$, then for each $d \leq b_I$ we do, for each child $t_i$:

- $d'_i := \arg\min_{d \leq b_I} \{c^{opt}(t_i, d')\}$ (resolving ties arbitrarily),
- \( t(d, i) := d'_i \) (store the value $d'_i$ in the variable $t(d, i)$), and
- $c^{opt}(t, d) := c^{opt}(t, d) + c^{opt}(t_i, d'_i)$.

After this phase we know the value of an optimal solution $\sigma$, and to get hold of an assignment to the variables of $I'$ with this value all we have to do is process $T$ top-down, starting from the root. This means we start by setting $\sigma(t_0) := \bar{d}_{t_0}$, where $\bar{d}_{t_0} := \arg\min_{d \leq b_I} c^{opt}(t_0, d)$ (again, resolving ties arbitrarily). Then $\sigma$ is extended with a value for each child of the root of $T$: for each child $t_i$ of $t_0$, $\sigma$ is extended by setting $\sigma(t_i)$ to the value $t(\bar{d}_{t_0}, i)$ resulting from the bottom-up phase. This procedure is then carried out recursively for each vertex of $T$, so that $\sigma$ is extended to a solution over all variables of $I'$, from which we can recover an optimal solution to $I$.

By the size arguments given above, the bottom-up and top-down phases spends $||I||^O(r)$ time at each node of the tree, hence making the total running time polynomial for each fixed $r$. This concludes the proof for the \textsc{MinHom} case.

From an instance $I = (V, D, C, Q)$ of $\text{VCSP}(A_{\infty})$, we will demonstrate how to create an equivalent instance $I' = (V', D', C', \{c_d\}_{d \in D})$ of $\text{MinHom}(H)$. Begin by dropping all tuples of every constraint $\rho^i x_1, \ldots, x_r$ that are mapped to $\infty$. Now, set $V' = V \cup \{a_1, \ldots, a_{|C'|}\}$, where each $a_i$ is a fresh auxiliary variable in $I'$. Furthermore, $D'$ is $D$ extended with a fresh value for each (finite-valued) tuple that is left in the relation of each constraint of $I$. Intuitively, mapping the variable $a_i$ to a fresh value from the corresponding constraint encodes that exactly that tuple is going to contribute to an optimal solution for $I$. This is accomplished by altering the $i$th constraint $\rho^i x_1, \ldots, x_r, \in C$, so that its scope in $C'$ becomes $(x_1, \ldots, x_r, a_i)$, and the tuples of its relation are extended with the fresh values in $D'$ created for each tuple above. Finally, we set $c_d \equiv 0$ for $d \in D$ and the costs for the new domain values are defined to correspond to the cost, in $I$, of the tuple it belongs to if the variable mapped to it is the correct fresh variable, and zero otherwise. That is, the whole cost of each tuple is determined by where its associated fresh variable is mapped. All in all, $\text{fhw}(H(I')) = \text{fhw}(H(I))$, and we have neither added any tuples to the relations compared to $I$ nor more than doubled the length of the tuples, which means we have manufactured an instance of $\text{MinHom}(A_{\infty})$ with bounded fractional hypertree width in linear time, which establishes the theorem. \qed
4. **Unbounded Arity**

4.4 Conclusions and Outlook

We have investigated structural properties that can make counting the solutions to constraint satisfaction problems tractable and identified two new classes of polynomial-time solvable #CSPs, thereby confirming a conjecture of Dalmau and Jonsson [DJ04]. Our results make bounded fractional hypertree width the strictly most general known hypergraph property that allows #CSP to be solved in polynomial time. Since the result for fractional hypertree width subsumes the result for (generalised) hypertree width with respect to polynomial time solvability, the proof of the latter result might seem a bit redundant. However, since we currently only have an $O(w^3)$ approximation algorithm for fractional hypertree width, our algorithm for bounded (generalised) hypertree width should prove much more efficient for problems belonging to this class.

We have also investigated structural properties that can make solving constraint optimisation problems tractable and devised algorithms to identify a new large class of polynomial time solvable VCSPs and CSOPs. Also in this case, our results make bounded fractional hypertree width the strictly most general known hypergraph property that allows VCSP and CSOP problems to be solved optimally in polynomial time.

Figure 4.4 shows some of the known tractable hypergraph properties. (Note that the elements of this Venn diagram are sets of hypergraphs; for example, the set “bounded tree-width” contains every set $H$ of hypergraphs with bounded tree-width.) As discussed above, all the inclusions in the figure are proper.

As we have seen, the key property used in tractability results based on hypergraph properties is that if some width measure is at most $w$, then for every bag $B$, the projection of the instance to $B$ has at most $||I||^w$ solutions. A natural question at this point is if there is a measure smaller than fractional hypertree width that can be used to bound the number of solutions in the bags of a tree decomposition. It turns out that the answer to this question is no; by a result of Atserias et al. [AGM08], we have that if the fractional hypertree width of a decomposition is at least $w$, then there are (arbitrarily large) instances $I$ where the projection to some bag has $||I||^\Omega(w)$ solutions. Thus it would seem that there is no width measure better than fractional hypertree width. We can get around this “optimality” by using a new width measure called *submodular width*, invented by Marx [Mar10b], that is not larger than fractional hypertree width. The high-level ideas behind this approach is that the when picking the decomposition, the choice can depend not only on the hypergraph of the instance, but on the actual constraint relations, and that we can branch on adding further restrictions, and apply different tree decompositions to each resulting instance.

By making trivial modifications to Marx' ingenious argumentation for the plain CSP case it follows that assuming the exponential time hypothesis (that there is no $2^{o(n)}$ time algorithm for $n$-variable 3SAT) [IPZ01], if $A$ is a recursively enumerable class of relational structures, then $\#\text{CSP}(A,\_)$
4.4. Conclusions and Outlook

Bounded (generalised) hypertree width
Bounded fractional edge cover number
Bounded fractional hypertree width
Bounded tree width

Figure 4.5: Hypergraph properties that make #CSP, VCSP, and MINHOM polynomial time solvable.

is fixed-parameter tractable if and only if the hypergraphs of the structures in $A$ have bounded submodular width. An obvious question for further research is determining the precise complexity of those classes of hypergraphs that have bounded submodular width but unbounded fractional hypertree width. One possibility is that #CSP is solvable in polynomial time for every such class, but this seems improbable since the fpt result splits each instance into a double exponential number of instances that are uniform in a particularly convenient way. Exploring this new measure in the context of our optimisation problems could also be a good venue for further research.

Gottlob, Greco, and Scarcello [GGS09] have shown that MAX CSP($A$, _) is tractable if $A$ is a class of structures such that the tree-width of the incidence graph of each $A \in A$ is bounded. It can be shown that the tree-width of the incidence graph of a structure is at most the tree-width of the Gaifman graph of the structure and that there are classes of structures with incidence graphs of bounded tree-width and Gaifman graphs of unbounded tree-width [GS03]. The proof of Gottlob, Greco, and Scarcello is by a clever reduction. It would be interesting to investigate if some other VCSP($A$, _) could stand the same treatment. Of course, the result begs the question if MAX CSP($A$, _) is not in polynomial time when the incidence graphs of $A$ do not have bounded tree-width, or if there exists an even larger class of tractable instances.
In the two previous chapters we have investigated the computational complexity of CSP-related problems by considering fixed arbitrary values for some structural invariant of the (hyper)graph structure of the constraints. In this chapter, we instead investigate the MINHOM and VCSP problems by looking at them through the lens of parameterised complexity and considering the problems as parameterised by the tree-width of primal, dual, and incidence graphs, in combination with several other basic parameters such as domain size and arity.

5.1 Introduction

In this part of the thesis, we have been focusing on the so called structural restrictions. The algorithmic results described in previous chapters, where problem instances having a width parameter that is bounded by some fixed integer $w$ is considered, gives rise to a class $P_w$ of tractable instances. The larger $w$ gets, the larger is the resulting tractable class $P_w$. A typical time complexity of algorithms of this type from the literature are of the form $O(||I||^{f(w)})$ for instances $I$ from the class $P_w$. Hopefully, $f(w)$ denotes a slow growing function, but even then, since $w$ appears in the exponent, such algorithms become impractical when large instances are considered. It would be much better if we could find an algorithm with time complexity of the form $O(f(w) \cdot ||I||^c)$, where $f$ is an arbitrary computable function and $c$ is a constant independent of $w$ and $I$. The question of which width parameters allow algorithms of the later type for our optimization problems, and which do not, is the subject of this chapter of the thesis.
5. Parameterised Complexity

As mentioned in Chapter 2.3, parameterised complexity theory relaxes the tractability notion in exactly the way mentioned above — by admitting algorithms whose running time is exponential in some parameter of the problem instance that can be expected to be small in the typical application. In this chapter, we determine exactly those combinations of width parameters and basic parameters (to be defined later) such as number of variables, number of values, largest size of a constraint scope, etc. that make MinHom tractable in this fixed-parameter sense. To accomplish this, we make heavy use of the machinery developed by Samer and Szeider [SS10] for studying such parameterisations of the plain CSP. Remarkably, our results for MinHoms are analogous to those of Samer and Szeider [SS10] for classical CSP. As it turns out, acyclic MinHoms behave similarly to acyclic CSPs, so in hindsight the MinHom results may not be that surprising, though they still require different reductions than those used by Samer and Szeider for CSPs and usage of the non-trivial novel machinery of Gottlob et al. [GGS09].

We also make the same investigations for VCSP, but, interestingly enough, are not able to get a complete classification with respect to all parameters considered in this chapter. As we will see, while some hardness results escapes us, additional non-trivial arguments and more powerful reductions, coupled with techniques devised in Chapter 4, lets us extend all algorithmic results to the VCSP setting.

The rest of this chapter is organized as follows. Section 5.2 introduces the requisite parameterised problem definitions. Section 5.3 contains proofs of our tractability results, while Section 5.4 concerns the hardness results. Finally, Section 5.5 concludes the chapter and presents possible future work.

5.2 Parameterised Constraint Optimisation Problems

The following definitions and conventions used in the chapter are completely analogous for the VCSP case. We consider any computable function \( p \) that, to a MinHom instance \( I \), assigns a non-negative integer \( p(I) \) to be a MinHom parameter. For MinHom parameters \( p_1, \ldots, p_r \), we consider the following generic parameterised problem:

**MinHom**\((p_1, \ldots, p_r)\)

**Instance:** A MinHom instance \( I \) and non-negative integers \( k_1, \ldots, k_r \) with \( p_1(I) \leq k_1, \ldots, p_r(I) \leq k_r \).

**Parameters:** \( k_1, \ldots, k_r \)

**Output:** The cost of a minimum cost satisfying assignment for \( I \), “no” if \( I \) admits no such assignment.

By slight abuse of notation, we will also write MinHom\((S)\) for a set \( S \) of parameters, assuming an arbitrary but fixed ordering of the parameters in \( S \). We will write MinHom_{\text{Bool}}\((S)\) to denote MinHom\((S)\) with the domain \{0, 1\}, and MinHom_{\text{Bin}}\((S)\) to denote MinHom\((S)\) where all constraints have arity at most 2. Note that we formulate MinHom\((p_1, \ldots, p_r)\) as a
promises problem in the sense that for solving the problem we do not need to verify the assumption $p_1(I) \leq k_1, \ldots, p_r(I) \leq k_r$.

For a MINHOM instance $I$ we have the following width parameters:

- the tree-width of dual graphs, $\text{tw}^d$,
- the tree-width of incidence graphs, $\text{tw}^*$,
- the (generalized) hypertree-width, $(g)\text{hw}$,
- the spread-cut-width, $\text{scw}$,
- the fractional hypertree-width, $\text{fhw}$, and
- the submodular width, $\text{smw}$,

and the following basic parameters:

- the number of variables, $\text{vars}(I) = |V|$,
- the number of values, $\text{dom}(I) = |D|$,
- the number of constraints, $\text{cons}(I) = |C|$,
- the largest size of a constraint scope, $\text{arity}(I) = \max_{c \in C} |\text{var}(c)|$,
- the largest size of a relation, $\text{dep}(I) = \max_{c \in C} |\text{rel}(c)|$,
- the largest number of occurrences of a variable, $\text{deg}(I) = \max_{x \in V} |\text{con}(x)|$,
- the largest overlap between two constraint scopes, $\text{ovl}(I) = \max_{C, C' \in C, C \neq C'} |\text{var}(C) \cap \text{var}(C')|$, and
- the largest difference between two constraint scopes, $\text{diff}(I) = \max_{C, C' \in C} |\text{var}(C) \setminus \text{var}(C')|$.

The concept of domination among sets of parameters is heavily utilized by Samer and Szeider [SS10] and plays an equally important role in our classification, since it allows us to consider only a few border cases. Let $S$ and $S' = \{p'_1, p'_2, \ldots, p'_{r'}\}$ be two sets of MINHOM parameters. $S$ dominates $S'$ if for every $p \in S$ there exists an $r'$-ary computable function $f$ that is monotonically increasing in each argument, such that for every MINHOM instance $I$ we have $p(I) \leq f(p'_1(I), p'_2(I), \ldots, p'_{r'}(I))$. If $S$ or $S'$ is a singleton, we omit the braces to improve readability.
5. Parameterised Complexity

5.3 Tractability Results

If we parameterise MINHOM by the domain size, then solving the problem to optimality remains \( \text{NP}-\text{hard} \), since e.g. 3-colourability can be expressed as a MINHOM problem with constant domain. This means that \( \text{MINHOM}(\text{dom}) \) is not fixed-parameter tractable unless \( \text{P} = \text{NP} \). On the other hand, if we add the number of variables to the parameter list, then \( \text{MINHOM}(\text{vars}, \text{dom}) \) is trivially fixed-parameter tractable since we can find a minimum cost satisfying assignment to an instance \( I \) by checking all \( \text{dom}(I)^{\text{vars}(I)} \) possible assignments. However, if we do not bound the domain size, we obtain a \( \text{W}[1] \)-hard problem. This follows from the results of Papadimitriou and Yannakakis \cite{PY99} that CSP(\text{vars}) is a \( \text{W}[1] \)-complete problem. The hardness part of this theorem follows from a reduction from \( p\text{-CLIQUE} \). Since \( \text{tw}(I) \leq \text{vars}(I) \) holds by trivial reasons for every MINHOM instance \( I \), the \( \text{W}[1] \)-hardness of \( \text{MINHOM}(\text{tw}) \) is a direct consequence of this result.

We start by establishing the fundamental result that, despite the \( \text{W}[1] \)-hardness of \( \text{MINHOM}(\text{tw}) \), additionally bounding the domain size renders the problem fixed-parameter tractable by dynamic programming over tree decompositions. This provides a nice view on how the proofs of the forthcoming results will be constructed.

**Proposition 5.1.** \( \text{MINHOM}(\text{tw}, \text{dom}) \) is fixed-parameter tractable.

**Proof.** Let \( I = (V, D, C, \{c_d\}_{d \in D}) \) be an instance of \( \text{MINHOM}(\text{tw}, \text{dom}) \) with \( k = \text{tw}(I) \) and \( d = \text{dom}(I) \). Without loss of generality, we assume that no constraint scope in \( I \) contains multiple occurrences of variables (since they can be removed by simple preprocessing of \( I \)). From the bound \( k \) on the tree-width of the instance it follows that \( \text{arity}(I) \leq k + 1 \).

Let \( (T, (B_t)_{t \in V(T)}) \) be a tree decomposition of width \( k \) of the Gaifman graph of \( I \) with \( \mathcal{O}(|V|) \) nodes. Now we build a solution equivalent instance \( I' = (V, D, C', \{c'_d\}_{d \in D}) \). For every node \( t \in T \), the new instance \( I' \) has a constraint \( C' = (B_t, R_t) \), where \( R_t = D|B_t| \). Then, for every constraint \( C = (S, R) \in C \) of \( I \) such that \( S \subseteq B_t \), we remove all tuples from \( R_t \) that do not match with \( R \).

It is not hard to see that the instances \( I \) and \( I' \) have the same set of satisfying assignments. In the first step above, we make the constraint relations of \( I' \) contain all possible tuples for the variables in their constraint scopes, which means that any satisfying assignment of \( I \) is a satisfying assignment of \( I' \). Moreover, any satisfying assignment of \( I' \) is a satisfying assignment of \( I \), because every constraint of \( I \) must be satisfied, after the second step.

Note that computing \( I' \) from \( I \) is feasible in time \( \mathcal{O}(|V| \cdot d^{k+1}) \), making the transformation an fpt-reduction.

Observe that, by construction, \( (T, (B_t)_{t \in V(T)}) \) is a tree decomposition of \( I' \) such that the bags of the decomposition are hyperedges of the hypergraph of \( I' \), and that, because of the connectedness condition of tree decompositions \( (T, (B_t)_{t \in V(T)}) \) is a join tree of the hypergraph of \( I' \). As we now have an acyclic instance of MINHOM at hand, we can apply the algorithm...
of Gottlob et al. [GGS09] that lies behind Theorem 4.11 to solve $I'$ in time $O(|V| \cdot d^{k+1} \cdot \log(d^{k+1}))$. We note that dealing with the cost functions of $\text{MINHOM}$ does not provide any overhead, asymptotically, with respect to Yannakakis' algorithm for plain CSPs.

In the remainder of this section we set out to investigate the effect of different parameterisations on the fixed-parameter tractability of our optimization problems. As mentioned in the introduction, Samer and Szeider [SS10] have done precisely this for the plain CSP. Taking their lead, we will try to derive an analogue of their classification theorem for our optimization problems. We will start by proving our tractability results. The proofs of these results will have the VCSP problems as main focus, and derive the $\text{MINHOM}$ results as byproducts. As it turns out, we are only able to prove the hardness results needed for a complete classification theorem for the $\text{MINHOM}$ case, but we will discuss the shortcomings (with respect to VCSPs) of the current hardness proofs under way.

With tree-width of the dual graph as starting point, additionally bounding the largest difference between two constraint scopes, besides the domain size, turns out to yield fixed-parameter tractability.

**Proposition 5.2.** VCSP$(\text{tw}^d, \text{dom}, \text{diff})$ and $\text{MINHOM}(\text{tw}^d, \text{dom}, \text{diff})$ are fixed-parameter tractable.

**Proof.** Let $I = (V, D, C, \overline{Q})$ be an instance of VCSP$(\text{tw}^d, \text{dom}, \text{diff})$ with $k = \text{tw}^d(I)$, $d = \text{dom}(I)$, and $q = \text{diff}(I)$. Furthermore, let $(T, (B_t)_{t \in V(T)})$ be a tree decomposition of width $k$ of the dual graph of $I$ with $O(|C|)$ nodes. Now we build a solution equivalent instance $I' = (V, D, C', \overline{Q})$ by computing the join of all constraints in $B_t$ for every node $t \in V(T)$. Of course, the tuples in each join have to carry associated costs. This means that we add a pre-processing step before doing the join computation, where the cost of each tuple in each relation is divided by the number of bags that the corresponding constraint occurs in. Then, the cost of each tuple in a join becomes the sum of the (preprocessed) costs of the participating tuples. This way, we have preserved the costs of all solutions

In the same way as in the proof of Proposition 5.1 we note that, by construction, $(T, (B_t)_{t \in V(T)})$ is a tree decomposition of $I'$ such that the bags of the decomposition are hyperedges of the hypergraph of $I'$, and that, because of the connectedness condition of tree decompositions $(T, (B_t)_{t \in V(T)})$ is a join tree of the hypergraph of $I'$.

Observe that the join of two constraints, whose relations contain at most $p$ tuples, contains at most $pd^k$ tuples. This means that the join of $k + 1$ constraints can be computed in time $O(k p^2 d^k)$ and that there are at most $pd^{k(k+1)}$ tuples in each relation of $I'$. Since $k$, $d$, and $q$ are our parameters this means that we have an fpt-reduction from $I$ to $I'$. To finish up, we can now use the same structure preserving procedure as in Theorem 4.12 to transform $I'$ to an acyclic $\text{MINHOM}$ instance and ultimately
use the algorithm behind Theorem 4.11. Hence, $\mathcal{I}'$ can be solved in time $O(|\mathcal{C}| \cdot pd(q(k+1) \cdot \log(pdq(k+1)))$.

Finally, the corresponding MINHOM($tw^d$, dom, diff) case can obviously be handled by a simplified version of the above procedure. \hfill $\square$

With respect to tree-width of the dual graph it also suffices to bound the size of the largest relation to get fixed-parameter tractability.

**Proposition 5.3.** $\text{VCSP}(tw^d, \text{dep})$ and $\text{MINHOM}(tw^d, \text{dep})$ are fixed-parameter tractable.

**Proof.** Let $\mathcal{I} = (V, D, C, \{c_d\}_{d \in D})$ be an instance of $\text{VCSP}(tw^d, \text{dep})$. To start, we once again use the procedure outlined in the second part of the proof of Theorem 4.12, this time to produce a $\text{MINHOM}(tw^d, \text{dep})$ instance $\mathcal{I}'$, with $k = tw^d(\mathcal{I}')$ and $p = \text{dep}(\mathcal{I}')$, in linear time. To continue, we compute a tree decomposition $(T, (B_t)_{t \in V(T)})$ of the dual graph of $\mathcal{I}'$ of width $k$. This tree decomposition is then used to create an equivalent acyclic $\text{MINHOM}$ instance $\mathcal{I}''$ by computing the join of all constraints in $B_t$ for every node $t \in V(T)$. By the bound $k$ on the tree-width of the dual graph of $\mathcal{I}'$, each join computation involves at most $k + 1$ constraints and can be completed in time $O(kp^{k+1})$. Since $k$ and $p$ are our parameters this means that we have an fpt-reduction from $\mathcal{I}'$ to $\mathcal{I}''$.

Note also that each relation in $\mathcal{I}''$ has at most $p^{k+1}$ tuples. This, together with the fact that $\mathcal{I}''$ is an acyclic $\text{MINHOM}$ instance, which makes the $\text{MINHOM}$ algorithm of Gottlob et al. \cite{GCS09} applicable, establishes the proposition. \hfill $\square$

By generalizing an idea of Samer and Szeider \cite{SamerS10}, we can make further use of our new tractability results by a procedure called splitting, in which an instance of one of our optimization problems is transformed into an instance where each variable does not occur in the scope of more than three constraints. Let $\mathcal{I} = (V, D, C, \{c_d\}_{d \in D})$ be a $\text{MINHOM}$ instance, $x \in V$, and $\{C_1, \ldots, C_r \in \mathcal{C}\} = \text{con}(x)$, with $r > 3$. We construct a new $\text{MINHOM}$ instance $\mathcal{I}' = (V', D, C', \{c_d\}_{d \in D})$ as follows: Take $x' \notin V$ as a fresh variable and put $V' = V \cup \{x'\}$. We want to ensure that the new variable $x'$ plays the same role as $x$ in two of the constraints in $\mathcal{I}$. Take a new constraint $C_{x=x'} = ((x,x'), =_D)$, where $=_D$ is the equality relation on $D$, $\{(d,d) : d \in D\}$. For $i \in \{1, 2\}$, let $C'_i$ denote the new constraint we get from $C_i$ by replacing $x$ by $x'$ in the scope of $C_i$. Finally, put $C' = (C \setminus \{C_1, C_2\}) \cup \{C'_1, C'_2, C_{x=x'}\}$. By construction, $\mathcal{I}$ and $\mathcal{I}'$ are either both consistent or both inconsistent. To ensure that the new variables and constraints do not alter the cost of any satisfying assignment, we set $c_d(x') = 0$ for every $d \in D$. By repeating this construction $r - 3$ times, we are left with a $\text{MINHOM}$ instance where $x$ occurs in the scopes of at most three constraints. It is now possible to treat all other variables (that need treatment) in the same way to finally obtain an instance $\mathcal{I}^*$ where all variables occur in the scope of at most three constraints. If this is the case, we say that $\mathcal{I}^*$ is obtained from $\mathcal{I}$ by splitting.
To summarise, given a MINHOM instance \( I \), we can obtain in polynomial time a MINHOM instance \( I^* \) such that (i) a minimum cost homomorphism for \( I \) corresponds precisely to a minimum cost homomorphism for \( I^* \) and vice versa, and (ii) each variable of \( I^* \) occurs in the scopes of at most three constraints of \( I^* \). The procedure is polynomial time since we obtain \( I^* \) by repeating the steps sketched above \( \sum_{x \in V} \max(0, |\text{con}(x)| - 3) \) times.

In the VCSP case, the splitting procedure has to be slightly modified to take the generated equality constraints into account. This is easy to accomplish by setting the costs of tuples that are members of the equality relation on \( D \) to some fixed integer \( \varepsilon \) and all other costs to \( \infty \), and then remembering to subtract \( \varepsilon \) times the number of equality constraints from the cost of the optimal solution of the transformed instance.

It is important to note that, since different orderings of the constraints in each step are possible, splitting does not yield a unique instance \( I^* \). In particular, a bad choice of ordering can lead to an unbounded increase in incidence tree-width. Since splitting is a key part of our results we choose to illustrate this fact with a family of MINHOM instances \( I_n \) adapted from Samer and Szeider [SS10] (by giving larger examples and taking the opportunity to correct a few minor errors in their description of these instances). \( I_n \) has variables \( x_i \) for \( 1 \leq i \leq n \) and \( y_{ij} \) for \( 0 \leq i \leq n \) and \( 1 \leq j \leq 2n \). Furthermore, \( I_n \) has constraints \( C_{ij} \) for \( 0 \leq i \leq n \) and \( 0 \leq j \leq 2n \), with \( x_i \in \text{var}(C_{ij}) \) if and only if \( i \in \{i',i'+1\} \), and \( y_{ij} \in \text{var}(C_{ij}) \) if and only if \( i = i' \) and \( j \in \{j',j'+1\} \). Figure 5.1(a) is an illustration of the incidence graph of \( I_3 \). Since three cops can search the incidence graph of \( I_n \) in the Seymour-Thomas search game [Bod98], \( tw^*(I_n) = 3 \).

Now imagine we are going to split \( I_3 \) and that \( x_1 \) is the first variable picked. A possible ordering of the constraints with \( x_1 \) in their scopes is \((C_{1,6}, C_{0,6}, C_{1,5}, C_{0,5}, \ldots, C_{1,0}, C_{0,0})\). So, after the first step, a new variable \( x_{1}^{(1)} \) occurs in the scopes of \( C_{1,6} \) and \( C_{0,6} \) instead of \( x_1 \). Now, a possible ordering of \( \text{con}(x_1) \) is \((C_{x_1=x_{1}^{(1)}}, C_{1,5}, C_{0,5}, C_{1,4}, C_{0,4}, \ldots, C_{1,0}, C_{0,0})\). After this step, a new variable \( x_{1}^{(2)} \) occurs in the scopes of \( C_{x_1=x_{1}^{(1)}} \) and \( C_{1,5} \) instead of \( x_1 \), with a possible new ordering of constraints in \( \text{con}(x_1) \) being \((C_{x_1=x_{1}^{(2)}}, C_{0,5}, C_{1,4}, C_{0,4}, C_{1,3}, \ldots, C_{1,0}, C_{0,0})\). If we continue this procedure and apply it to \( x_2 \) and \( x_3 \) as well, we end up with an instance \( I_n^* \) with incidence graph as depicted in Figure 5.1(b). It is not hard to generalize this construction formally to arbitrary \( n \). Furthermore, it is easy to see that these incidence graphs \( I_n^* \) all contain a \((2n+1) \times (2n+1)\) grid as a minor. Using the now well-known facts, that the tree-width of an \( r \times r \) grid equals \( r \), and that the tree-width of a graph is at least as large as the tree-width of any of its minors (cf. Proposition 2.15), the conclusion is that \( tw^*(I_n^*) \geq 2n+1 \).

Fortunately, it is possible to choose the ordering of constraints in such a way that the incidence tree-width increases by at most one. Since our alterations to the procedure only involves manipulating costs, this follows easily from the corresponding result for the CSP case [SS10].
5. Parameterised Complexity

(a) The incidence graph of $I_3$.

(b) A possible incidence graph of $I^*_3$.

Figure 5.1: Black dots correspond to arbitrary constraints, gray squares to equality constraints, and white dots to variables.

Lemma 5.4. Given a MINHOM (VCSP) instance $I$ and a tree decomposition of width $k$ of the incidence graph of $I$, we can, in polynomial time, obtain a MINHOM (VCSP) instance $I^*$ with incidence tree-width at most $k + 1$ and construct a tree decomposition of the incidence graph of $I^*$ of width at most $k + 1$ such that each bag contains at most $k + 1$ variables.

Returning to our example, applying the splitting procedure implicit in this result to the MINHOM instance $I_n$ as defined above, we obtain a MINHOM instance of incidence tree-width $3$. The resulting incidence graph for the case $n = 3$ is depicted in Figure 5.2. Using the above lemma, we can now construct fpt-reductions to the corresponding fixed-parameter tractable cases of Proposition 5.2 and Proposition 5.3.

Proposition 5.5. VCSP($tw^*$, dep), MINHOM($tw^*$, dep), VCSP($tw^*$, dom, diff), and MINHOM($tw^*$, dom, diff) are fixed-parameter tractable.

Proof. Let $I = (V, D, C, \{c_d\}_{d \in D})$ be an instance of MINHOM($tw^*$, dep) with $k = tw^I(I)$ and $p = dep(I)$. First we compute a tree decomposition $(T, (B_t)_{t \in V(T)})$ of the incidence graph of $I$ of width $k$. Then we apply Lemma 5.4 to obtain, by splitting, the instance $I^*$ and an accompanying tree decomposition $(T', (B_t)_{t \in V(T')})$ of the incidence graph of $I^*$ of width at most $k + 1$. To continue, we would like a tree decomposition $(T'', (B_t)_{t \in V(T'')})$ of the dual graph of $I^*$. This can be obtained by replacing each variable $x$ in each bag of $(T', (B_t)_{t \in V(T')})$ by the constraints in $I^*$ in which $x$ occurs. Since each bag of $(T', (B_t)_{t \in V(T')})$ contains at most $k + 1$ variables, the width...
5.3. Tractability Results

of $(T', (B_i)_{i \in V(T')})$ is at most $3(k + 1)$. Finally we can let $k' = 3(k + 1)$ and $p' = \max(p, |=D|)$. In this way, $(I^*, k', p')$ is an instance of $\text{MinHom}(tw^d, \text{dep})$.

The result for $\text{MinHom}(tw^*, \text{dom}, \text{diff})$ follows completely analogously and by observing that $\text{dom}(I^*)$ and $\text{diff}(I^*)$ are bounded by computable functions of $\text{dom}(I)$ and $\text{diff}(I)$, respectively. Samer and Szeider call such parameters splitting compatible.

The corresponding VCSP cases are handled analogously, by using the splitting procedure to get an fpt-reduction to a known fixed-parameter tractable case, remembering to account for the costs of the generated equality constraints as mentioned in the description of the splitting procedure. □

In contrast with our earlier proofs, and all tractability results of Samer and Szeider [SS10], this final result makes involved use of the more powerful fpt Turing reduction.

**Proposition 5.6.** $\text{VCSP}(\text{dom}, \text{cons}, \text{ovl})$ and $\text{MinHom}(\text{dom}, \text{cons}, \text{ovl})$ are fixed-parameter tractable.

**Proof.** In what follows, we will need the fact that $\text{MinHom}(\text{dom}, \text{cons}, \text{arity})$ is fixed-parameter tractable. To this end, let $\mathcal{I} = (V, D, C, \{c_d\}_{d \in D})$ be an instance of $\text{MinHom}(\text{dom}, \text{cons}, \text{arity})$ with $m = \text{dom}(\mathcal{I})$, $c = \text{cons}(\mathcal{I})$, and $a = \text{arity}(\mathcal{I})$. Observe that $\text{dep}(\mathcal{I}) \leq m^a$ holds trivially. Furthermore, there is a trivial tree decomposition of the dual graph of $\mathcal{I}$ of width $\text{cons}(\mathcal{I}) - 1$. Thus, $\text{tw}^d(\mathcal{I}) \leq c - 1$. This means that we have an fpt-reduction from $\text{MinHom}(\text{dom}, \text{cons}, \text{arity})$ to $\text{MinHom}(\text{tw}^d, \text{dep})$ and the fact follows.

To proceed, we let $\mathcal{I} = (V, D, C, \{c_d\}_{d \in D})$ be an instance of $\text{MinHom}(\text{dom}, \text{cons}, \text{ovl})$ with $m = \text{dom}(\mathcal{I})$, $c = \text{cons}(\mathcal{I})$, and $l = \text{ovl}(\mathcal{I})$. By
our parameterisation, each constraint has at most \( l \cdot (c - 1) \) variables in its scope occurring in the scopes of other constraints in \( \mathcal{I} \). This means that if all we wanted was to decide the consistency of \( \mathcal{I} \), only these \( l \cdot (c - 1) \) variables would be relevant and all others could be projected out. Hence, we could transform \( \mathcal{I} \) by an fpt-reduction into an instance \( \mathcal{I}' \) with \( \text{dom}(\mathcal{I}') = m \), \( \text{cons}(\mathcal{I}') = c \), and \( \text{arity}(\mathcal{I}') \leq l \cdot (c - 1) \). This is one way of showing that CSP(dom, cons, ovl) is fixed-parameter tractable.

However, in this case we can not be sure that the minimum cost satisfying assignment for \( \mathcal{I}' \) returned by the call to the algorithm for \( \text{MINHOM}(\text{dom}, \text{cons}, \text{arity}) \) actually is part of a minimum cost satisfying assignment to our instance \( \mathcal{I} \) of \( \text{MINHOM}(\text{dom}, \text{cons}, \text{ovl}) \). Let \( \sigma' : V' \rightarrow D \) be the assignment returned by the call to the algorithm for \( \text{MINHOM}(\text{dom}, \text{cons}, \text{arity}) \). It is now straightforward to go through the tuples in the relation for each constraint \( c \in \mathcal{C} \), isolate the ones consistent with \( \sigma' \), check which one of the isolated tuples have the minimum cost, extend \( \sigma' \) to an assignment \( \sigma : V \rightarrow D \) having minimum possible cost given the partial assignment \( \sigma' \), and record \( \sigma \) as a candidate solution to \( \mathcal{I} \). We note that this is possible to do in time polynomial in the size of \( \mathcal{I} \).

The next step is to remove all tuples from the relations of \( \mathcal{I}' \) consistent with \( \sigma' \) (i.e. one tuple from each relation of every constraint in \( \mathcal{I}' \)) and repeat the whole process of calling the algorithm for \( \text{MINHOM}(\text{dom}, \text{cons}, \text{arity}) \) with the new (smaller) \( \mathcal{I}' \) and extending the returned partial assignment \( \sigma' \) to an assignment \( \sigma : V \rightarrow D \), consistent with \( \sigma' \), having minimum possible cost. If this cost is smaller than the cost for our current candidate solution we have obtained a new solution candidate.

By repeating this process we will eventually end up with the instance \( \mathcal{I}' \) being trivially inconsistent because at least one of its constraint relations will be empty. This will occur in a number of steps that is polynomial in the size of \( \mathcal{I} \), since we remove a tuple from each constraint relation of \( \mathcal{I}' \) in each step. Thus, we have an fpt Turing reduction from \( \text{MINHOM}(\text{dom}, \text{cons}, \text{ovl}) \) to \( \text{MINHOM}(\text{dom}, \text{cons}, \text{arity}) \).

The VCSP(dom, cons, ovl) case is very similar to the corresponding MINHOM case. In fact, we can use the fpt-algorithm for \( \text{MINHOM}(\text{dom}, \text{cons}, \text{arity}) \) as a subroutine. Let \( \mathcal{I} = (\mathcal{V}, \mathcal{D}, \mathcal{C}, \mathcal{Q}) \) be an instance of VCSP(dom, cons, ovl) with \( m = \text{dom}(\mathcal{I}), c = \text{cons}(\mathcal{I}), \) and \( l = \text{ovl}(\mathcal{I}) \). First, we drop all tuples in each relation that have infinite cost. Now, compute an instance \( \mathcal{I}' \) of \( \text{MINHOM}(\text{dom}, \text{cons}, \text{arity}) \) with the \( l \cdot (c - 1) \) variables that decides consistency of \( \mathcal{I} \) (without the dropped tuples). The costs of \( \mathcal{I}' \) may be set uniformly to zero. The remainder of the fpt Turing reduction is constructed analogously to the corresponding MINHOM case.

### 5.4 The Domination Lattice and Hardness Results

As nice as the above tractability results may be, we see that, for all we know, we could need literally over two thousand similar results to achieve a com-
5.4. The Domination Lattice and Hardness Results

plete classification with respect to all parameters under consideration. However, looking a bit closer at the proof of Proposition 5.6, in particular the part where fixed-parameter tractability of $\text{MINHOM}(\text{dom, cons, arity})$ is established gives us hope. Perhaps it is the case that the kind of fpt-reduction used would be very common in a complete classification? This is what leads us to explore the concept of domination.

**Lemma 5.7.** Let $S$ and $S'$ be two sets of $\text{MINHOM}$ parameters such that $S$ dominates $S'$. Then $\text{MINHOM}(S')$ fpt-reduces to $\text{MINHOM}(S)$. The same holds for such VCSP parameters.

**Proof.** The lemma is almost immediate from the respective definitions. Let $S = \{p_1, p_2, \ldots, p_r\}$ and $S' = \{p'_1, p'_2, \ldots, p'_r\}$ such that $S$ dominates $S'$. By the definition of domination, there exists, for every $i = 1, \ldots, r$, an $r'$-ary computable function $f_i$ that is monotonically increasing in each argument such that for every $\text{MINHOM}$ instance $I$ we have $p_i(I) \leq f_i(p'_1(I), p'_2(I), \ldots, p'_r(I))$.

Consider an instance $(I, k'_1, \ldots, k'_r)$, where we have $p'_i(I) \leq k'_i$ for $1 \leq i \leq r'$ of $\text{MINHOM}(S')$. We now describe an fpt-reduction from $\text{MINHOM}(S')$ to $\text{MINHOM}(S)$. Put $k_i = f_i(k'_1, \ldots, k'_r)$ for $1 \leq i \leq r$. Since $f_i$ is monotonically increasing in each argument, we have $p_i(I) \leq f_i(p'_1(I), p'_2(I), \ldots, p'_r(I)) \leq f_i(k'_1, \ldots, k'_r) \leq k_i$. Hence $(I, k_1, \ldots, k_r)$ is an instance of $\text{MINHOM}(S)$ and we have our fpt-reduction.

The VCSP case is completely analogous. \qed

The following key lemma contains all domination results we need for our classification. Each result is straightforward to prove from basic and well-known properties of the parameters involved. To make the exposition clear and to convey the kind of considerations involved we repeat the proof of the lemma here. Note that parts 2, 4, and 6–15 are strict in the sense that $p$ dominates $p'$ but $q$ does not dominate $p$.

**Lemma 5.8** ([SS10] Lemma 2).

1. If $S \subseteq S'$, then $S$ dominates $S'$.
2. $\text{tw}$ dominates $\text{vars}$.
3. $\text{tw}$ dominates $\{\text{tw}^*, \text{arity}\}$.
4. $\text{tw}^d$ dominates $\text{cons}$.
5. $\text{tw}^d$ dominates $\{\text{tw}^*, \text{deg}\}$.
6. $\text{tw}^*$ dominates $\text{tw}$.
7. $\text{tw}^*$ dominates $\text{tw}^d$.
8. $\text{vars}$ dominates $\{\text{cons, arity}\}$.
9. $\text{dom}$ dominates $\{\text{cons, arity, dep}\}$.
10. $\text{cons}$ dominates $\{\text{vars, deg}\}$.
11. $\text{arity}$ dominates $\text{tw}$.
12. $\text{dep}$ dominates $\{\text{dom, arity}\}$.
13. $\text{deg}$ dominates $\text{tw}^d$.
14. $\text{ovl}$ dominates arity.
15. $\text{diff}$ dominates arity.
5. PARAMETERISED COMPLEXITY

Proof.

1. Obvious from the definition of domination.

2. There is always a trivial tree decomposition of the Gaifman graph of $\mathcal{I}$ of width $\text{vars}(\mathcal{I}) - 1$. Thus, $\text{tw}(\mathcal{I}) \leq \text{vars} - 1$.

3. As observed by Kolaitis and Vardi [KV00], every tree decomposition of the incidence graph can be transformed into a tree decomposition of the primal graph by replacing each constraint $C$ in the bags by $\text{var}(C)$. Thus, $\text{tw}(\mathcal{I}) \leq \text{tw}^*(\mathcal{I})(\text{arity}(\mathcal{I}) - 1)$.

4. There is always a trivial tree decomposition of the dual graph of $\mathcal{I}$ of width $\text{cons}(\mathcal{I}) - 1$. Thus, $\text{tw}^d(\mathcal{I}) \leq \text{cons} - 1$.

5. Every tree decomposition of the incidence graph can be transformed into a tree decomposition of the dual graph by replacing each variable $x$ in the bags by $\text{con}(x)$. Thus, $\text{tw}^d(\mathcal{I}) \leq \text{tw}^*(\mathcal{I})(\text{deg}(\mathcal{I}) - 1)$. This is symmetric to case 3.

6. This follows from the fact that $\text{tw}^*(\mathcal{I}) \leq \text{tw}(\mathcal{I}) + 1$ holds, as shown by Kolaitis and Vardi [KV00].

7. A symmetric argument to case 6 gives $\text{tw}^*(\mathcal{I}) \leq \text{tw}^d(\mathcal{I}) + 1$, and the case follows.

8. Recall that we assume w.l.o.g. that every variable occurs in at least one scope and every domain element occurs in at least one relation. It is obvious that $\text{vars}(\mathcal{I}) \leq \text{cons}(\mathcal{I}) \cdot \text{arity}(\mathcal{I})$.

9. Recalling the same assumption as in case 8, it is obvious that $\text{dom}(\mathcal{I}) \leq \text{cons}(\mathcal{I}) \cdot \text{arity}(\mathcal{I}) \cdot \text{dep}(\mathcal{I})$.

10. Obviously, $\text{cons}(\mathcal{I}) \leq \text{vars}(\mathcal{I}) \cdot \text{deg}(\mathcal{I})$.

11. A constraint of arity $r$ yields a clique on $r$ vertices in the Gaifman graph. It is well known that if a graph $G$ contains a clique on $r$ vertices then $\text{tw}(G) \geq r - 1$ [Bod98]. Thus, $\text{arity}(\mathcal{I}) \leq \text{tw}(\mathcal{I}) + 1$.

12. Obviously, $\text{dep}(\mathcal{I}) \leq \text{dom}(\mathcal{I})^{\text{arity}(\mathcal{I})}$.

13. A symmetric argument to case 11 (i.e. that a variable of degree $r$ yields a clique on $r$ vertices in the dual graph) gives $\text{deg}(\mathcal{I}) \leq \text{tw}^d(\mathcal{I}) + 1$.

14. Obviously, $\text{ovl}(\mathcal{I}) \leq \text{arity}(\mathcal{I})$.

15. Obviously, $\text{diff}(\mathcal{I}) \leq \text{arity}(\mathcal{I})$.

Starting from the fixed-parameter tractability results we have at hand, we can use Lemmas 5.7 and 5.8 to identify further subsets $S$ of parameters for which $\text{MinHom}(S)$ and $\text{VCSP}(S)$ are fixed-parameter tractable.
Corollary 5.9. Let \( S \subseteq \{ \text{tw}, \text{tw}', \text{tw}^*, \text{vars}, \text{dom}, \text{cons}, \text{arity}, \text{dep}, \text{deg}, \text{ovl}, \text{diff} \} \). Then \( \text{MINHOM}(S) \) and \( \text{VCSP}(S) \) are fixed-parameter tractable if \( S \) contains at least one of the following 14 sets as subset:

- \( \{ \text{dom}, \text{cons}, \text{arity} \} \)
- \( \{ \text{dom}, \text{cons}, \text{ovl} \} \)
- \( \{ \text{dom}, \text{cons}, \text{diff} \} \)
- \( \{ \text{tw}', \text{dom}, \text{arity} \} \)
- \( \{ \text{tw}', \text{dom}, \text{diff} \} \)
- \( \{ \text{tw}, \text{dom}, \text{diff} \} \)
- \( \{ \text{tw}', \text{dep} \} \)
- \( \{ \text{tw}', \text{dom}, \text{diff} \} \)
- \( \{ \text{tw}, \text{dep} \} \)
- \( \{ \text{tw}', \text{dom}, \text{arity} \} \)

As a matter of fact, going through all possible subsets identifies three sole “sources” of fixed-parameter tractability; all other subsets that give rise to fixed-parameter tractable parameterised \( \text{MINHOM} \) and \( \text{VCSP} \) problems are dominated by at least one of \( \{ \text{tw}', \text{dep} \} \), \( \{ \text{tw}', \text{dom}, \text{diff} \} \), and \( \{ \text{dom}, \text{cons}, \text{ovl} \} \). Let us assume that \( \text{MINHOM}(S) \) is \( \text{W}[1] \)-hard for the sets of parameters that are not dominated by these three sets. This would give us a complete classification:

Theorem 5.10 (Classification Theorem). Let \( S \subseteq \{ \text{tw}, \text{tw}', \text{tw}^*, \text{vars}, \text{dom}, \text{cons}, \text{arity}, \text{dep}, \text{deg}, \text{ovl}, \text{diff} \} \). If \( \{ \text{tw}', \text{dep} \} \), \( \{ \text{tw}', \text{dom}, \text{diff} \} \), or \( \{ \text{dom}, \text{cons}, \text{ovl} \} \) dominates \( S \), then \( \text{MINHOM}(S) \) is fixed-parameter tractable. Otherwise \( \text{MINHOM}(S) \) is not fixed-parameter tractable (unless \( \text{FPT} = \text{W}[1] \)).

By Lemma 5.8 this means that for each of the sets \( S \) that are not dominated by \( \{ \text{tw}', \text{dep} \} \), \( \{ \text{tw}', \text{dom}, \text{diff} \} \), or \( \{ \text{dom}, \text{cons}, \text{ovl} \} \), if \( S' \subseteq S \), then \( \text{MINHOM}(S') \) is also \( \text{W}[1] \)-hard. Consequently, it is sufficient to consider those sets of parameters that are not subsets of another set. Doing this yields a characterization dual to Corollary 5.9 which, together with the sets listed in Corollary 5.9 establishes the domination lattice in Figure 5.3. If two sets of parameters are domination equivalent (i.e., if they dominate each other), we only consider one of them in the lattice.

The proof of the Classification Theorem uses the following proposition, which can easily be read off from the domination lattice.

Proposition 5.11. Let \( S \subseteq \{ \text{tw}, \text{tw}', \text{tw}^*, \text{vars}, \text{dom}, \text{cons}, \text{arity}, \text{dep}, \text{deg}, \text{ovl}, \text{diff} \} \). If \( S \) is not dominated by any of \( \{ \text{tw}', \text{dep} \} \), \( \{ \text{tw}', \text{dom}, \text{diff} \} \), or \( \{ \text{dom}, \text{cons}, \text{ovl} \} \), then \( S \) dominates \( \{ \text{dom}, \text{cons} \} \), \( \{ \text{tw}', \text{dom}, \text{ovl} \} \), \( \{ \text{cons}, \text{arity} \} \), or \( \{ \text{dom}, \text{arity}, \text{deg} \} \).

Proof (of the Classification Theorem). In view of Proposition 5.11 the theorem is established if we show \( \text{W}[1] \)-hardness of (i) \( \text{MINHOM}(\text{dom}, \text{cons}) \), (ii) \( \text{MINHOM}(\text{tw}', \text{dom}, \text{ovl}) \), (iii) \( \text{MINHOM}(\text{cons}, \text{arity}) \), and (iv) \( \text{MINHOM}(\text{dom}, \text{arity}, \text{deg}) \). Before going any further we have to observe that the corresponding four parameterised CSP problems are all \( \text{W}[1] \)-hard by the results of Samer and Szeider [SS10]. This means that the four \( \text{MINHOM} \) hardness proofs we now seek are trivial consequences of Samer and Szeider’s results. In principle, we would also like to show the four corresponding \( \text{VCSP} \) problems to be \( \text{W}[1] \)-hard. This motivates looking closer at the proofs of Samer and Szeider [SS10] to see if they could be converted also to proofs for \( \text{VCSP} \).
Figure 5.3: Domination lattice. The sets $S$ for which MinHom($S$) is not fixed-parameter tractable are indicated by shaded boxes. A set $S$ dominates a set $S'$ if and only if there is a path starting from $S$ and running upwards to $S'$. Domination between sets in the lattice is strict.

(i): Samer and Szeider [SS10] have shown that CSP_{BOOLE}(cons) is $W[1]$-hard. Thus, we immediately have that MinHom_{BOOLE}(cons) is $W[1]$-hard. The proof is by a simple reduction from $p$-CLIQUE. Unfortunately, the relation constructed in this reduction is a subset of $\{0, 1\}^{2^n}$, where $n$ is the number of vertices in the graph we are trying to find a clique in, so we would not have time to construct a corresponding VCSP instance while keeping the procedure an fpt-reduction.

(ii): In this case, Samer and Szeider [SS10] have demonstrated a clever reduction from $p$-INDEPENDENT SET to CSP_{BOOLE}(tw^d, ovl), thus showing this problem to be $W[1]$-hard. This result implies MinHom_{BOOLE}(tw^d, ovl) to be $W[1]$-hard. Using this reduction for the VCSP case fails for similar reasons as in case (i).

(iii): For this case, the result holds even if we only allow binary constraints, i.e., MinHom_{BIN}(cons) is $W[1]$-hard. Samer and Szeider [SS10] have shown that CSP_{BIN}(cons) is $W[1]$-hard. The proof is by the same reduction from $p$-CLIQUE that Papadimitriou and Yannakakis [PY99] used to show $W[1]$-hardness of CSP(vars). Since the arity of all generated constraints in this reduction is 2, we can actually use it to prove that also VCSP(cons, arity) is $W[1]$-hard.

(iv): Samer and Szeider [SS10] have shown that CSP_{BIN}(dom, deg) is NP-hard, due to a reduction from graph 3-colourability. As in case (iii), also this reduction can be used to show $W[1]$-hardness of VCSP(dom, arity, deg). In contrast to the previous hardness results, we show that the MinHom case is NP-hard to solve to optimality due to the fact that the problem can be used to encode the MAXIMUM INDEPENDENT SET problem in bounded degree.
5.5. Conclusions

graphs. Hence, \( \text{MINHom}(\text{dom}, \text{arity}, \text{deg}) \) is not fixed-parameter tractable unless \( P = NP \). This holds even if we allow only binary constraints and Boolean domain.

So, we want to show that \( \text{MINHom}_{\text{BIN,BOOLE}}(\text{deg}) \) is \( \text{NP}-\text{hard} \). Consider a graph \( G = (V, E) \). Assume that each vertex has degree at most 3, since \( \text{MAXIMUM INDEPENDENT SET} \) remains \( \text{NP}-\text{hard} \) under this restriction \cite{GJS74}. We construct a binary Boolean instance of \( \text{MINhom}(\text{dom}, \text{arity}, \text{deg}) \)\( I = (V, \{0, 1\}, C) \), where \( C \) contains constraints \( ((u, v), (0, 0), (0, 1), (1, 0)) \) for every \( uv \in E \). It is not hard to see that the set of satisfying assignments of \( I \) characterizes the independent sets in \( G \) and, thus, that \( \text{MINHom}(\text{dom, arity, deg}) \) is equivalent to the \( \text{WEIGHTED MAXIMUM INDEPENDENT SET} \) problem on instances of this type. Since \( \text{deg}(I) = 3 \), the result follows.

As is evident from the proof of Theorem 5.10, we are only able to, at the time of writing, prove \( \text{W}[1]\)-hardness of \( \text{VCSP}(\text{cons, arity}) \) and \( \text{VCSP}(\text{dom, arity, deg}) \), which is why we do not have a complete classification for this problem type. As a final observation we note that the notion of domination lets us extend the \( \text{W}[1]\)-hardness results of the Classification Theorem to all parameters that are more general than the tree-width of incidence graphs.

Corollary 5.12. The problems \( \text{MINHom}(p, \text{dom}) \) and \( \text{MINHom}_{\text{BOOLE}}(p) \) are \( \text{W}[1]\)-hard if \( p \) is any of the parameters \( \text{tw}^*, (g)\text{hw}, \text{scw}, \text{fhw}, \text{and smw}. \)

5.5 Conclusions

In this chapter of the thesis, we have built extensively on the framework of Samer and Szeider \cite{SS10} to classify the parameterised complexity of \( \text{MINHom} \) and \( \text{VCSP} \) problems for combinations of natural parameters including the tree-width of primal, dual, and incidence graphs, domain size, largest size of a constraint relation, and the largest size of a constraint scope. The most obvious direction of further research is to try and find the missing \( \text{W}[1]\)-hardness proofs for \( \text{VCSP}(\text{dom, cons}) \) and \( \text{VCSP}(\text{tw}^d, \text{dom, ovl}) \) required to get a complete classification result also for \( \text{VCSP} \).

A priori, there seems to be no apparent reason why our optimization problems should behave exactly the same as in the CSP classification of Samer and Szeider \cite{SS10}, that is, that the problems should be fixed-parameter tractable for exactly the same parameterisations. Indeed, our tractability results depend on the \( \text{MINHom} \) algorithm of Gottlob et al. \cite{GCS09}, which surprisingly does not provide any overhead, asymptotically, with respect to Yannakakis’ algorithm for plain CSPs. Furthermore, we have had to employ our techniques from Chapter 4, resort to various other non-trivial tricks and, in some cases, devised a stronger type of structure preserving reduction than Samer and Szeider, that nevertheless preserves fixed-parameter tractability. Hence, examples of parameters that separate \( \text{MINHom} \)-
5. Parameterised Complexity

MINHOM, VCSP, and CSP fixed-parameter complexity would be most enlightening. Flum and Grohe [FG02] have developed a parameterised complexity theory for counting problems. Using their framework to study parameterisations of #CSP similar to those considered in this chapter would of course be an interesting subject.

Finally we want to point out that, as an immediate corollary to the hardness proof for MINHOM(dom, arity, deg), we receive that this problem is APX-hard, and that MINHOM(dom, arity) is not even in APX, but poly-APX-hard and -complete under appropriate reductions. This follows from well-known results on the approximability of MAXIMUM INDEPENDENT SET [BEP05, BF95, KMSV98, PY91]. As a further example of the expressivity of MINHOM we note that it is not hard to show that MINHOM(dom, arity) contains MAX CUT and NEAREST CODEWORD as subproblems. Perhaps there are similar ways of showing approximation hardness for the three other W[1]-hard key cases of MINHOM?
Separation, Graphs, and Approximability
In this part of the thesis we introduce a new method for studying the approximability of optimisation problems. This method lets us draw conclusions concerning the approximability of one optimisation problem based on the approximability properties of another optimisation problem. In Part II of the thesis, we saw some results concerning the structurally restricted $\text{MAX CSP}$ problem. The motivating question behind the work in this part instead concerns the approximability properties of $\text{MAX CSP}$ under certain constraint language restrictions: Imagine that we have a good approximation algorithm for the $\text{MAX CSP}(\_, \{H\})$ problem, for some simple undirected graph $H$. If we now are given an instance of $\text{MAX CSP}(\_, \{G\})$ for some other graph $G$, such that $H \rightarrow G$, how well would we be off if we simply applied the approximation algorithm for $\text{MAX CSP}(\_, \{H\})$ to the $\text{MAX CSP}(\_, \{G\})$ instance? Intuitively, we expect that if $H$ and $G$ are similar, in some vague sense, a bound on the approximability of $\text{MAX CSP}(\_, \{H\})$ could perhaps be translated to some useful bound on the approximability of $\text{MAX CSP}(\_, \{G\})$. Indeed, this is precisely what happens.

More concretely, our question above lead us to introduce, and study, an instance of a type of approximation-preserving reduction called continuous reduction [Sim89]. A continuous reduction allows the transfer of constant ratio approximation results from one optimisation problem to another. In a wholly abstract setting our novel method relies on a binary parameter on optimisation problems which measures the degradation in the approximation guarantee of a given continuous reduction. We call this parameter the separation of the two problems.
The separation parameter is then used to study a concrete family of optimisation problems called the maximum $H$-colourable subgraph problems, or MAX $H$-COL for short. This family includes the problems MAX $k$-CUT for which good approximation ratios are known [FJ97]. Starting from these ratios, we use the notion of separation to obtain general approximation results for MAX $H$-COL. Our main contribution in relation to approximation is Theorem 6.31 which gives a constant approximation ratio of \(1 - \frac{1}{r} + \frac{2 \ln k}{k^2} (1 - \frac{1}{r} + o(1))\) for MAX $H$-COL, when the graph $H$ has clique number $r$ and chromatic number $k$. Asymptotically, this result outperforms the best algorithm for approximating MAX $H$-COL, for arbitrary graphs $H$, also having an easily computable bound on the guaranteed performance. Theorem 6.31 also says that it is NP-hard to approximate MAX $H$-COL better than within \(1 - \frac{1}{r} + \frac{2 \ln r}{r^2} (1 + o(1))\) when the graph $H$ has clique number $r$ and chromatic number $k$, conditioned on the truth of the unique games conjecture (UGC) [Kho02].

Actually, the MAX $H$-COL problem is precisely the MAX CSP($\Gamma$, $\{H\}$) problem mentioned in the motivating question behind all the work in this part. As we already noted, this problem is a special case to the general MAX CSP problem. In the literature, the established way to state this is to say that MAX $H$-COL is a MAX CSP($\Gamma$) problem, where $\Gamma$, the constraint language, is the set containing the single, binary, and symmetric relation given by the edge set of $H$. Raghavendra [Rag08] has presented approximation algorithms for every MAX CSP($\Gamma$) problem based on semi-definite programming. Under the UGC, these algorithms optimally approximate MAX CSP($\Gamma$) in polynomial-time, i.e. no other polynomial-time algorithm can approximate the problem substantially better. However, it seems notoriously difficult to determine the approximation ratio implied by this result, for a given constraint language: Raghavendra and Steurer [RS09] show that this ratio can in principal be computed, but the algorithm is doubly exponential in the size of the domain. In combination with our results, such ratios could be used to confirm or disprove the UGC.

In the setting of MAX $H$-COL, we view separation as a binary graph parameter. While the initial motivation for introducing this parameter was to study the approximability of optimisation problems, it turns out that separation is a parameter of independent interest in graph theory. We investigate this aspect of separation in Chapter 7. Among the most striking results is the connection between separation and a (generalisation of) cubical colourings and fractional covering by cuts previously studied by Šámal [Šam05, Šam06, Šam12]. This connection uncovers a correspondence between a family of chromatic numbers and our separation parameter. These chromatic numbers are a generalisation of Šámal’s cubical chromatic number. We believe that the alternative view on our parameter these insights provide can give great benefits towards the understanding of its properties.
6.1 Introduction

Before we consider continuous reductions and the separation parameter, we begin by recalling the definition of abstract optimisation problems. An optimisation problem \( M \) is defined over a set of instances \( I_M \); each instance \( I \in I_M \) has an associated finite set \( \text{Sol}_M(I) \) of feasible solutions. The objective is, given an instance \( I \), to find a feasible solution of optimum value, with respect to some measure (objective function) \( m_M(I, f) \), where \( f \in \text{Sol}_M(I) \). The optimum of \( I \) is denoted by \( \text{Opt}_M(I) \), and is defined as the largest measure of any solution to \( I \). We will make the assumption that every instance of every problem considered has some feasible solution and that all feasible solutions have positive rational measure. Then, the following quantity is always defined, where \( I \in I_M \) and \( f \in \text{Sol}_M(I) \).

\[
R_M(I, f) = \frac{m_M(I, f)}{\text{Opt}_M(I)}.
\]

A solution \( f \in \text{Sol}_M(I) \) to an instance \( I \) of a maximisation problem \( M \) is called \( r \)-approximate if it satisfies

\[
R_M(I, f) \geq r.
\]

Example 6.1. In Example 1.12 we saw that the objective of the MAXIMUM INDEPENDENT SET problem is to find an independent set of maximum size in a given input graph \( G = (V, E) \). If we restrict the instances of the problem to non-empty simple graphs \( G \), and disallow the empty set as a solution, we know that every instance of the problem has some feasible solution and that all feasible solutions have positive measure. In the MAXIMUM SET PACKING problem, we are given a system \( S = S_1, \ldots, S_r \) of finite sets, and the objective is to compute a maximum set packing, i.e., a maximum number of pairwise disjoint sets from the system. For this problem, we can also disallow the empty set as a feasible solution to make sure the problem conforms with our assumptions above.

Our main focus will be on the following type of reduction.

Definition 6.2 (Simon [Sim89]; Crescenzi [Cre97]). A reduction \( \phi, \gamma \) from \( N \) to \( M \) is called a continuous reduction if a positive constant \( \alpha \) exists such that, for every \( I \in I_N \) and \( f \in \text{Sol}_M(\phi(I)) \), it holds that

\[
R_M(\phi(I), f) \geq \alpha \cdot R_M(\phi(I), f). \tag{6.1}
\]

Example 6.3. Continuous reductions between the problems from Example 6.1 with \( \alpha = 1 \) can be obtained from well-known mutual reductions between the two problems: We associate with each vertex \( v \in V \) the set of its incident edges. A subset \( V' \subseteq V \) is the independent if and only if the corresponding subsystem of sets is a set packing. A converse reduction with \( \alpha = 1 \) is obtained by associating a vertex with each set of the system. Two vertices are connected by an edge if and only if the corresponding sets have

\[\text{We will only consider maximisation problems in this part of the thesis.}\]
6. Separation and Approximability

A non-empty intersection. A subsystem of sets is then a set packing if and only if the corresponding vertices are independent.

Every continuous reduction is also an \( \mathcal{A} \)-reduction and hence preserves membership in \( \text{APX} \). More specifically, we have the following fact.

**Proposition 6.4** (Simon [Sim89]). Assume that there is a continuous reduction from \( N \) to \( M \) with a constant \( \alpha \). If \( M \) can be approximated within a constant ratio \( r \), then \( N \) can be approximated within \( \alpha \cdot r \).

Proposition 6.4 says that it is possible to approximate the problem of \( M \) to a certain degree, \( \alpha \cdot r \), and that this approximation can be cascaded to \( N \) through the \( \mathcal{A} \)-reduction.

Needless to say, the separation is difficult to compute in the general case. Thus, we henceforth concentrate on one particular optimisation problem that is parameterised by graphs. It is, however, important to note that the parameter \( s \) can be defined over many different types of optimisation problems and it is by no means restricted to problems parameterised by graphs.

### 6.1.1 The Maximum \( H \)-Colourable Subgraph Problem

Denote by \( \mathcal{G} \) the set of all non-empty, simple, and undirected graphs. For a graph \( G \in \mathcal{G} \), let \( \mathcal{W}(G) \) be the set of weight functions \( w : E(G) \rightarrow \mathbb{Q}_{\geq 0} \) such that \( w \) is not identically 0. For \( w \in \mathcal{W}(G) \), we let \( \|w\|_1 = \sum_{e \in E(G)} w(e) \) denote the total weight of \( G \) with respect to \( w \).

Let \( G \) and \( H \) be graphs in \( \mathcal{G} \). If both \( G \rightarrow H \) and \( H \rightarrow G \), then \( G \) and \( H \) are said to be homomorphically equivalent. This is denoted by \( G \equiv H \). Let \( n(G) \) and \( e(G) \) denote the number of vertices and edges in \( G \), respectively. Furthermore, let \( \omega(G) \) denote the clique number of \( G \); the greatest integer \( r \) such that \( K_r \rightarrow G \), and let \( \chi(G) \) denote the chromatic number of \( G \); the least integer \( n \) such that \( G \rightarrow K_n \). We now consider the following collection of problems, parameterised by a graph \( H \in \mathcal{G} \).

#### Problem 8.

The weighted maximum \( H \)-colourable subgraph problem, or \( \text{MAX} \; H\text{-COL} \), for short, is the maximisation problem with:

**Instance:** An edge-weighted graph \( (G, w) \), where \( G \in \mathcal{W} \) and \( w \in \mathcal{W}(G) \).

**Solution:** A subgraph \( G' \) of \( G \) such that \( G' \rightarrow H \).

**Measure:** The weight of \( G' \) with respect to \( w \).
Example 6.6. Let $G$ be a graph in $\mathcal{G}$. For $k \geq 2$, a $k$-cut in $G$ is the set of edges between $S_i$ and $S_j$, $i \neq j$, where $S_1, \ldots, S_k$ is a partition of $V(G)$. The $\text{MAX } k$-cut problem asks for the size of a largest $k$-cut. This problem is readily seen to be equivalent to finding a largest subgraph of $G$ which has a homomorphism to the complete graph $K_k$, i.e., finding a largest $k$-colourable subgraph of (a uniformly edge-weighted graph) $G$. Hence, for each $k \geq 2$, the problem $\text{MAX } k$-cut is included in the collection of $\text{MAX } H$-Col problems. It is well known that $\text{MAX } k$-cut is $\text{APX}$-complete, for $k \geq 2$ \cite{ACGK99}.

Given an edge-weighted graph $(G, w)$, denote by $mc_H(G, w)$ the measure of an optimal solution to the problem $\text{MAX } H$-Col. Denote by $mc_k(G, w)$ the (weighted) size of a largest $k$-cut in $(G, w)$. This notation is justified by the equivalence of the problems $\text{MAX } k$-cut and $\text{MAX } K_k$-Col. The decision version of $\text{MAX } H$-Col, the $H$-colouring problem, has been extensively studied (see the monograph by Hell and Nešetřil \cite{HN04c} and its many references). Hell and Nešetřil \cite{HN90} were the first to show that this problem is in $\text{P}$ if $H$ contains a loop or is bipartite, and $\text{NP}$-complete otherwise.

Assuming that $M \to N$, we consider the reduction $\phi_1, \gamma_1$ defined as follows. The function $\phi_1$ maps an instance $(G, w) \in I_N$ to $(G, w) \in I_M$ and the function $\gamma_1$ maps a solution $G' \in \text{Sol}_M(G, w)$ to the solution $G' \in \text{Sol}_N(G, w)$. Here, $m_M(\phi_1(I), f) = m_N(I, f)$, so the separation defined in \eqref{eq:6.2} takes the following simplified form.

$$s(M, N) = \inf_{G \in \mathcal{G}} \frac{mc_M(G, w)}{mc_N(G, w)} . \quad (6.3)$$

In Section \ref{sec:6.2} we show that this is indeed a continuous reduction. Proposition \ref{prop:6.4} therefore implies the following.

Lemma 6.7. Let $M \to N$ be two graphs in $\mathcal{G}$. If $mc_M$ can be approximated within $\alpha$, then $mc_N$ can be approximated within $\alpha \cdot s(M, N)$. If it is $\text{NP}$-hard to approximate $mc_N$ within $\beta$, then $mc_M$ is not approximable within $\beta/s(M, N)$, unless $\text{P} = \text{NP}$.

Example 6.8. Goemans and Williamson \cite{GW95} give an algorithm for $\text{MAX CUT}$ which is a $0.87856$-approximating algorithm for $\text{MAX } K_2$-Col. In Proposition \ref{prop:6.12} we will see that $s(K_2, C_{11}) = 10/11$. We can now apply Lemma 6.7 with $M = K_2$ and $N = C_{11}$, and we find that this $\text{MAX CUT}$-algorithm approximates $\text{MAX } C_{11}$-Col within $0.87856 \cdot s(K_2, C_{11}) \approx 0.79869$.

6.1.2 Chapter Outline

The basic properties of separation for the $\text{MAX } H$-Col family are worked out in Section \ref{sec:6.2}. The main result, Theorem \ref{thm:6.9}, provides a simplification of \eqref{eq:6.3} which we then use to obtain explicit values and bounds on separation. This simplified expression for $s(M, N)$ depends directly on the structure of the automorphism group and thereby the symmetries of $N$. In particular, this shows that the reduction $\phi_1, \gamma_1$ defined above is continuous. A linear programming formulation of separation is presented in Section \ref{sec:6.2.3}.
In Section 6.3, we use separation to study the approximability of MAX H-COL and investigate optimality issues, for several classes of graphs. Comparisons are made to the bounds achieved by the general MAX 2-CSP-algorithm by Håstad [Hås05]. Our investigation covers a spectrum of graphs, ranging from graphs with few edges and/or containing long shortest cycles to an asymptotic result, Theorem 6.31 for arbitrary graphs. We also look at graphs in the $G(n,p)$ random graph model, pioneered by Erdős and Rényi [ER60]. A discussion of open question and possible future work can be found in Section 6.4.

6.2 Basic Properties of the Separation Parameter

In this section we establish a basic theorem on separation, Theorem 6.9, and derive a number of results from it. We give exact values for the separation parameter in some cases and a general bound which shows that the reduction $\phi_1, \gamma_1$ is continuous. We also give a linear programming formulation for the separation parameter.

Let $M$ and $N \in \mathcal{G}$ be graphs and let $A = \text{Aut}^*(N)$ be the edge automorphism group of $N$, i.e. $\pi \in A$ acts on $E(N)$ by permuting the edges. The graph $N$ is called edge-transitive if $A$ acts transitively on $E(N)$. Let $\hat{W}(N)$ be the set of weight functions $w \in W(N)$ that satisfy $\|w\|_1 = 1$, and for which $w(e) = w(\pi \cdot e)$ for all $e \in E(N), \pi \in \text{Aut}^*(N)$. That is, $w$ is constant on the orbits of $\text{Aut}^*(N)$.

**Theorem 6.9.** Let $M, N \in \mathcal{G}$. Then,

$$s(M, N) = \inf_{w \in \hat{W}(N)} mc_M(N, w).$$

In particular, when $N$ is edge-transitive,

$$s(M, N) = mc_M(N, 1/e(N)).$$

We postpone the proof of Theorem 6.9 to Section 6.2.2.

6.2.1 Exact Values and Bounds

The Turán graph $T(n, r)$ is a graph formed by partitioning a set of $n$ vertices into $r$ subsets, with sizes as equal as possible, and connecting two vertices whenever they belong to different subsets. The properties of Turán graphs are given by the following theorem.

**Theorem 6.10 (Turán [Tur41]).** The following holds:

1. $e(T(n, r)) = \left\lfloor \left(1 - \frac{1}{r}\right) \cdot \frac{n^2}{2}\right\rfloor$;
2. $\omega(T(n, r)) = \chi(T(n, r)) = r$;
3. if $G$ is a graph such that $e(G) > e(T(n(G), r))$, then $\omega(G) > r$. 

100
6.2. Basic Properties of the Separation Parameter

Using Turán’s theorem, we can determine the separation exactly when the second parameter is a complete graph.

**Proposition 6.11.** Let \( H \) be a graph with \( \omega(H) = r \) and let \( n \) be an integer such that \( \chi(H) \leq n \). Then,

\[
s(H, K_n) = e(T(n, r))/e(K_n) = \left[ \left( 1 - \frac{1}{r} \right) \cdot \frac{n^2}{2} \right] / \binom{n}{2}.
\]

**Proof.** The graph \( K_n \) is edge-transitive. Therefore, by the second part of Theorem [6.9], it suffices to show that \( \omega(T(n, r)) = e(T(n, r))/e(K_n) \). By definition, \( T(n, r) \) is an \( r \)-partite subgraph of \( K_n \), so \( T(n, r) \to H \). Hence, \( \omega(T(n, r)) \geq e(T(n, r))/e(K_n) \). Conversely, any subgraph \( G \) of \( K_n \) such that \( G \to H \) must satisfy \( \omega(G) \leq \omega(H) = r \). Thus, by Theorem [6.10](3), \( e(G) \leq e(T(n, r)) \) which implies \( \omega(T(n, r))/e(K_n) \leq e(T(n, r))/e(K_n) \). \( \square \)

Next we consider separation between cycles. The even cycles are all bipartite and therefore homomorphically equivalent to \( K_2 \). The odd cycles, on the other hand, form a chain between \( K_2 \) and \( C_3 = K_3 \) in the following manner:

\[
K_2 \to \cdots \to C_{2i+1} \to C_{2i-1} \to \cdots \to C_3 = K_3.
\]

Note that the chain is infinite on the \( K_2 \)-side. The following proposition gives the separation between pairs of graphs in this chain. The value depends only on the target graph.

**Proposition 6.12.** Let \( m > k \) be positive integers. Then,

\[
s(K_2, C_{2k+1}) = s(C_{2m+1}, C_{2k+1}) = \frac{2k}{2k+1}.
\]

**Proof.** Since \( C_{2k+1} \) is edge-transitive, it suffices, by Theorem [6.2], to show that the maximum 2-cut and the maximum 2m + 1-cut of \( C_{2k+1} \) both contain 2k edges, i.e., that \( \omega(C_{2k+1}) = 2k = \omega(C_{2m+1}) \). Such cuts clearly exist since after removing one edge from \( C_{2k+1} \), the remaining subgraph is isomorphic to a path, and therefore homomorphic to \( K_2 \) (and to \( C_{2m+1} \)). Moreover, these cuts are the largest possible: \( C_{2k+1} \not\to K_2 \) and \( C_{2k+1} \not\to C_{2m+1} \). \( \square \)

Given two graphs \( M, H \in \mathcal{G} \), it may be difficult to determine \( s(M, H) \) directly. However, if we know that \( H \) is “homomorphically sandwiched” between \( M \) and another graph \( N \), so that \( M \to H \to N \), then it turns out that we can use \( s(M, N) \) as a lower bound for \( s(M, H) \). More generally, we have the following lemma.

**Lemma 6.13.** The following implications hold.

\[
H \to N \implies s(M, N) \leq s(M, H),
\]

\[
M \to K \implies s(M, N) \leq s(K, N).
\]
Proof. Assume that $G'$ is a subgraph of $G$ such that $G' \to H$, and $H \to N$. Then, $G' \to N$, so any solution $G' \subseteq G$ to $(G, w)$ as an instance of Max $H$-Col is also a solution to $(G, w)$ as an instance of Max $N$-Col of the same measure. It follows that $mc_H(G, w) \leq mc_N(G, w)$, so

$$s(M, H) = \inf_{G \in \mathcal{G}} \frac{mc_M(G, w)}{mc_H(G, w)} \geq \inf_{G \in \mathcal{G}} \frac{mc_M(G, w)}{mc_N(G, w)} = s(M, N).$$

The second part follows similarly.

We will see several applications of Lemma 6.13 in the following sections, but first we will use it to get a general lower bound on the separation parameter. Since this lower bound is strictly greater than 0, it follows from this result that the reduction $\phi_1, \gamma_1$ is continuous.

**Proposition 6.14.** Let $M \in \mathcal{G}$ be a fixed graph. Then, for any $N \in \mathcal{G}$,

$$1 \geq s(M, N) > \sum_{\{u,v\} \in E(M)} \frac{\deg(u) \deg(v)}{2e(M)^2}.$$

Proof. The upper bound follows from Theorem 6.9. For the lower bound, let $n = \chi(N)$ be the chromatic number of $N$, so that $N \to K_n$. Then, $s(M, N) \geq s(M, K_n) = mc_M(K_n, 1/e(K_n))$ by Lemma 6.13 and the second part of Theorem 6.9 respectively.

To give a lower bound on $mc_M(K_n, 1/e(K_n))$ we apply a standard probabilistic argument. Let $f : V(K_n) \to V(M)$ be a function, chosen randomly as follows: for every $v_k \in V(K_n)$, and $v_m \in V(M)$, the probability that $f(v_k) = v_m$ is equal to $\deg(v_m)/2e(M)$. Every possible function $f$ appears with non-zero probability and each function defines a subgraph of $K_n$ by including those edges that are mapped by $f$ to edges in $M$. We will show that there is at least one function that defines a subgraph $K \subseteq K_n$ with the right number of edges.

For $e \in E(K_n)$, and $e' = \{u,v\} \in E(M)$, let $Y_{e,e'} = 1$ if $f$ maps $e$ to $e'$, and $Y_{e,e'} = 0$ otherwise. Then, $E(Y_{e,e'}) = 2 \cdot \deg(u) \deg(v)/(2e(M))^2$. Let $X_e = 1$ if $f$ maps $e$ to some edge in $E(M)$, and $X_e = 0$ otherwise, so that $X_e = \sum_{e \in E(M)} Y_{e,e'}$. Then, the total number of edges in $K$ is equal to $\sum_{e \in E(K_n)} X_e$, and by linearity of expectation,

$$E(e(K)) = \sum_{e \in E(K_n)} E(X_e) = e(K_n) \sum_{\{u,v\} \in E(M)} \frac{\deg(u) \deg(v)}{2e(M)^2}.$$

Finally, we note that for an arbitrary fixed vertex $v_m \in V(M)$, the function defined by $f(v) = v_m$ for all $v \in V(K_n)$ defines the empty subgraph, and has a non-zero probability. Since $K_n$ and $M$ are non-empty we have $E(e(K)) > 0$, so there must exist at least one $f$ which defines a $K$ with strictly more than the expected total number of edges.

\[ \square \]
From Theorem 6.9, it follows that \( b \) and is a well-studied graph parameter \([\text{Alo}96; \text{BZ}03; \text{BL}86; \text{HS}82; \text{Loc}90]\).

103
The solutions to a \textsc{Max H-Col} instance have an alternative description, which is better suited for calculations: for any vertex map \( f : V(G) \to V(H) \), let \( f^\# : E(G) \to E(H) \) be the (partial) edge map induced by \( f \) (i.e. \( f^\# \) maps an edge \( \{u, v\} \) in \( E(G) \) to \( \{f(u), f(v)\} \) if the latter is in \( E(H) \), and otherwise, \( f^\#(\{u, v\}) \) is undefined). Each vertex map \( f \) then determines a subgraph \( G' = f^\#^{-1}(H) \subseteq G \), but two distinct functions, \( f \) and \( g \), do not necessarily determine distinct subgraphs. In this notation \( h : V(G) \to V(H) \) is a graph homomorphism precisely when \( (h^\#)^{-1}(E(H)) = E(G) \) or, alternatively, when \( h^\# \) is a total function. The set of solutions to an instance \((G, w)\) of \textsc{Max H-Col} can then be taken to be the set of vertex maps \( f : V(G) \to V(H) \) with the measure

\[
m_H(f) = \sum_{e \in (f^\#)^{-1}(E(H))} w(e). \tag{6.4}
\]

We will predominantly use this description of solutions.

Let \( f : V(G) \to V(H) \) be a solution to the instance \((G, w)\) of \textsc{Max H-Col}, and define \( w_f \in \mathcal{W}(H) \) as follows:

\[
w_f(e) = \sum_{e' \in (f^\#)^{-1}(e)} \frac{w(e')}{mc_H(G, w)}. \tag{6.5}
\]

Note that \( \|w_f\|_1 = 1 \) if and only if \( f \) is optimal. The following lemma and its corollary establishes half of Theorem 6.9.

**Lemma 6.17.** Let \( M, N \in \mathcal{G} \) be two graphs. Then, for every \( G \in \mathcal{G} \), every \( w \in \mathcal{W}(G) \), and any solution \( f \) to \((G, w)\) of \textsc{Max N-Col}, it holds that

\[
\frac{mc_M(G, w)}{mc_N(G, w)} \geq mc_M(N, w_f).
\]

**Proof.** Arbitrarily choose an optimal solution \( g : V(N) \to V(M) \) to the instance \((N, w_f)\) of \textsc{Max M-Col}. Then, \( g \circ f \) is a solution to \((G, w)\) as an instance of \textsc{Max M-Col} (see Figure 6.1). Note that we have \((g \circ f)^\# =
(f#)−1 ◦ (g#)−1. The measure of this solution is thus
\[ m_M(g ◦ f) = \sum_{e \in (g ◦ f)#^{-1}(E(M))} w(e) = \sum_{e' \in (g#)^{-1}(E(M))} \sum_{e \in (f#)^{-1}(e')} w(e) = \sum_{e' \in (g#)^{-1}(E(M))} w_f(e') \cdot mc_N(G, w) = mc_M(N, w_f) \cdot mc_N(G, w). \]

Clearly, the measure of g ◦ f is at most mc_M(G, w), so
\[ mc_M(G, w) \leq mc_M(N, w_f) \cdot mc_N(G, w). \]

The result follows after division by mc_N(G, w).

From Lemma 6.17, we have the following corollary, which shows that it is possible to eliminate G ∈ G from the infimum in the definition of s.

**Corollary 6.18.** Let M, N ∈ G be two graphs. Then,
\[ s(M, N) = \inf_{w \in \mathcal{W}(N)} mc_M(N, w). \]

**Proof.** First, we fix some optimal solution f = f(G, w) for each choice of (G, w). By taking infima over G and w on both sides of the inequality in Lemma 6.17, we then have
\[ s(M, N) \geq \inf_{G \in \mathcal{G}} \inf_{w \in \mathcal{W}(G)} mc_M(N, w_f) \geq \inf_{w \in \mathcal{W}(N)} mc_M(N, w), \]
where the second inequality holds since \( \|w_f\|_1 = 1 \) for any optimal f.

For the other direction, we specialise G to N, and restrict w to obtain:
\[ s(M, N) \leq \inf_{w \in \mathcal{W}(N)} mc_M(N, w) = \inf_{w \in \mathcal{W}(N)} mc_M(N, w). \]

This concludes the proof.

**Proof.** (of Theorem 6.9) From Corollary 6.18, we have that
\[ s(M, N) = \inf_{w \in \mathcal{W}(N)} mc_M(N, w) \leq \inf_{w \in \mathcal{W}(N)} mc_M(N, w). \]

To complete the first part of the theorem, it will be sufficient to prove that for any graph G ∈ G and w ∈ \( \mathcal{W}(G) \), there is a \( w' \in \mathcal{W}(N) \) such that the following inequality holds.
\[ \frac{mc_M(G, w)}{mc_N(G, w)} \geq mc_M(N, w'). \]

Taking infima on both sides of this inequality then shows that
\[ s(M, N) \geq \inf_{w' \in \mathcal{W}(N)} mc_M(N, w'). \]
Let \( A = \text{Aut}^*(N) \) be the automorphism group of \( N \) and let \( \pi \in A \) be an arbitrary automorphism. If \( f \) is an optimal solution to \((G, w)\) as an instance of \( \text{MAX N-COL} \), then so is \( \pi \circ f \). By Lemma 6.17, inequality (6.6) is satisfied by \( w_{\pi \circ f} \). Summing \( \pi \) in this inequality over \( A \) gives
\[
|A| \cdot \frac{mc_M(G, w)}{mc_N(G, w)} \geq \sum_{\pi \in A} mc_M(N, w_{\pi \circ f}) \geq mc_M(N, \sum_{\pi \in A} w_{\pi \circ f}),
\]
where the last inequality follows from Lemma 6.16. The weight function \( \sum_{\pi \in A} w_{\pi \circ f} \) is determined as follows:
\[
\sum_{\pi \in A} w_{\pi \circ f}(e) = \sum_{\pi \in A} \frac{\sum_{e' \in (f^\pi)^{-1}(\pi(e))} w(e')}{mc_N(G, w)} = \frac{|A|}{|Ae|} \cdot \frac{\sum_{e' \in (f^\pi)^{-1}(Ae)} w(e')}{mc_N(G, w)},
\]
where \( Ae \) denotes the orbit of \( e \) under \( A \). We have now shown that the inequality in (6.6) is satisfied by \( w' = \sum_{\pi \in A} w_{\pi \circ f} / |A| \), and that \( w' \) is in \( \hat{W}(N) \). The first part of the theorem follows.

For the second part, note that when the automorphism group \( A \) acts transitively on \( E(N) \), there is only one orbit \( Ae = E(N) \) for all \( e \in E(N) \). Then, the weight function \( w' \) is given by
\[
w'(e) = \frac{1}{e(N)} \cdot \frac{\sum_{e' \in (f^\pi)^{-1}(E(N))} w(e')}{mc_N(G, w)} = \frac{1}{e(N)} \cdot \frac{mc_N(G, w)}{mc_N(G, w)},
\]
since \( f \) is optimal. \( \square \)

6.2.3 A Linear Programming Formulation

In this section, we first derive a linear program for the separation parameter based on Corollary 6.18. Later we will see how to reduce the size of this program, but it serves as a good first exercise, and it will also be used for comparison with the linear program studied in Section 7.2.

Each vertex map \( f : V(N) \to V(M) \) induces an edge map \( f^\# \), which provides a lower bound on the separation:
\[
\sum_{e \in (f^\#)^{-1}(E(M))} w(e) \leq s(M, N). \tag{6.7}
\]

By Corollary 6.18, we want to find the least \( s \) such that for some weight function \( w \in \mathcal{W}(N), \|w\|_1 = 1 \), the inequalities (6.7) hold. Let the variables of the linear program be \( \{w_e\}_{e \in E(N)} \) and \( s \). We then have the following linear program for \( s(M, N) \).

Minimise \( s \)
subject to \( \sum_{e \in (f^\#)^{-1}(E(M))} w_e \leq s \) for each \( f : V(N) \to V(M) \),
\[
\sum_{e \in E(M)} w_e = 1,
\]
\( w_e, s \geq 0 \)
Given an optimal solution \( \{ \tilde{w}_e \}_{e \in E(N)} \), a weight function which minimises \( mc_M(N, w) \) is given by \( w(e) = w_e \) for \( e \in E(N) \). The measure of this solution is \( s = s(M, N) \). The program will clearly be very large with \( |E(N)| + 1 \) variables and \( |V(M)|^{|V(N)|} \) inequalities. Fortunately it can be improved upon.

From Theorem 6.9, it follows that in order to determine \( s(M, N) \), it is sufficient to minimise \( mc_M(N, w) \) over \( w \in \mathcal{W}(N) \). We can use this to describe a smaller linear program for computing \( s(M, N) \). Let \( A_1, \ldots, A_r \) be the orbits of \( \text{Aut}^*(N) \). The measure of a solution \( f \) when \( w \in \mathcal{W}(N) \) is equal to \( \sum_{i=1}^r f_i \cdot w_i \), where \( w_i \) is the weight of an edge in \( A_i \) and \( f_i \) is the number of edges in \( A_i \) which are mapped to an edge in \( M \) by \( f \). Note that given a \( w \), the measure of a solution \( f \) depends only on the vector \( (f_1, \ldots, f_r) \in \mathbb{N}^r \), which we call the signature of \( f \). Therefore, take the solution space to be the set of such vectors.

\[
F = \{ (f_1, \ldots, f_r) \mid f \text{ is a solution to } (N, w) \text{ of } \text{MAX } M\text{-COL} \}.
\]

Let the variables of the linear program be \( w_1, \ldots, w_r \) and \( s \), where \( w_i \) represents the weight of each element in the orbit \( A_i \) and \( s \) is an upper bound on the solutions.

Minimise \( s \)

subject to \( \sum_{i=1}^r f_i \cdot w_i \leq s \) \( \text{ for each } (f_1, \ldots, f_r) \in F \),

\[
\sum_{i=1}^r |A_i| \cdot w_i = 1,
\]

\( \sum_{i=1}^r w_i, s \geq 0 \)

Given an optimal solution \( w_\ast, s_\ast \) to this program, a weight function which minimises \( mc_M(N, w) \) is given by \( w(e) = w_i \) for \( e \in A_i \). The measure of this solution is \( s = s(M, N) \).

**Example 6.19.** The wheel graph on \( k \) vertices, \( W_k \), is a graph that contains a cycle of length \( k - 1 \) plus a vertex \( v \), which is not in the cycle, such that \( v \) is connected to every other vertex. We call the edges of the \( k - 1 \)-cycle outer edges and the remaining \( k - 1 \) edges spokes. It is easy to see that the clique number of \( W_k \) is equal to 4 when \( k = 4 \) (\( W_4 \) is isomorphic to \( K_4 \)) and that it is equal to 3 in all other cases. Furthermore, \( W_k \) is 3-colourable if and only if \( k \) is odd, and 4-colourable otherwise. This implies that for odd \( k \), the wheel graphs are homomorphically equivalent to \( K_3 \).

We will determine \( s(K_3, W_k) \) for even \( k \geq 6 \) using the previously described construction of a linear program. The first three wheel graphs for even \( k \) are shown in Figure 6.2. Note that the group action of \( \text{Aut}^*(W_k) \) on \( E(W_k) \) has two orbits, one which consists of all outer edges and one which consists of all the spokes. If we remove one outer edge or one spoke from \( W_k \), then the resulting graph can be mapped homomorphically to \( K_3 \). Therefore, it suffices to choose \( F = \{ f, g \} \) with \( f = (k - 1, k - 2) \) and \( g = (k - 2, k - 1) \).
Figure 6.2: The wheel graphs \(W_4, W_6, \text{ and } W_8\).

since all other solutions will have a smaller measure than at least one of these. The program for \(W_k\) looks as follows:

Minimise \(s\)
subject to \((k - 1) \cdot w_1 + (k - 2) \cdot w_2 \leq s\)
\((k - 2) \cdot w_1 + (k - 1) \cdot w_2 \leq s\)
\((k - 1) \cdot w_1 + (k - 1) \cdot w_2 = 1\)
\(w_1, w_2, s \geq 0\)

The optimal solution to this program is given by \(w_1 = w_2 = 1/(2k - 2)\), with \(s(K_3, W_k) = s = (2k - 3)/(2k - 2)\).

**Example 6.20.** In the previous example, the two weights in the optimal solution were equal. Here, we provide another example, where the weights turn out to be different for different orbits. The circular complete graph \(K_{8/3}\) has vertex set \(\{v_0, v_1, \ldots, v_7\}\), which is placed uniformly around a circle. There is an edge between any two vertices which are at a distance at least 3 from each other. Figure 6.3 depicts this graph.

![Figure 6.3: The graph \(K_{8/3}\).](image)

We will now calculate \(s(K_2, K_{8/3})\). Each vertex is at a distance 4 from exactly one other vertex, which means that there are 4 such edges. These edges, which are dashed in the figure, form one orbit under the action of
6.3. Approximation Bounds for MAX H-COL

The remaining 8 edges (solid) form a second orbit. Let $V(K_2) = \{u_0, u_1\}$. We can obtain a solution $f$ by mapping $f(v_i) = u_0$ if $i$ is even, and $f(v_i) = u_1$ if $i$ is odd. This solution will map all solid edges to $K_2$, but none of the dashed, hence $f = (0, 8)$. We obtain a second solution $g$ by mapping $g(v_i) = u_0$ for $0 \leq i < 4$, and $g(v_i) = u_1$ for $4 \leq i < 8$. This solution will map all but two of the solid edges in $K_{8/3}$ to $K_2$, hence $g = (4, 6)$. The inequalities given by $f$ and $g$ imply the inequalities given by any other solution, so we have the following program for $s(K_2, K_{8/3})$:

Minimise $s$
subject to $0 \cdot w_1 + 8 \cdot w_2 \leq s$
$4 \cdot w_1 + 6 \cdot w_2 \leq s$
$4 \cdot w_1 + 8 \cdot w_2 = 1$
$w_1, w_2, s \geq 0$

The optimal solution to this program is given by $w_1 = 1/20, w_2 = 1/10$, and $s(K_2, K_{8/3}) = s = 4/5$.

6.3 Approximation Bounds for MAX H-COL

In this section we apply the reduction $\phi_1, \gamma_1$ and use some of the explicit values obtained for $s$ in Section 6.2 to bound the approximation ratio of MAX H-COL for various families of graphs. First, we would like to remind the reader of some earlier results and also give a hint of what to expect when we start studying the approximability of MAX H-COL.

The probabilistic argument in Proposition 6.14 shows that MAX H-COL is in APX. Furthermore, Jonsson, Krokhin, and Kuivinen [JKK09] have shown that whenever $H$ is loop-free, MAX H-COL does not admit a PTAS, and otherwise the problem is trivial. Let us have a closer look at a concrete, well-known example: the MAX CUT problem. As mentioned in Chapter 1, this problem was one of Karp’s original 21 NP-complete problems [Kar72] and has a trivial approximation ratio of 1/2, which is obtained by assigning each vertex randomly to either part of the partition. We also saw that the trivial randomized algorithm is easy to derandomize; [SG76] gave the first such approximation algorithm. The 1/2 ratio was in fact essentially the best known ratio for MAX CUT until 1995, when Goemans and Williamson [GW95], using semidefinite programming (SDP), achieved the ratio 0.87856 mentioned in Example 6.8. Frieze and Jerrum [FJ97] determined lower bounds on the approximation ratios for MAX $k$-CUT using similar SDP techniques. Sharpened results for small values of $k$ have later been obtained by Klerk, Pasechnik, and Warners [KPW04]. Håstad [Has05] has shown that SDP is a universal tool for solving the general MAX 2-CSP problem (where every constraint only involves two variables) over any (finite) domain, in the sense that it establishes non-trivial approximation results for all of those problems. Until recently no other method than SDP was known to yield a non-trivial approximation ratio for MAX CUT. Trevisan [Tre09] broke this barrier by using techniques.
from algebraic graph theory to reach an approximation guarantee of 0.531. [Sot09] later improved this bound to 0.6142 by a more refined analysis.

Khot [Kho02] suggested the unique games conjecture as a possible direction for proving inapproximability properties of some important optimisation problems. The conjecture states the following (equivalent form from [KKMO07]):

**Conjecture 6.21.** Given any $\delta > 0$, there is a prime $p$ such that given a set of linear equations $x_i - x_j = c_{ij} \pmod{p}$, it is $\text{NP}$-hard to decide which one of the following is true:

- There is an assignment to the $x_i$'s which satisfies at least a $1 - \delta$ fraction of the constraints.
- All assignments to the $x_i$'s can satisfy at most a $\delta$ fraction of the constraints.

Under the assumption that the UGC holds, Khot, Kindler, Mossel, and O’Donnel [KKMO07] proved the approximation ratio achieved by the Goemans and Williamson algorithm for $\text{MAX CUT}$ to be essentially optimal and also provided upper bounds on the approximation ratio for $\text{MAX k-CUT}$, $k > 2$. The proof for the $\text{MAX CUT}$ case crucially relies on Gaussian Analysis. In particular, it uses Borell’s Theorem to answer the question of partitioning $\mathbb{R}^n$ into two sets of equal Gaussian measure so as to minimize the Gaussian noise-sensitivity, thereby transferring a Fourier analytic question to a geometric one.

Recently, Isaksson and Mossel [IM09] showed that a similar geometric conjecture have further implications for the approximability of $\text{MAX k-CUT}$.

**Conjecture 6.22.** The standard $k$-simplex partition is the most noise-stable balanced partition of $\mathbb{R}^n$ with $n \geq k - 1$.

A partition of $\mathbb{R}^n$ into $k$ measurable sets $A_1, \ldots, A_k$ is called balanced if each $A_i$ has Gaussian measure $1/k$. The $\epsilon$-noise sensitivity is defined as the probability that two $(1 - 2\epsilon)$-correlated $n$-dimensional standard Gaussian points $x, y \in \mathbb{R}^n$ belong to different sets in the partition. The standard $k$-simplex partition of $\mathbb{R}^n$ is obtained by letting $\mathbb{R}^n = \mathbb{R}^{k-1} \times \mathbb{R}^{n-k+1}$ and then partitioning $\mathbb{R}^{k-1}$ into $k$ regular simplicial cones. Assuming this standard simplex conjecture (SSC) and the UGC, Isaksson and Mossel show that the Frieze and Jerrum SDP relaxation obtains the optimal approximation ratio for $\text{MAX k-CUT}$.

### 6.3.1 A General Reduction

Our main tool will be a generalisation of the reduction introduced in Section 6.1.1. Let $M$ and $N$ be (arbitrary) undirected graphs and consider the following reduction, $\phi_2$, $\gamma_2$, from $\text{MAX N-COL}$ to $\text{MAX M-COL}$: The function $\phi_2$ maps an instance $(G, w) \in I_N$ to $(G, w) \in I_M$. Let $f : V(G) \to V(M)$
be a solution to \((G, w)\). Let \(g : V(M) \to V(N)\) be an optimal solution in \(\text{Sol}_N(M, w_f)\) (see Equation 6.5 for the definition of \(w_f\)). The function \(\gamma_2\) maps \(f\) to \(g \circ f\).

**Proposition 6.23.** The reduction \(\phi_2, \gamma_2\) from Max M-Col to Max N-Col is continuous with constant \(s(M, N) \cdot s(N, M)\).

**Proof.** First, we argue that \(\gamma_2\) can be computed in polynomial time. We must show that \(g\) can be found in polynomial time. An optimal solution to \((M, w_f)\) can be obtained by brute force. This takes \(|V(M)|^{|V(N)|}\) times the time to evaluate a candidate solution \(g'\). The measure of a solution depends on \(w_f\), and thereby on \(f\), but for a given candidate \(g'\), it can clearly be obtained in polynomial time. Since \(M\) and \(N\) are fixed, the total time is polynomial in the size of \((G, w)\).

Next, we show that \(\phi_2, \gamma_2\) is continuous with constant \(s(M, N) \cdot s(N, M)\).

\[
m_N(g \circ f) = m_M(f) \cdot mc_N(M, w_f) \\
\geq m_M(f) \cdot \inf_{w \in W(M)} mc_N(M, w) \\
= m_M(f) \cdot s(N, M),
\]

where the final equality follows from Corollary 6.18. From the definition of \(s\) (see Equation 6.3) we have the inequality \(mc_N(G, w) \leq mc_M(G, w) / s(M, N)\) for all \(G \in \mathcal{G}\) and \(w \in W(G)\). Consequently, with \(I = (G, w)\),

\[
R_N(I, \gamma_2(I, f)) = \frac{m_N(g \circ f)}{mc_N(G, w)} \\
\geq s(M, N) \cdot s(N, M) \cdot \frac{m_M(f)}{mc_M(G, w)} \\
= s(M, N) \cdot s(N, M) \cdot R_M(\phi_2(I), f).
\]

Since \(s(M, N), s(N, M) > 0\), it follows that the reduction is continuous. \(\square\)

As a direct consequence (using Proposition 6.4), we get the following generalisation of Lemma 6.7

**Lemma 6.24.** Let \(M, N \in \mathcal{G}\). If \(mc_M\) can be approximated within \(a\), then \(mc_N\) can be approximated within \(a \cdot s(M, N) \cdot s(N, M)\). If it is \(\text{NP}\)-hard to approximate \(mc_N\) within \(\beta\), then \(mc_M\) is not approximable within \(\beta / (s(M, N) \cdot s(N, M))\), unless \(P = \text{NP}\).

The parameter \(s(M, N)\) enjoys many properties which make it act as some kind of distance between the graphs \(M\) and \(N\). Lemma 6.13 stands out as a particularly clear example. This interpretation is also present in our main application of the parameter, namely Lemma 6.23. As was noted in the introduction, this result essentially says that graphs which are “close” in \(s\), in the sense of having an \(s\)-value close to 1, also have closely related approximation ratios. In fact, it is possible to show that \(1 - s(M, N) \cdot s(N, M)\) is a metric on the space of graphs taken modulo homomorphically equivalent.
6. Separation and Approximability

The homomorphism relation $\rightarrow$ defines a quasi-order, but not a partial order on the set $\mathcal{G}$. The failing axiom is that of antisymmetry, since $G \equiv H$ does not necessarily imply $G = H$. To remedy this, let $\mathcal{G}_\equiv$ denote the set of equivalence classes of $\mathcal{G}$ under $\equiv$. The relation $\rightarrow$ is defined on $\mathcal{G}_\equiv$ in the natural way and $(\mathcal{G}_\equiv, \rightarrow)$ is a lattice denoted by $\mathcal{C}_\mathcal{G}$. For a more in-depth treatment of this lattice, see Hell and Nešetřil [HN04b]. Here we endow $\mathcal{G}_\equiv$ with a metric $\bar{d}$ defined in the following way: for $M, N \in \mathcal{G}_\equiv$,

$$d(M, N) = 1 - s(M, N) \cdot s(N, M)$$

Proposition 6.15 shows that $\bar{d}$ is well-defined as a function on the set $\mathcal{G}_\equiv$. We now show that $\bar{d}$ is indeed a metric on this space.

**Lemma 6.25.** The pair $(\mathcal{G}_\equiv, \bar{d})$ forms a metric space.

**Proof.** Positivity and symmetry follows immediately from the definition and the fact that $0 \leq s(M, N) \leq 1$ for all $M$ and $N$. Since $s(M, N) = 1$ if and only if $N \rightarrow M$, it also holds that $d(M, N) = 0$ if and only if $M$ and $N$ are homomorphically equivalent. That is, $d(M, N) = 0$ if and only if $M$ and $N$ represent the same member of $\mathcal{G}_\equiv$. Furthermore, for graphs $M, N$ and $K \in \mathcal{G}$:

$$s(M, N) \cdot s(N, K) = \inf_{\omega \in W(\mathcal{G})} \frac{mc_M(G, \omega)}{mc_N(G, \omega)} \cdot \inf_{\omega \in W(\mathcal{G})} \frac{mc_N(G, \omega)}{mc_K(G, \omega)} \leq \inf_{\omega \in W(\mathcal{G})} \left( \frac{mc_M(G, \omega)}{mc_N(G, \omega)} \cdot \frac{mc_N(G, \omega)}{mc_K(G, \omega)} \right) = s(M, K).$$

Therefore, with $a = s(M, N) \cdot s(N, M), b = s(N, K) \cdot s(K, N)$ and $c = s(M, K) \cdot s(K, M) \geq a \cdot b$,

$$d(M, N) + d(N, K) - d(M, K) = 1 - a + 1 - b - (1 - c) \geq 1 - a - b + a \cdot b = (1 - a) \cdot (1 - b) \geq 0,$$

which shows that $d$ satisfies a triangle inequality.

In what follows, our main algorithmic tools will be the following two theorems.

**Theorem 6.26** (Goemans and Williamson [GW95]). MAX CUT can be approximated within

$$a_{GW} = \min_{0 < \theta < \pi} \frac{\theta / \pi}{(1 - \cos \theta) / 2} \approx 0.87856.$$

A few logarithms will appear in the upcoming expressions. We fix the notation $\ln x$ for the natural logarithm of $x$, and $\log y$ for the base-2 logarithm of $y$. 

112
6.3. Approximation Bounds for Max $H$-Col

**Theorem 6.27** (Frieze and Jerrum [FJ97]). Max $k$-cut can be approximated within

$$\alpha_k = 1 - \frac{1}{k} + \frac{2 \ln k}{k^2} (1 + o(1)).$$

We note that Klerk, Pasechnik, and Warners [KPW04] have presented the sharpest known bounds on $\alpha_k$ for small values of $k$. Table 6.1 lists $\alpha_{GW}$ together with the first of these lower bounds.

<table>
<thead>
<tr>
<th>$k$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_k \geq$</td>
<td>0.87856</td>
<td>0.836008</td>
<td>0.857487</td>
<td>0.876610</td>
<td>0.891543</td>
</tr>
</tbody>
</table>

Table 6.1: The best known lower bounds on $\alpha_k$ for $k = 2, \ldots, 6$.

Håstad [Hås05] has shown the following.

**Theorem 6.28** (Håstad [Hås05]). There is an absolute constant $c > 0$ such that $mc_H$ can be approximated within

$$1 - \frac{t(H)}{n^2} \cdot \left(1 - \frac{c}{n^2 \log n}\right),$$

where $n = n(H)$ and $t(H) = n^2 - 2e(H)$.

We will compare the performance of Håstad’s algorithm on Max $H$-Col with the performance of the algorithms in Theorems 6.26 and 6.27 when analysed using the reduction $\phi_2, \gamma_2$ and estimates, or exact values, of the separation parameter. For this purpose, we introduce two functions, $FJ_k$ and $Hå$, such that, if $H$ is a graph, then $FJ_k(H)$ denotes the best bound on the approximation guarantee when Frieze and Jerrum’s algorithm for Max $k$-cut is applied to the problem $mc_H$, while $Hå(H)$ is the guarantee when Håstad’s algorithm is used to approximate $mc_H$. This comparison is not entirely fair since Håstad’s algorithm was not designed with the goal of providing optimal results; the goal was to beat random assignments. However, it is currently the best known algorithm for approximating Max $H$-Col, for arbitrary $H \in G$, which also provides an easily computable bound on the guaranteed approximation ratio; this is in contrast with the conjectured optimal algorithms of Raghavendra [Rag08] (see the discussion in Section 7.3).

Near-optimality of our approximation method will be investigated using results depending on Khot’s unique games conjecture (Conjecture 6.21). Thus, we will henceforth assume that the UGC is true, which implies the following inapproximability results.

**Theorem 6.29** (Khot, Kindler, Mossel, and O’Donnel [KKMO07]). The following holds modulo the truth of the unique games conjecture:

- For every $\epsilon > 0$, it is NP-hard to approximate $mc_2$ within $\alpha_{GW} + \epsilon$. 

113
6. Separation and Approximability

- It is \textbf{NP}-hard to approximate \( m_c^k \) within

\[
1 - \frac{1}{k} + \frac{2 \ln k}{k^2} + O\left(\frac{\ln \ln k}{k^2}\right).
\]

Furthermore, under the assumption that the standard simplex conjecture (Conjecture 6.22) is true, we have the following strengthening of Theorem 6.29.

**Theorem 6.30** (Isaksson and Mossel [IM09]). The following holds modulo the truth of the UGC and SSC:

- For every \( \epsilon > 0 \), it is \textbf{NP}-hard to approximate \( m_c^k \) within \( \alpha_k + \epsilon \).

6.3.2 Asymptotic Performance

We start by deriving a general, asymptotic, result on the performance of our method.

**Theorem 6.31.** Let \( H \in \mathcal{G} \) be a graph with \( \omega(H) = r \) and \( \chi(H) = k \). Then,

\[
F_{jk}(H) = 1 - \frac{1}{r} + \frac{2 \ln r}{r^2} \left(1 - \frac{1}{r} + o(1)\right),
\]

where \( o(1) \) is with respect to \( k \).

Furthermore, \( m_c^H \) cannot be approximated within

\[
1 - \frac{1}{k} + \frac{2 \ln r}{r^2} (1 + o(1)), \quad \text{where} \quad o(1) \quad \text{is with respect to} \quad r.
\]

**Proof.** By Proposition 6.11 we have

\[
s(H, K_k) = \left(1 - \frac{1}{r}\right) \cdot \frac{k^2}{2} + O(1) / \binom{k}{2}
\]

\[
= \left(1 - \frac{1}{r}\right) \left(1 + \frac{1}{k-1}\right) + O(k^{-2}).
\]

By Lemma 6.24 and Theorem 6.27 we then have

\[
F_{jk}(H) = s(H, K_k) \cdot \alpha_k =
\]

\[
= \left(1 - \frac{1}{r}\right) \left(1 + \frac{1}{k-1}\right) + O(k^{-2}) \left(1 - \frac{1}{k} + \frac{2 \ln k}{k^2} (1 + o(1))\right)
\]

\[
= \left(1 - \frac{1}{r}\right) \left(1 - \frac{1}{k} + \frac{1}{k-1} - \frac{1}{k(k-1)} + \frac{2 \ln k}{k^2} (1 + o(1))\right),
\]

and the first inequality follows, since \(- \frac{1}{k} + \frac{1}{k-1} - \frac{1}{k(k-1)} = 0\).
6.3. Approximation Bounds for \( \text{MAX} \ H\text{-COL} \)

For the second part, we have \( s(K_r, H) \geq s(K_r, K_k) = s(H, K_k) \) by Lemma 6.13 and Proposition 6.11 so

\[
1/s(K_r, H) \leq \frac{k^2}{2} \left( 1 - \frac{1}{r} \right) \left( \frac{k^2}{2} + O(1) \right) 
= \left( 1 + \frac{1}{r-1} \right) \left( 1 - \frac{1}{k} \right) + O(k^{-2}).
\]

Note the similarity between this upper bound and the expression for \( s(H, K_k) \). In fact, without losing any precision in the following calculations, we could replace the \( O(k^{-2}) \)-term by \( O(r^{-2}) \). By Lemma 6.24, \( m_{CH} \) cannot be approximated within \( a_r/s(K_r, H) \). An upper bound for \( a_r/s(K_r, H) \) can now be calculated as in the first part with \( r \) and \( k \) swapped. In the final expression we drop \(-\frac{1}{k}\) from the last parenthesis since this is absorbed by \( o(1) \).

To give an upper bound on the performance of Håstad’s algorithm, we can proceed as follows: Let \( n = n(H) \) and \( r = \omega(H) \), \( k = \chi(H) \) as in the proposition. By Theorem 6.10, \( e(H) \leq \left( 1 - \frac{1}{r} \right) \cdot \frac{n^2}{2} \), hence \( \frac{2e(H)}{n^2} \leq 1 - \frac{1}{r} \), and

\[
H\hat{a}(H) = 1 - \left( 1 - \frac{2e(H)}{n^2} \right) \left( 1 - \frac{c}{n^2 \log n} \right) 
= \frac{2e(H)}{n^2} \left( 1 - \frac{c}{n^2 \log n} \right) + \frac{c}{n^2 \log n} 
\leq 1 - \frac{1}{r} + \frac{c}{k^2 \log k}.
\]

We see that our algorithm performs asymptotically better.

6.3.3 Some Specific Graph Classes

Next, we investigate the performance of our method on sequences of graphs “tending to” \( K_2, K_3 \) in the sense that the separation of \( K_2 \) (\( K_3 \)) and a graph \( H_k \) from the sequence tends to 1 as \( k \) tends to infinity. In several cases, the girth of the graphs plays a central role. The girth of a graph \( G \) is the length of a shortest cycle in \( G \). The odd girth of \( G \) is the length of a shortest odd cycle in \( G \). Hence, if \( G \) has odd girth \( g \), then \( C_g \to G \), but \( C_{2k+1} \not\to G \) for \( 3 \leq 2k+1 < g \).

Proposition 6.32. We have the following bounds.

1. Let \( k \geq 1 \). Then, \( F_{L}^2(C_{2k+1}) \geq \frac{2k+1}{2k+2} \cdot \alpha_{GW} \) and \( m_{C_{2k+1}} \) cannot be approximated within \( \frac{2k+1}{2k+2} \cdot \alpha_{GW} + \epsilon \), for any \( \epsilon > 0 \).

2. Let \( m > k \geq 4 \) and let \( H \) be a graph with odd girth \( g \geq 2k+1 \) and minimum degree \( \delta(H) \geq \frac{2m-1}{2k+1} \). Then, \( F_{L}^2(H) \geq \frac{2k+1}{2k+2} \cdot \alpha_{GW} \) and \( m_{H} \) cannot be approximated within \( \frac{2k+1}{2k+2} \cdot \alpha_{GW} + \epsilon \), for any \( \epsilon > 0 \).

115
3. Let \( H \) be a planar graph with girth at least \( g = \frac{20}{3} \). Then, \( FJ_2(H) \geq \frac{2k}{2k+1} \cdot \alpha_{\text{GW}} \) and \( mc_H \) cannot be approximated within \( \frac{2k+1}{2k} \cdot \alpha_{\text{GW}} + \epsilon \), for any \( \epsilon > 0 \).

4. Let \( k \geq 6 \) be even. Then, \( FJ_3(W_k) \geq \frac{2k-3}{2k+1} \cdot \alpha_3 \) and \( mc_{W_k} \) cannot be approximated within \( \frac{2k-3}{2k+1} \cdot \alpha_3 + \epsilon \), for any \( \epsilon > 0 \).

Proof. (1) From Lemma 6.12, we see that \( s(K_2, C_{2k+1}) = \frac{2k}{2k+1} \) which implies (using Lemma 6.24) that \( FJ_2(C_k) \geq \frac{2k}{2k+1} \cdot \alpha_{\text{GW}} \). Furthermore, \( mc_C \) cannot be approximated within \( \alpha_{\text{GW}} + \epsilon' \) for any \( \epsilon' > 0 \). From the second part of Lemma 6.24, we get that \( mc_{C_{2k+1}} \) is not approximable within \( \frac{2k+1}{2k} \cdot (\alpha_{\text{GW}} + \epsilon') \) for any \( \epsilon' \). With \( \epsilon' = \epsilon \), the result follows.

(2) Lai and Liu [LL00] show that if \( H \) is a graph with the stated properties, then there exists a homomorphism from \( H \) to \( C_{2k+1} \). Thus, \( K_2 \rightarrow H \rightarrow C_{2k+1} \) which implies that \( s(K_2, H) \geq s(K_2, C_{2k+1}) = \frac{2k}{2k+1} \). By Lemma 6.24, \( FJ_2(H) \geq \frac{2k}{2k+1} \cdot \alpha_{\text{GW}} \), but \( mc_H \) cannot be approximated within \( \frac{2k+1}{2k} \cdot \alpha_{\text{GW}} + \epsilon \) for any \( \epsilon > 0 \).

(3) Borodin, Kim, Kostochka, and West [BKKW04] show that there exists a homomorphism from \( H \) to \( C_{2k+1} \). The result follows as for case (2).

(4) We know from Example 6.19 that \( K_3 \rightarrow W_k \) and \( s(K_3, W_k) = \frac{2k-3}{2k-2} \). The result again follows by Lemma 6.24.

We can compare the results of Proposition 6.32 to the performance of Håstad’s algorithm as follows: Let \( n = n(H) \). In (1), we have \( c(H) = n \); for (2), Dutton and Brigham [DB91] have given an upper bound on \( c(H) \) of asymptotic order \( n^{1+2/(g-1)} \); in (3), \( e(H) \leq 3n - 6 \), since \( H \) is planar; and finally in (4), we have \( e(H) = 2(n - 1) \). Now note that by ignoring lower-order terms in the expression for \( H(H) \) in Theorem 5.28, we get \( H(H) = \frac{2(2n)}{n^2} (1 + o(1)) \). Hence, for each case (1)–(4), \( H(H) \to 0 \) as \( n \to \infty \).

Proposition 6.32(3) can be strengthened and extended in several ways: For \( K_4 \)-minor-free graphs, Pan and Zhu [PZ02] have given odd girth-restrictions for \( 2k+1 \)-colourability which is better than the girth-restriction in Proposition 6.32(3). Dvořák, Škreklovský, and Valla [DSV08] have proved that every planar graph \( H \) of odd girth at least 9 is homomorphic to the Petersen graph, \( P \). The Petersen graph is edge-transitive and its bipartite density is known to be \( 4/5 \) (cf. [PZ03]). In other words, \( s(K_2, P) = 4/5 \). Consequently, \( mc_P \) can be approximated within \( \frac{5}{4} \cdot \alpha_{\text{GW}} \) but not within \( \frac{5}{4} \cdot \alpha_{\text{GW}} + \epsilon \) for any \( \epsilon > 0 \). This is an improvement on the bounds in Proposition 6.32(3) for planar graphs with girth strictly less than 13. We can also consider graphs embeddable on higher-genus surfaces. For instance, Proposition 6.32(3) is true for graphs embeddable on the projective plane, and it is also true for graphs of girth strictly greater than \( \frac{20k}{3} \) whenever the graphs are embeddable on the torus or Klein bottle. These bounds are direct consequences of results by Borodin, Kim, Kostochka, and West [BKKW04].
6.3. Approximation Bounds for Max H-Col

6.3.4 Random Graphs

Finally, we look at random graphs. Let \( G(n, p) \) denote the random graph on \( n \) vertices in which every edge is chosen uniformly at random, and independently, with probability \( p = p(n) \). We say that \( G(n, p) \) has a property \( A \) asymptotically almost surely (a.a.s.) if the probability that it satisfies \( A \) tends to 1 as \( n \) tends to infinity. Here, we let \( 0 < p < 1 \) be a fixed constant.

**Proposition 6.33.** Let \( H \in G(n, p) \). Then, a.a.s.,

\[
F_J(H) = 1 - \frac{\ln(1/p)}{2\ln n} (1 + o(1)).
\]

**Proof.** For \( H \in G(n, p) \) it is well known that a.a.s. \( \omega(H) \) assumes one of at most two values around \( \frac{2\ln n}{\ln(1/p)} + \Theta(\ln \ln n) \) \cite{Mat72, BE76}. Let \( r = \omega(H) \). By Theorem 6.31,

\[
F_J(H) = 1 - \frac{1}{r} + \frac{2\ln k}{k^2} \left( 1 - \frac{1}{r} + o(1) \right) = 1 - \frac{\ln(1/p)}{2\ln n} (1 + o(1)).
\]

\( \square \)

For a comparison to Håstad’s algorithm, note that \( e(H) = \frac{n^2p}{2}(1 + o(1)) \) a.a.s. for \( H \in G(n, p) \), so

\[
H\dot{a}(H) = \frac{2e(H)}{n^2} (1 + o(1)) = p + o(1).
\]

The slow logarithmic growth of the clique number of \( G(n, p) \) works against our method in this case. However, we still manage to achieve an approximation ratio tending to 1 unlike Håstad’s algorithm which ultimately is restricted by the density of the edges.

We conclude this section by looking at what happens for graphs \( H \in G(n, p) \) when \( p \) is no longer chosen to be a constant, but we instead let \( np \) tend to a constant \( \epsilon < 1 \) as \( n \to \infty \). The following theorem allows us to do this.

**Theorem 6.34 (Erdős and Rényi \cite{ER60}).** Let \( c \) be a positive constant and \( p = \frac{\epsilon}{n} \). If \( c < 1 \), then a.a.s. no component in \( G(n, p) \) contains more than one cycle, and no component has more than \( \frac{\ln n}{c} - \ln \ln c \) vertices.

Now we see that if \( np \to \epsilon \) when \( n \to \infty \), then \( G(n, p) \) almost surely consists of components with at most one cycle. Thus, each component resembles a cycle where, possibly, trees are attached to certain vertices of the cycle, and each component is homomorphically equivalent to the cycle it contains. Proposition 6.12 is therefore applicable in this part of the \( G(n, p) \) spectrum.
6.4 Discussion and Open Problems

Our initial idea of separation as applied to the MAX $H$-COL problem lead us to a binary graph parameter that, in a sense, measures how close one graph is to be homomorphic to another. While not apparent from the original definition in (6.3), which involves taking an infimum over all possible instances, we have shown that the parameter can be computed effectively by means of linear programming. Given a graph $H$ and known approximability properties for MAX $H$-COL, this parameter allows us to deduce bounds on the corresponding properties for MAX $H'$-COL for graphs $H'$ that are “close” to $H$. Our approach can be characterised as local; the closer the separation of two graphs is to 1, the more precise are our bounds. We have succeeded in showing that given a “base set” containing the complete graphs, our method can be used to derive good bounds on the approximability of MAX $H$-COL for any graph $H$.

For the applications in Section 6.3, we have used the complete graphs as our base set of known problems. We have shown that this set of graphs is sufficient for achieving new, non-trivial bounds on several different classes of graphs. That is, when we apply Frieze and Jerrum’s algorithm \[FJ97\] to MAX $H$-COL, we obtain results comparable to, or better than those guaranteed by Håstad’s MAX 2-CSP algorithm \[Has05\], for the classes of graphs we have considered. This comparison should however be taken with a grain of salt. The analysis of Håstad’s MAX 2-CSP algorithm only aims to prove it better than a random assignment, and may leave room for strengthening of the approximation guarantee. At the same time, we are overestimating the distance for most of the graphs under consideration. It is likely that both results can be improved, within their respective frameworks. When considering inapproximability, we have relied on the unique games conjecture. Weaker inapproximability results, independent of the UGC, exist for both MAX CUT \[Has01\] and MAX $k$-CUT \[KKLP97\], and they are applicable in our setting. We emphasise that our method is not per se dependent on the truth of the UGC.

We conclude this discussion by considering some possible extensions of our approximability results. We have already noted that MAX $H$-COL is a special case of the MAX CSP($\Gamma$) problem, parameterised by a finite constraint language $\Gamma$. It should be relatively clear that we can define a generalised separation parameter on a pair of general constraint languages. This would constitute a novel method for studying the approximability of MAX CSP—a method that may cast some new light on the performance of Raghavendra’s algorithm. As a way of circumventing the hardness result by Khot, Kindler, Mossel, and O’Donnel \[KKMO07\], Kaporis, Kiouris, and Stavropoulos \[KKS06\] show that $mc_2$ is approximable within 0.952 for any given average degree $d$, and asymptotically almost all random graphs $G$ in $G(n, m = \lfloor \frac{d}{2}n \rfloor)$. Here, $G(n, m)$ is the probability space of random graphs on $n$ vertices and $m$ edges, selected uniformly at random. A different ap-
proach is taken by Coja-Oghlan, Moore, and Sanwalani [COMS05] who give an algorithm that approximates $m_c$ within $1 - O(1/\sqrt{np})$ in expected polynomial time, for graphs from $G(n, p)$. Kim and Williams [KW11] give an algorithm for finding a cut with value at least an additive constant $k$ better than $\alpha_{GW}$ times the value of an optimal cut (provided such a cut exists) in a given graph, if you are willing to spend time exponential in $k$ to do so. In a similar vein, [CJM11] show how to use time exponential in $k$ to find a cut better than the Edwards-Erdős bound, i.e., with value $e(G)/2 + (n(G) - 1)/4 + k$ in a given graph $G$ or decide that no such cut exists. It would be interesting to study whether separation can be used to extend these results to improved approximability bounds on MAX $H$-COL.
In order to apply our separation method successfully to the \textsc{Max H-Col} problem, but also to get a better understanding of the separation parameter, we want to compute some explicit values of $s(M, N)$ for various graphs $M$ and $N$. To this end, we turn to the \textit{circular complete graphs} in Section 7.1. We take a close look at 3-colourable circular complete graphs, and amongst other things, find that there are regions of such graphs on which separation is constant. The application of these results to \textsc{Max H-Col} relies heavily on existing graph homomorphism results, and in this context we will see that a conjecture by Jaeger \cite{jaeger1993} has precise and interesting implications (see Section 7.3).

Another way to study separation is to relate it to known graph parameters. In Section 7.2 we show that our parameter is closely related to a fractional edge-covering problem and an associated “chromatic number”, and that we can pass effortlessly between the two views, gaining insights into both. This part is highly inspired by work of Šámal \cite{salam2005, salam2006, salam2012} on \textit{cubical colourings and fractional covering by cuts}. In particular, our connection to Šámal’s work brings about a new family of chromatic numbers that provides us with an alternative way of computing our parameter. We also use our knowledge of the behaviour of separation to decide two conjectures concerning cubical colourings.

Finally, we summarise the future prospects and open problems of the method in Section 7.3.
7. Separation as a Graph Parameter

7.1 Circular Complete Graphs

The successful application of our method relies on the ability to compute $s(M, N)$ for various graphs $M$ and $N$. In Section 6.2.3, we saw how this can be accomplished by the means of linear programming. This insight is put to use here in the context of circular complete graphs. We have already come across examples of such graphs in the form of (ordinary) complete graphs, cycles, and the graph $K_{8/3}$ in Figure 5.3. We will now take a closer look at them.

Definition 7.1. Let $p$ and $q$ be positive integers such that $p \geq 2q$. The circular complete graph, $K_{p/q}$, has vertex set $\{v_0, v_1, \ldots, v_{p-1}\}$ and edge set $\{\{v_i, v_j\} \mid q \leq |i - j| \leq p - q\}$.

The image to keep in mind is that of the vertices placed uniformly around a circle with an edge connecting two vertices if they are at a distance at least $q$ from each other.

Example 7.2. Some well-known graphs are extreme cases of circular complete graphs:

- The complete graph $K_n, n \geq 2$ is a circular complete graph with $p = n$ and $q = 1$.
- The cycle graph $C_{2k+1}, k \geq 1$ is a circular complete graph with $p = 2k + 1$ and $q = k$.

These are the only examples of edge-transitive circular complete graphs.

A fundamental property of the circular complete graphs is given by the following theorem.

Theorem 7.3 (see Hell and Nešetřil [HN04c]). For positive integers $p, q, p',$ and $q'$,

$$K_{p/q} \to K_{p'/q'} \iff \frac{p}{q} \leq \frac{p'}{q'}$$

Due to this theorem, we may assume that whenever we write $K_{p/q}$, the positive integers $p$ and $q$ are relatively prime.

One of the main reasons for studying circular complete graphs is that they refine the notion of complete graphs. In particular, that they refine the notion of the chromatic number $\chi(G)$. Note that an alternative definition of $\chi(G)$ is given by $\chi(G) = \inf\{n \mid G \to K_n\}$. With this in mind, the following is a natural extension of proper graph colouring, and the chromatic number.

Definition 7.4. The circular chromatic number, $\chi_c(G)$, of a graph $G$ is defined as $\inf\{p/q \mid G \to K_{p/q}\}$. A homomorphism from $G$ to $K_{p/q}$ is called a (circular) $p/q$-colouring of $G$. 122
7.1. Circular Complete Graphs

We will now take a closer look at the symmetries of the circular complete graphs, as we are especially interested in what the orbits of the edges under the action of the automorphism group of these graphs look like. Let \( \delta_p(v_i, v_j) = j - i \mod p \). \( \delta_p(v_i, v_j) \) is then the directed circular distance (in positive direction) between \( v_i \) and \( v_j \). Furthermore, let \( \delta_p(v_i, v_j) = \min \{ \delta_p(v_i, v_j), \delta_p(v_j, v_i) \} \) denote the corresponding undirected circular distance.

**Lemma 7.5.** Let \( p, q \) and \( c \) be positive integers with \( p \geq 2q \) and \( q \leq c \leq p/2 \). Then, if \( K_{p/q} \) is the circular complete graph with \( V(K_{p/q}) = \{v_0, v_1, \ldots, v_{p-1}\} \), the edges \( \{v_i, v_j\} \) form an orbit of \( E(K_{p/q}) \) under \( \text{Aut}^*(K_{p/q}) \).

**Proof.** Since the graph is vertex-transitive it is obvious that there is an automorphism that maps \( v_i \) to \( v_j \) where \( \delta(v_i, v_j) = \delta(v_j, v_i) \). For two vertices \( v_i, v_j \) there are \( p - 2q + 1 - \delta(v_i, v_j) \) vertices which are adjacent to both \( v_i \) and \( v_j \), \( \delta(v_i, v_j) \leq p - 2q \), otherwise there are no vertices that are adjacent to both \( v_i \) and \( v_j \). Now, if there exists a \( \pi \in \text{Aut}^*(K_{p/q}) \) such that \( \delta(v_k, v_l) = \delta(\pi(v_k), \pi(v_l)) \), then there is at least one vertex \( v_i \), with \( \delta(v_k, v_l) \leq p - 2q \) and \( \delta(\pi(v_k), \pi(v_l)) = \delta(v_i, v_j) \). Consequently, we have that \( \delta(v_k, v_l) \neq \delta(\pi(v_k), \pi(v_l)) \), which makes such an automorphism impossible since the \( p - 2q + 1 - \delta(v_i, v_j) \) vertices adjacent to both \( v_i \) and \( v_j \) must be, and are the only ones which can be mapped to the \( p - 2q + 1 - \delta(\pi(v_k), \pi(v_l)) \) vertices that are adjacent to both \( \pi(v_k) \) and \( \pi(v_l) \). \( \square \)

**Corollary 7.6.** The graph \( K_{p/q} \) has \( \left\lceil \frac{p - 2q + 1}{2} \right\rceil \) orbits.

**Example 7.7.** \( \text{Aut}^*(K_{10/3}) \) has three orbits. Figure 7.1 depicts \( K_{10/3} \) and these orbits. \( A_1 \) (solid) is connecting vertices of a distance 3, \( A_2 \) (dashed) is connecting vertices of a distance 4, and \( A_3 \) (dotted) is connecting vertices of a distance 5.

For more on the circular complete graphs and the circular chromatic number, see the book by Hell and Nešetřil [HN04c], and the survey by Zhu [Zhu01].

We will now make an investigation of the separation parameter \( s(K_r, K_t) \) for rational numbers \( 2 \leq r < s \leq 3 \). In Section 7.1.1 we fix \( r = 2 \) and choose \( t \) so that \( \text{Aut}^*(K_t) \) has few orbits. We find some interesting properties of these numbers which lead us to look at certain “constant regions” in Section 7.1.2 and at the case \( r = 2 + 1/k \), in Section 7.1.3. Our method is based on solving a relaxation of the linear program (6.9) which was presented in Section 6.2.3 combined with arguments that the chosen relaxation in fact finds the optimum in the original program. Most of the calculations, which involve some rather lengthy ad hoc constructions of solutions, are left out. The complete proofs can be found in the technical report [EFJT09a].
7. Separation as a Graph Parameter

![Graph Diagram]

Figure 7.1: $K_{10/3}$ with vertices $\{v_0, \ldots, v_9\}$ and orbits $A_1, A_2$ and $A_3$.

7.1.1 Maps to an Edge

We consider $s(K_2, K_t)$ for $t = 2 + n/k$ with $k > n \geq 1$, where $n$ and $k$ are integers. The number of orbits of $\text{Aut}^*(K_t)$ then equals $\lceil (n+1)/2 \rceil$. We will denote these orbits by $A_c = \{\{v_i, v_j\} \in E(K_p/q) \mid j - i \equiv q + c - 1 \pmod{p}\}$, for $c = 1, \ldots, \lceil (n+1)/2 \rceil$. Since the number of orbits determine the number of variables of the linear program (6.9), we choose to begin our study of $s(K_2, K_t)$ using small values of $n$. For $n = 1$ we have seen that the graph $K_{2+1/k}$ is isomorphic to the cycle $C_{2k+1}$. For $n = 2$ we can assume that $k$ is odd in order to have $2k + n$ and $k$ relatively prime. We will write this number as $t = 2 + 2/(2k - 1) = \frac{4k}{2k - 1}$. Note that the graph $K_{8/3}$ from Example 6.20 is covered by Proposition 7.8. The argument in that example is very similar to the proof of the general case.

**Proposition 7.8.** Let $k \geq 1$ be an integer. Then, $s(K_2, K_{\frac{4k}{2k-1}}) = \frac{2k}{2k-1}$.

**Proof.** Let $V(K_{\frac{4k}{2k-1}}) = \{v_0, v_1, \ldots, v_{4k-1}\}$ and $V(K_2) = \{w_0, w_1\}$. We start by presenting two maps, or signatures, $f, g : V(K_{\frac{4k}{2k-1}}) \to V(K_2)$. The map $f$ sends $v_i$ to $w_0$ if $i$ is even and to $w_1$ if $i$ is odd. Our naming convention for the orbits of $\text{Aut}^*(K_{p/q})$ now means that $f$ maps all of $A_1$ to $K_2$ but none of the edges in $A_2$, so $f = (4k, 0)$. The solution $g$ sends a vertex $v_i$ to $w_0$ if
0 ≤ i < 2k and to w_i if 2k ≤ i < 4k. It is not hard to see that g = (4k - 2, 2k).

It remains to argue that these two solutions suffice to determine s. But we see that any map \( h = (h_1, h_2) \) with \( h_2 > 0 \) must cut at least two edges in the even cycle \( A_1 \). Therefore, \( h_1 \leq 4k - 2 \), so \( h \leq g \), component-wise. The proposition now follows by solving the relaxation of the linear program (6.9) using only the two inequalities obtained from \( f \) and \( g \).

**Example 7.9.** For \( K_{12/5} \), see Figure 7.2, we have one signature \( g \) with \( g^{-1}(w_0) = \{v_0, v_1, v_2, v_3, v_4, v_5\} \) and \( g^{-1}(w_6) = \{v_6, v_7, v_8, v_9, v_{10}, v_{11}\} \). We see that from the orbit \( A_1 \) (solid), only \( v_0v_5 \) and \( v_5v_{11} \) are not mapped to \( K_2 \), while all edges from \( A_2 \) (dotted) is mapped to \( K_2 \). We also have the solution \( f \) with \( f^{-1}(w_0) = \{v_0, v_2, v_4, v_6, v_8\} \) and \( f^{-1}(w_1) = \{v_1, v_3, v_5, v_7, v_9, v_{11}\} \) which maps all edges from \( A_1 \), but none from \( A_2 \), to \( K_2 \).

![Figure 7.2: \( K_{12/5} \) with vertices \( \{v_0, \ldots, v_{11}\} \)](image)

For \( n = 3 \), \( t = 2 + 3/k \), we see that if \( k \equiv 0 \pmod{3} \), then \( K_t \) is an odd cycle. If \( k \equiv 2 \pmod{3} \), then we can let \( k = 3k' - 1 \) and observe that \( 2 + 1/k' \leq t \leq 2 + 2/(2k' - 1) \). Hence, by Theorem 7.3 Lemma 6.13 and known values for \( s \), we have

\[
\frac{2k'}{2k' + 1} = s(K_2, C_{2k'+1}) \geq s(K_2, K_t) \geq s(K_2, K_{\frac{4k}{3k+1}}) = \frac{2k'}{2k' + 1}.
\]

It follows that \( s(K_2, K_t) = 2k'/(2k' + 1) = (2k + 2)/(2k + 5) \) as well. Therefore we assume that \( t \) is of the form \( 2 + 3/(3k + 1) = \frac{6k + 5}{3k + 1} \) for an integer \( k \geq 1 \).

**Proposition 7.10.** Let \( k \geq 1 \) be an integer. Then,

\[
s(K_2, K_{\frac{4k+1}{3k+1}}) = \frac{6k^2 + 8k + 3}{6k^2 + 11k + 5} = 1 - \frac{3k + 2}{(k + 1)(6k + 5)}.
\]
Proceeding to the cases for graphs $K_t$, with $t = 2 + 4/k$, we find that we only need to consider the case when $k \equiv 1 \pmod{4}$, since we otherwise end up inside the constant intervals we have only just discovered. This means then, that we have graphs $K_t$ with $t = 2 + 4/(4k + 1) = \frac{8k + 6}{4k + 1}$ for integers $k \geq 1$.

**Proposition 7.11.** Let $k \geq 1$ be an integer. Then,

$$s(K_2, K_{\frac{8k + 6}{4k + 1}}) = \frac{8k^2 + 6k + 2}{8k^2 + 10k + 3} = 1 - \frac{4k + 1}{(k + 1/2)(8k + 6)}.$$

The expressions for $s$ in Propositions 7.10 and 7.11 have some interesting regularities, but for $n \geq 5$ it becomes much harder to choose a suitable set of solutions. Using brute force computer calculations, we have determined the first two values (for $k = 1, 2$) in each of the cases $t = 2 + 5/(5k + 1)$ ($t = 17/6, 27/11$) and $t = 2 + 6/(6k + 1)$ ($t = 20/7, 32/13$). These values are summarised in Table 7.1.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$s(K_2, K_t)$</th>
<th>$t = 2 + 5/(5k + 1)$</th>
<th>$t = 2 + 6/(6k + 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>322/425 ≈ 0.7576</td>
<td>67/89 ≈ 0.7528</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5/6 ≈ 0.8333</td>
<td>94/113 ≈ 0.8319</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1: Some parameter values determined by brute force computer calculations

### 7.1.2 Constant Regions

In the previous section we saw that $s(K_2, C_{2k+1}) = s(K_2, K_{\frac{8k + 6}{4k + 1}})$ and used it to prove that $s(K_2, K_t)$ is constant in the interval $t \in [2 + 1/k, 2 + 2/(2k − 1)]$. This is a special case of a phenomenon described more generally in the following proposition.

**Proposition 7.12.** Let $k \geq 1$, and let $r$ and $t$ be rational numbers such that $2 \leq r < \frac{2k + 1}{k} \leq t \leq \frac{4k}{2k − 1}$. Then,

$$s(K_r, K_t) = \frac{2k}{2k + 1}.$$  

**Proof.** From Theorem 7.3, we have the following chain of homomorphisms.

$$K_2 \rightarrow K_r \rightarrow K_{\frac{2k+1}{k}} \rightarrow K_t \rightarrow K_{\frac{4k}{2k − 1}}.$$  

By Lemma 6.13, this implies

$$s(K_r, K_{\frac{2k+1}{k}}) \geq s(K_2, K_{\frac{4k}{2k − 1}}) = \frac{2k}{2k + 1}.$$  

126
but since $K_{2k+1}$ $\nRightarrow$ $K_r$, and $K_{2k+1}$ is edge-transitive with $2k + 1$ edges, $s(K_r, K_{2k+1}) \leq \frac{2k}{2k+1}$ and therefore $s(K_r, K_{2k+1}) = \frac{2k}{2k+1}$. Two more applications of Lemma 6.13 show that

$$\frac{2k}{2k+1} = s(K_r, K_{2k+1}) \geq s(K_r, K_t) \geq s(K_2, K_{\frac{2k}{2k+1}}) = \frac{2k}{2k+1},$$

which proves the proposition. □

We find that there are intervals $I_k = \{ t \in \mathbb{Q} \mid 2 + 1/k \leq t \leq 2 + 2/(2k - 1) \}$ where the function $s_r(t) = s(K_r, K_t)$ is constant for any $2 \leq r < (2k + 1)/k$. In Figure 7.3 these intervals are shown for the first few values of $k$. The intervals $I_k$ form an infinite sequence with endpoints tending to 2.

$$
\begin{array}{cccccccc}
 t & 2 & 2.16 & 2.17 & 2.5 & 2.6 & 2.8 & 3 \\
 s_r(t) & \frac{8}{5} & \frac{6}{7} & \frac{4}{5} & \frac{8}{5} & \frac{6}{7} & \frac{4}{5} & \frac{8}{5} \\
\end{array}
$$

Figure 7.3: The intervals $I_k$ marked for $k = 2, 3, 4$.

It turns out that similar intervals appear throughout the space of circular complete graphs. Indeed, it follows from Proposition 6.11 that for a positive integer $n$ and a rational number $r$ such that $2 \leq r \leq n$, we have

$$s(K_r, K_n) = \left\lfloor \left(1 - \frac{1}{|r|}\right) \cdot \frac{n^2}{2} \right\rfloor / \left\lfloor \frac{n}{2} \right\rfloor. \quad (7.1)$$

From (7.1) we see that $s(K_r, K_n)$ remains constant for rational numbers $r$ in the interval $k \leq r < k + 1$, where $k$ is any fixed integer $k < n$. Furthermore, for positive integers $k$ and $m$, we have

$$e(T(km - 1, k)) = \left\lfloor \left(1 - \frac{1}{k}\right) \cdot \frac{(km - 1)^2}{2} \right\rfloor = \left\lfloor \frac{(k-1)km^2}{2} - (k - 1)m + k - 1}{2k} \right\rfloor = \left\lfloor \frac{(k-1)km^2}{2} - (k - 1)m = \left(\frac{k}{2}\right)m^2 \cdot \left(1 - \frac{2}{km}\right) \right\rfloor = e(T(km, k)) \cdot \left(\frac{km - 1}{2}\right) / \left(\frac{km}{2}\right).$$

Thus, $s(K_k, K_{km-1}) = s(K_k, K_{km})$. When we combine this fact with (7.1) and Lemma 6.13 we find that $s(K_r, K_t)$ is constant on each region $(r, t) \in [k, k + 1) \times [km - 1, km]$.
7. Separation as a Graph Parameter

7.1.3 Maps to Odd Cycles

It was seen in Proposition 7.12 that \( s(K_r, K_t) \) is constant on the region \((r, t) \in [2, 2 + 1/k] \times I_k \). In this section, we will study what happens when \( r \) remains in \( I_k \), but \( t \) assumes the value \( 2 + 1/k \). A first observation is that the absolute jump of the function \( s(K_r, K_t) \) when \( r \) goes from being less than \( 2 + 1/k \) to \( r = 2 + 1/k \) must be largest for \( t = 2 + 2/(2k - 1) \). Let \( V(K_{2+2/(2k-1)}) = \{v_0, \ldots, v_{2k-1}\} \) and \( V(K_{2+1/k}) = \{w_0, \ldots, w_{2k}\} \), and let the function \( f \) map \( v_i \) to \( w_j \), with \( j = \left\lfloor \frac{2k+1}{4k} \cdot i \right\rfloor \). Then, \( f \) maps all edges except \( \{v_0, v_{2k-1}\} \) from the orbit \( A_1 \) to some edge in \( K_r \). Since the subgraph \( A_1 \) is isomorphic to \( C_{4k} \), any map to an odd cycle must exclude at least one edge from \( A_1 \). It follows that \( f \) alone determines \( s \), and we can solve the linear program \((6.9)\) to obtain \( s(K_{2+1/k}, K_{2+2/(2k-1)}) = (4k - 1)/4k \). Thus, for \( r < 2 + 1/k \), we have

\[
s(K_{2+1/k}, K_{2+2/(2k-1)}) - s(K_r, K_{2+2/(2k-1)}) = \frac{2k - 1}{4k(2k + 1)}.
\]

Smaller \( t \in I_k \) can be expressed as \( t = 2 + 1/(k - x) \), where \( 0 < x < 1/2 \). We will write \( x = m/n \) for positive integers \( m \) and \( n \) which implies the form \( t = 2 + n/(kn - m) \), with \( m < n/2 \). For \( m = 1 \), it turns out to be sufficient to keep two inequalities from \((6.9)\) to get an optimal value of \( s \). From this we get the following result.

**Proposition 7.13.** Let \( k, n \geq 2 \) be integers. Then,

\[
s(C_{2k+1}, K_{2kn-1}+n) = \frac{(2(kn - 1) + n)(4k - 1)}{(2(kn - 1) + n)(4k - 1) + 4k - 2}.
\]

There is still a non-zero jump of \( s(K_r, K_t) \) when we move from \( K_r < 2 + 1/k \) to \( K_r = 2 + 1/k \), but it is smaller, and tends to 0 as \( n \) increases. For \( m = 2 \), we have \( 2(kn - m) + n \) and \( kn - m \) relatively prime only when \( n \) is odd. In this case, it turns out that we need to include an increasing number of inequalities to obtain a good relaxation. Furthermore, we are not able to ensure that the obtained value is the optimum of the original \((6.9)\). We will therefore have to settle for a lower bound on \( s \). Brute force calculations have shown that, for small values of \( k \) and \( n \), equality holds in Proposition 7.14. We conjecture this to be true in general.

**Proposition 7.14.** Let \( k \geq 2 \) be an integer and \( n \geq 3 \) be an odd integer. Then,

\[
s(C_{2k+1}, K_{2(kn-2)+n}) \geq \frac{(2(kn - 2) + n)(\xi_n(4k - 1) + (2k - 1))}{(2(kn - 2) + n)(\xi_n(4k - 1) + (2k - 1)) + (4k - 2)(1 - \xi_n)},
\]

where \( \xi_n = \frac{z_1^{n+1} + z_2^{n+1}}{4} \), and \( z_1^{-1}, z_2^{-1} \) are the roots of \( 2k - 3 \frac{4k - 2}{2} z^2 - 2z + 1 \).
7.2 Fractional Covering by H-cuts

In the following, we generalise the work of Šámal [Šám05; Šám06; Šám12] on fractional covering by cuts. We obtain a complete correspondence between a family of chromatic numbers, \( \chi_H(G) \), and \( s(H, G) \). These chromatic numbers are generalisations of Šámal’s cubical chromatic number \( \chi_q(G) \); the latter corresponds to the case when \( H = K_2 \). Two more expressions for \( \chi_H(G) \) are given in Section 7.2.2. We believe that these alternative views on the separation parameter can provide great benefits to the understanding of its properties. We transfer a result in the other direction, in Section 7.2.3, disproving a conjecture by Šámal on \( \chi_q \), and settle another conjecture by him in the positive, in Section 7.2.4.

7.2.1 Separation as a Chromatic Number

We start by recalling the notion of a fractional colouring of a hypergraph. Let \( G \) be a (hyper-) graph with vertex set \( V(G) \) and edge set \( E(G) \subseteq 2^{V(G)} \). A subset \( J \) of \( V(G) \) is called independent in \( G \) if no edge \( e \in E(G) \) is a subset of \( J \).

Let \( J \) denote the set of all independent sets of \( G \) and for a vertex \( v \in V(G) \), let \( J(v) \) denote all independent sets which contain \( v \). Let \( J_1, \ldots, J_n \in J \) be a collection of independent sets.

**Definition 7.15.** An \( n/k \) independent set cover is a collection \( J_1, \ldots, J_n \) of independent sets in \( J \) such that every vertex of \( G \) is in at least \( k \) of them. The fractional chromatic number \( \chi_f(G) \) of \( G \) is given by the following expression.

\[
\chi_f(G) = \inf\left\{ \frac{n}{k} \mid \text{there exists an } n/k \text{ independent set cover of } G \right\}.
\]

The definition of fractional covering by cuts mimics that of fractional covering by independent sets, but replaces vertices with edges and independent sets with certain cut sets of the edges. Let \( G \) and \( H \) be undirected simple graphs and \( f \) be an arbitrary vertex map from \( G \) to \( H \). Recall that the map \( f \) induces a partial edge map \( f^\#: E(G) \to E(H) \). We will call the pre-image of \( E(H) \) under \( f^\# \) an \( H \)-cut in \( G \). When \( H \) is a complete graph \( K_k \), this is precisely the standard notion of a \( k \)-cut in \( G \). Let \( C \) denote the set of \( H \)-cuts in \( G \) and for an edge \( e \in E(G) \), let \( C(e) \) denote all \( H \)-cuts which contain \( e \). The following definition is a generalisation of cut \( n/k \)-covers [Šám06] to arbitrary \( H \)-cuts.

**Definition 7.16.** An \( H \)-cut \( n/k \)-cover of \( G \) is a collection \( C_1, \ldots, C_N \in C \) such that every edge of \( G \) is in at least \( k \) of them.

The graph parameter \( \chi_H \) is defined as:

\[
\chi_H(G) = \inf\left\{ \frac{n}{k} \mid \text{there exists an } H \text{-cut } n/k \text{-cover of } G \right\}.
\]
Šámal [Šám06] called the parameter $\chi_{K_2}(G)$, the cubical chromatic number of $G$. Both the fractional chromatic number and the cubical chromatic number also have linear programming formulations. This, in particular, shows that the value in the infimum of the corresponding definition is obtained exactly for some $n$ and $k$. For our generalisation of the cubical chromatic number, the linear program is the following:

$$\text{Minimise } \sum_{C \in C} f(C)$$
$$\text{subject to } \sum_{C \in C(e)} f(C) \geq 1 \text{ for all } e \in E(G),$$
$$f : C \rightarrow \mathbb{Q}_{\geq 0}.$$  \hspace{1cm} (7.2)

**Proposition 7.17.** The graph parameter $\chi_H(G)$ is given by the optimum of the linear program in (7.2).

**Proof.** The proof is completely analogous to those for the corresponding statements for the fractional chromatic number (cf. Godsil and Royle [GR01]) and for the cubical chromatic number (Lemma 5.1.3 in Šámal [Šám06]). Let $C_1, \ldots, C_n$ be an $H$-cut $n/k$-cover of $G$. The solution $f(C) = 1/k$ if $C \in \{C_1, \ldots, C_n\}$, and $f(C) = 0$ otherwise, has a measure of $n/k$. For our generalisation of the cubical chromatic number, the linear program is

Minimise $\sum_{C \in C} f(C)$
subject to $\sum_{C \in C(e)} f(C) \geq 1$ for all $e \in E(G)$,
$f : C \rightarrow \mathbb{Q}_{\geq 0}$.

Thus, the optimum of the linear program is at most $\chi_H(G)$.

For the other direction, note that the coefficients of the program (7.2) are integral. Hence, there is a rational optimal solution $f^*$. Let $N$ be the least common multiple of the divisors of $f^*(C)$ for $C \in C$. Assume that the measure of $f^*$ is $n/k$. Construct a collection of $H$-cuts by including the cut $C$ a total of $N \cdot f^*(C)$ times. This collection covers each edge at least $N$ times using $\sum_{C \in C} N \cdot f^*(C) = N \cdot n/k$ cuts, i.e. it is an $H$-cut $n/k$-cover, so $\chi_H(G)$ is at most equal to the optimum of (7.2). \hfill \Box

We are now ready to work out the correspondence to separation.

**Proposition 7.18.** The identity $\chi_H(G) = 1/s(H,G)$ holds for all $G, H \in \mathcal{G}$.

**Proof.** Consider the dual program of (7.2).

Maximise $\sum_{e \in E(G)} g(e)$
subject to $\sum_{e \in X} g(e) \leq 1$ for all $H$-cuts $X \in C$,
$g : E(G) \rightarrow \mathbb{Q}_{\geq 0}$.

In (7.3) let $1/s = \sum_{e \in E(G)} g(e)$ and make the variable substitution $w = g \cdot s$. This leaves the following program.

Maximise $s^{-1}$
subject to $\sum_{e \in X} w(e) \leq s$ for all $H$-cuts $X \in C$,
$\sum_{e \in E(G)} w(e) = 1$,
$w : E(G) \rightarrow \mathbb{Q}_{\geq 0}$.

130
7.2. Fractional Covering by $H$-cuts

Since $\max s^{-1} = (\min s)^{-1}$, a comparison with the linear program given for $s$ in (6.8) establishes the proposition. \hfill \Box

7.2.2 More Guises of Separation

For fractional colourings, it is well-known that an equivalent definition is obtained by taking $\chi_f(G) = \inf\{n/k \mid G \to K_{n,k}\}$, where $K_{n,k}$ denotes the Kneser graph, the vertex set of which is the $k$-subsets of $[n]$ and with an edge between $u$ and $v$ if $u \cap v = \emptyset$. For $H = K_2$, a corresponding definition of $\chi_H(G) = \chi_q(G)$ was obtained by Šámal [Šám06] through taking the infimum over $n/k$ for $n$ and $k$ such that $G \to Q_{n/k}$. Here, $Q_{n/k}$ is the graph on vertex set $\{0,1\}^n$ with an edge between $u$ and $v$ if $d_H(u,v) \geq k$, where $d_H$ denotes the Hamming distance.

A parameterised graph family which determines a particular chromatic number in this way is sometimes referred to as a scale. In addition to the previously mentioned fractional chromatic number $\chi_f$, where the scale is the set of Kneser graphs, and the cubical chromatic number $\chi_q$, where the scale is $\{Q_{n,k}\}$, another prominent example is the circular chromatic number (Section 7.1) for which the scale is given by the family of circular complete graphs $K_{n/k}$.

We now generalise the family $\{Q_{n,k}\}$ to produce one scale for each $\chi_H$. To this end, let $H^n_k$ be the graph on vertex set $V(H)^n$ and an edge between $(u_1,\ldots,u_n)$ and $(v_1,\ldots,v_n)$ when $|\{i \mid \{u_i,v_i\} \in E(H)\}| \geq k$. The proof of the following proposition is straightforward, but instructive.

**Proposition 7.19.** For $G,H \in \mathcal{G}$, we have

$$\chi_H(G) = \inf\{n/k \mid G \to H^n_k\}. \quad (7.5)$$

**Proof.** Both sides are defined by infima over $n/k$. Therefore, it will suffice to show how to translate each of the parameterised objects (an $H$-cut $n/k$-cover on the left-hand side and a homomorphism from $G$ to $H^n_k$ on the right-hand side) into the other, for some given values of $n$ and $k$.

Let $h : V(G) \to H^n_k$ be a homomorphism and denote by $pr_i h$ the projection of $h$ onto the $i$th coordinate. Then, for each edge $e \in E(G)$, at least $k$ of the edge maps $(pr_i h)^k$ must map $e$ to an edge in $H$. Hence, the $H$-cuts $C_i = \{e \in E(G) \mid (pr_i h)^k(e) \in E(H)\}$, for $1 \leq i \leq n$, constitute an $H$-cut $n/k$-cover of $G$.

For the other direction, note that each $H$-cut $C_i$ can be defined by a vertex map $f_i : V(G) \to H$. For $v \in V(G)$, let $h'(v) = (f_1(v),f_2(v),\ldots,f_n(v))$. To verify that $h'$ is a homomorphism, note that for every edge $\{u,v\} \in E(G)$, at least $k$ of the $f_i$ must include $e$ in their corresponding cut $C_i$. Hence $|\{i \mid \{f_i(u),f_i(v)\} \in E(H)\}| \geq k$, so by definition $\{h'(u),h'(v)\} \in E(H^n_k)$. \hfill \Box

Šámal further notes that $\chi_q(G)$ is given by the fractional chromatic number of a certain hypergraph associated to $G$. Inspired by this, we provide a similar formulation in the general case.
Proposition 7.20. Let $G'$ be the hypergraph obtained from $G$ on vertex set $V(G') = E(G)$ with edge set $E(G')$ taken to be the set of minimal subgraphs $K \subseteq G$ such that $K \not\rightarrow H$. Then,

$$\chi_H(G) = \chi_f(G').$$

Proof. We will let $\mathcal{J}$ denote the set of independent sets in $G'$ and $C$ the set of $H$-cuts in $G$. The parameter $\chi_H(G)$ is the infimum of $n/k$ over all $n/k$-covers of $E(G)$ by sets in $C$. Similarly, the parameter $\chi_f(G')$ is the infimum of $n/k$ over all $n/k$-covers of $V(G') = E(G)$ by sets in $\mathcal{J}$. By definition, the independent sets of $G'$ correspond precisely to the edge sets $E(K)$ of those subgraphs $K \subseteq G$ such that $K \rightarrow H$. Hence, $C \subseteq \mathcal{J}$, so $\chi_H(G) \geq \chi_f(G')$.

On the other hand, assume that $K \subseteq G$ is a subgraph such that $K \rightarrow H$, and that the independent set $E(K) \in \mathcal{J}$ is not in $C$. Then, any homomorphism $h : V(K) \rightarrow V(H)$ induces an $H$-cut $C$ of $G$, and clearly we must have $E(K) \subseteq C$. Thus, we can replace all edge sets in a cover by sets from $C$, without violating the constraint that all edges are covered at least $k$ times. The proposition now follows. \hfill \Box

7.2.3 An Upper Bound

In Section 7.1, we obtained lower bounds on $s$ by relaxing the linear program (6.9). In most cases, the corresponding solution was proved feasible in the original program, and hence optimal. Now, we take a look at the only known source of general upper bounds for $s$.

Let $G, H \in \mathcal{G}$, with $H \rightarrow G$ and let $S$ be such that $S \rightarrow G$. Then, applying Lemma 6.13 followed by Theorem 6.9 gives

$$s(H, G) \leq s(H, S) = \inf_{w \in W(S)} mc_H(S, w) \leq mc_H(S, 1/|E(S)|). \quad (7.6)$$

We can therefore upper bound $s(H, G)$ by the least maximal $H$-cut taken over all subgraphs of $G$. For $H = K_2$, we have

$$s(K_2, G) \leq \min_{S \subseteq G} b(S),$$

where $b(S)$ denotes the bipartite density of $S$. Conjecture 5.5.3 in Šámal [Šámal06] suggested that this inequality, expressed on the form $\chi_d(S) \geq 1/(\min_{S \subseteq G} b(S))$, could be replaced by an equality. We answer this in the negative, using $K_{11/4}$ as our counterexample. Lemma 7.10 with $k = 1$ gives $s(K_2, K_{11/4}) = 17/22$. If $s(K_2, K_{11/4}) = b(S)$ for some $S \subseteq K_{11/4}$ it means that $S$ must have at least 22 edges. Since $K_{11/4}$ has exactly 22 edges it follows that $S = K_{11/4}$. However, a cut in a cycle must contain an even number of edges. Since the edges of $K_{11/4}$ can be partitioned into two cycles, we have that the maximum cut in $K_{11/4}$ must be of even size, hence $|E(K_{11/4})| \cdot b(K_{11/4}) \neq 17$. This is a contradiction.
7.2. Fractional Covering by H-cuts

7.2.4 Confirmation of a Scale

As a part of his investigation of the cubical chromatic number, Šámal [Šám06] set out to determine the value of $\chi_f(Q_{n/k})$ for general $n$ and $k$. For the fractional chromatic number and the circular chromatic number, results for such measuring of the scale exist and provide very appealing formulae: $\chi_f(K_{n,k}) = \chi_C(C_{n,k}) = n/k$. For $\chi_f(Q_{n/k}) = 1/s(K_2, Q_{n/k})$, we are immediately out of luck as $1/2 < s(K_2, G) \leq 1$, i.e. $1 < \chi_f(G) < 2$ for all non-empty graphs. For $1 < n/k < 2$, however, Šámal gave a conjecture (Conjecture 5.4.2 in Šámal [Šám06]). We complete the proof of his conjecture to obtain the following result.

**Proposition 7.21.** Let $k, n$ be integers such that $k \leq n < 2k$. Then,

$$\chi_f(Q_{n/k}) = \begin{cases} n/k & \text{if } k \text{ is even, and} \\ (n+1)/(k+1) & \text{if } k \text{ is odd.} \end{cases}$$

**Corollary 7.22.** Let $k, n$ be integers such that $1/2 < k/n \leq 1$. Then, $s(K_2, Q_{n/k}) = k/n$ if $k$ is even, and $(k+1)/(n+1)$ if $k$ is odd.

If we make sure that $k$ is even, possibly by multiplying both $k$ and $n$ by a factor two, we get the following interesting corollary.

**Corollary 7.23.** For every rational number $r$, $1/2 < r \leq 1$, there is a graph $G$ such that $s(K_2, G) = r$.

Šámal [Šám06] provides the upper bound for Proposition 7.21 and an approach to the lower bound of using the largest eigenvalue of the Laplacian of a subgraph of $Q_{n/k}$. The computation of this eigenvalue boils down to an inequality (Conjecture 5.4.6 in Šámal [Šám06]) involving some binomial coefficients. We first introduce the necessary notation and then prove the remaining inequality in Proposition 7.25, whose second part, for odd $k$, corresponds to one of the formulations of the conjecture. Proposition 7.23 then follows from Theorem 5.4.7 in Šámal [Šám06], conditioned on the result of this proposition.

Let $k, n$ be positive integers such that $k \leq n$, and let $x$ be an integer such that $1 \leq x \leq n$. For $k \leq n < 2k$, let $S_o(n,k,x)$ denote the set of all $k$-subsets of $[n]$ that have an odd-sized intersection with $[n] \setminus [n-x]$. Define $S_e(n,k,x)$ analogously as the $k$-subsets of $[n]$ that have an even-sized intersection with $[n] \setminus [n-x]$, i.e. $S_e(n,k,x) = \binom{[n]}{k} \setminus S_o(n,k,x)$. Let $N_o(n,k,x) = |S_o(n,k,x)|$ and $N_e(n,k,x) = |S_e(n,k,x)|$. Then,

$$N_o(n,k,x) = \sum_{\text{odd } t} \binom{x}{t} \binom{n-x}{k-t} \quad \text{and} \quad N_e(n,k,x) = \sum_{\text{even } t} \binom{x}{t} \binom{n-x}{k-t}.$$  

When $x$ is odd, the function $f : S_o(2k,k,x) \to S_e(2k,k,x)$, given by the complement $f(\sigma) = [n] \setminus \sigma$, is a bijection. Since $N_o(n,k,x) + N_e(n,k,x) = \binom{n}{k}$, we have

$$N_o(2k,k,x) = N_e(2k,k,x) = \frac{1}{2} \binom{2k}{k}.$$  \hfill (7.7)
Lemma 7.24. Assume that \( x \) is odd, with \( 1 \leq x < n = 2k - 1 \). Then, \( N_e(n,k,x) = N_e(n,k,x+1) \) and \( N_o(n,k,x) = N_o(n,k,x+1) \).

Proof. First, partition \( S_e(n,k,x) \) into \( A_1 = \{ \sigma \in S_e(n,k,x) \mid n-x \notin \sigma \} \) and \( A_2 = S_e(n,k,x) \setminus A_1 \). Similarly, partition \( S_e(n,k,x+1) \) into \( B_1 = \{ \sigma \in S_e(n,k,x+1) \mid n-x \notin \sigma \} \) and \( B_2 = S_e(n,k,x+1) \setminus B_1 \). Note that \( A_1 = B_1 \).

We argue that \( |A_2| = |B_2| \). To prove this, define the function \( f : 2^n \to 2^{n-1} \) by

\[
f(\sigma) = (\sigma \cap [n-x-1]) \cup \{ s-1 \mid s \in \sigma, s > n-x \}.
\]

That is, \( f \) acts on \( \sigma \) by ignoring the element \( n-x \) and renumbering subsequent elements so that the image is a subset of \([n-1]\). Note that \( f(A_2) = S_e(2k-2,k-1,x) \) and \( f(B_2) = S_e(2k-2,k-1,x) \). Since \( x \) is odd, it follows from (7.7) that \( |f(A_2)| = |f(B_2)| \). The first part of the lemma now follows from the injectivity of the restrictions \( f|A_2 \) and \( f|B_2 \). The second equality is proved similarly.

Proposition 7.25. Choose \( k, n \) and \( x \) so that \( k \leq n < 2k \) and \( 1 \leq x \leq n \). Then,

\[ N_e(n,k,x) \leq \binom{n-1}{k-1} \text{ for odd } k, \text{ and } N_o(n,k,x) \leq \binom{n-1}{k-1} \text{ for even } k. \]

Proof. We will proceed by induction over \( n \) and \( x \). The base cases are given by \( x = 1, x = n \), and \( n = k \). For \( x = 1 \),

\[ N_o(n,k,x) = \binom{n-1}{k-1} \text{ and } N_e(n,k,x) = \binom{n-1}{k} \leq \binom{n-1}{k-1}, \]

where the inequality holds for all \( n < 2k \). For \( x = n \) and odd \( k \), we have \( N_e(n,k,x) = 0 \), and for even \( k \), we have \( N_o(n,k,x) = 0 \). For \( n = k \),

\[ N_e(n,k,x) = 1 - N_o(n,k,x) = \begin{cases} 1 & \text{if } x \text{ is even,} \\ 0 & \text{otherwise.} \end{cases} \]

Let \( x > 1 \) and consider \( N_e(n,k,x) \) for odd \( k \) and \( k < n < 2k-1 \). Partition the sets \( \sigma \in S_e(n,k,x) \) into those for which \( n \in \sigma \) on the one hand and those for which \( n \notin \sigma \) on the other hand. These parts contain \( N_o(n-1,k-1,x-1) \) and \( N_e(n-1,k,x-1) \) sets, respectively. Since \( k-1 \) is even, and since \( k \leq n-1 < 2(k-1) \) when \( k < n < 2k-1 \), it follows from the induction hypothesis that

\[
N_e(n,k,x) = N_o(n-1,k-1,x-1) + N_e(n-1,k,x-1)
\leq \binom{n-2}{k-2} + \binom{n-2}{k-1} = \binom{n-1}{k-1}.
\]

The case for \( N_o(n,k,x) \) and even \( k \) is treated identically.
Finally, let \( n = 2k - 1 \). If \( x \) is odd, then Lemma 7.24 is applicable, so we can assume that \( x \) is even. Now, as before

\[
N_e(2k-1, k, x) = N_e(2k-2, k-1, x-1) + N_e(2k-2, k, x-1)
\leq \frac{1}{2} \binom{2k-2}{k-1} + \binom{2k-3}{k-1} = \binom{n-1}{k-1},
\]

where the first term is evaluated using (7.7). The same inequality can be shown for \( N_o(2k-1, k, x) \) and even \( k \), which completes the proof. 

\[\square\]

7.3 Discussion and Open Problems

For a graph \( G \) with a circular chromatic number \( r \) close to 2 we can use Lemma 6.13 to bound \( s(K_2, G) \geq s(K_2, K_r) \). Due to Proposition 7.8 we have also seen that with this method, we are unable to distinguish between the class of graphs with circular chromatic number \( 2 + 1/k \) and the (larger) class of graphs with circular chromatic number \( 2 + 2/(2k-1) \). Nevertheless, the method is quite effective when applied to sequences of graph classes for which the circular chromatic number tends to 2, as was the case in Proposition 6.32(1)–(3). Much of the extensive study conducted in this direction was instigated by the restriction of a conjecture by Jaeger [Ja88] to planar graphs. This conjecture is equivalent to the claim that every planar graph of girth at least \( 4k \) has a circular chromatic number at most \( 2 + 1/k \), for \( k \geq 1 \). The case \( k = 1 \) is Grötzsch’s theorem; that every triangle-free planar graph is 3-colourable. Currently, the best lower bound on the girth of a planar graph which implies a circular chromatic number of at most \( 2 + 1/k \) is \( 20k-2 \), and is due to Borodin, Kim, Kostochka, and West [BKKW04]. We remark that Jaeger’s conjecture implies a weaker statement in our setting. Namely, if \( G \) is a planar graph with girth greater than \( 4k \), then \( G \to C_k \) implies \( s(K_2, G) \geq s(K_2, C_k) = 2k/(2k+1) \). Deciding this to be true would certainly provide support for the original conjecture, and would be an interesting result in its own right. Our starting observation shows that the slightly weaker condition \( G \to K_{2+2/(2k-1)} \) implies the same result.

For edge-transitive graphs \( G \), it is not surprising that the expression \( s(K_r, G) \) assumes a finite number of values, as a function of \( r \). Indeed, Theorem 6.9 states that \( s(K_r, G) = m c_{K_r}(G, 1/e(G)) \), which leaves at most \( e(G) \) possible values for \( s \). This produces a number of constant intervals that are partly responsible for the constant regions of Proposition 7.12 and the discussion in Section 7.1.2. More surprising are the constant intervals that arise from \( s(K_r, K_{2+2/(2k-1)}) \). They give some hope that the behaviour of the separation parameter can be characterised more generally. We propose investigating the existence of more constant regions, and possibly showing that they tile the entire space.

In Section 7.2 we generalised the notion of covering by cuts due to Šámal. In doing this, we found a different interpretation of the separation parameter as an entire family of chromatic numbers. It is our belief that these
alternate viewpoints can benefit from each other. The refuted conjecture in Section 7.2.3 is an immediate example of this. It is tempting to look for a generalisation of Proposition 7.21 with $K_2$ replaced by an arbitrary graph $H$. A trivial upper bound of $s(H, H_n^k) \leq k/n$ is obtained from Proposition 7.19 but we have not identified anything corresponding to the parity criterion which appears in the case $H = K_2$. This leads us to believe that this bound can be improved upon. The approach of Šámal on the lower bound does not seem to generalise. The reason for this is that it uses bounds on maximal cuts obtained from the Laplacian of (a subgraph of) $Q_n/k$. We know of no such results for maximal $k$-cuts, with $k > 2$, much less for general $H$-cuts.
Conclusion
8 Final Thoughts

We take this opportunity to conclude the thesis with a chapter giving a more conceptual overview of the contributions in the thesis as well as an open-minded outlook on possible future extensions to these contributions.

8.1 Fractional Edge Covers and Combinatorial Auctions

In Part II of the thesis we dealt with structurally restricted problems from the CSP framework. In doing so, we have enlarged the landscape of tractability for several different CSP-related problems. In the case of bounded arity relational structures, the new classes of polynomial-time solvable problems are the largest possible, provided \( \text{FPT} \neq \text{W[1]} \). Our investigations of the unbounded arity case, as well as those concerning parameterised complexity are more open-ended with plenty of room for improvements. From a conceptual point of view, Part II of the thesis has the character of “levelling the playing field”, and confirming conjectures, in the sense that we had a lot of disparate knowledge concerning structurally restricted problems, and techniques for exploiting them; now we possess a more systematic and level view of this part of the computational complexity landscape.

The largest families of relational structures for which we have shown our CSP-related problems to be solvable in polynomial time are the ones having bounded fractional hypertree width. This is an interesting hypergraph invariant, deserving further attention; besides the approximation algorithm of Marx [Mar10a], Moll et al. [MTT12] give an exact algorithm computing the fractional hypertree width of an \( n \) vertex, \( m \) edge hypergraph in time \( O(1.734601^n \cdot m) \), leaving open the tantalising possibility of an exact
polynomial-time algorithm. Most probably, better knowledge about fractional edge covers themselves is needed to decide if this possibility can be turned into reality. To further emphasize the importance of this hypergraph invariant we choose to spend this section of the final chapter on sharing some thoughts about how it applies in the context of combinatorial auctions.

Auctions are used throughout today’s economy to allocate goods, resources, and services to agents, and there are numerous investigations into auction design. In this brief treatment, we will consider a type of auction where multiple items are sold, and where agents are allowed to simultaneously bid on combinations of items. This allows us to model the situation where an agents value for a bundle of items is not equal to the sum of the agents values for each individual item in that bundle. Combinatorial auctions provide a mechanism for modeling such behavior \cite{VV03}. This framework is currently used to regulate the interactions of agents on several application domains (cf. Sandholm and Suri \cite{SS03}), such as electricity markets \cite{MM96}, bandwidth auctions \cite{McM94}, and transportation exchanges \cite{San93}.

We will consider a combinatorial auction to consist of a set of $m$ items for sell by the auctioneer, and a set of $n$ bids from the agents interested in the items for sell. Each bid consists of a set of items and the price the agent is willing to pay for those items. The process of allocating items to bidders to maximize the profit for the auctioneer is known as the winner determination problem (WDP). The efficiency of the WDP greatly affects its real-world applicability, as waiting for a long time to determine a winner is not realistic. However, solving the WDP for unrestricted combinatorial auctions is known to be NP-hard \cite{RPH98}, and not even approximable in polynomial time unless \( P = NP \) \cite{San02;Zuc07}. With this in mind, research efforts have concentrated both on designing practically efficient algorithms for general auctions (e.g., \cite{Bou02;FLBS99;HB00;San06;ZN01}) and finding restrictions on auctions that produce polynomial time solvable winner determination problems (e.g. \cite{LOS02;Nis00;SS03;Ten00}).

Previously, the most general class of tractable combinatorial auctions came from constraining bidder interaction through their item graphs. These are graphs where each item is represented by exactly one node in the graph and where the subgraph induced by the items in each bid has to be connected. More specifically, the winner determination problem was shown to be solvable in polynomial time if its item graph is structured \cite{CD04}, or, more formally, has bounded tree-width. Unfortunately, Gottlob and Greco \cite{GG07} proved that it is NP-complete to check whether a combinatorial auction has a structured item graph of tree-width 3. This means that the above mentioned tractability result on structured item graphs is a lot less useful than it seems, since it either uses a given structured item graph or has to determine one efficiently. In the same paper, Gottlob and Greco started looking at constraining bidder interaction through a specific hypergraph associated with a combinatorial auction. They used the fact that an auction can be represented by a hypergraph $H$, where the set of nodes coincides with the set of items, and the set of edges are exactly the bids of the agents.
8.1. Fractional Edge Covers and Combinatorial Auctions

It is well-known that the winner determination problem is analogous to the maximum weighted set packing problem, denoted MaxWSP, on hypergraphs (e.g., Rothkopf, Pekec, and Harstad [RPH98]). In this problem, the hypergraph $H$ of an instance comes equipped with a weight function on its edges, and the objective is to find a set of hyperedges (a packing) of $H$, with maximum possible weight, such that the pairwise intersection of all edges in the set is empty.

As mentioned above, Gottlob and Greco [GG07] looked for a different kind of structural requirement that could be checked in polynomial time and still be used to identify polynomial-time solvable classes of maximum weighted set packing problems or, equivalently, winner determination problems. For a given hypergraph $H$, its dual $\overline{H}$ has nodes that are in one-to-one correspondence with hyperedges in $H$, and for each node in $H$, the dual has an edge consisting of the edges in $H$ containing the node. Gottlob and Greco [GG07] showed that MaxWSP is tractable on the class $C(\overline{H}, k)$, i.e., the class of instances whose dual hypergraphs have hypertree width bounded by $k$. Furthermore, they show that it is essential to look at dual hypergraphs $\overline{H}$, since MaxWSP remains NP-hard even when $H$ has hypertree width 1. More surprisingly, they also prove that the classes of instances that are tractable according to the hypertree-based decomposition method are strictly larger than those according to the structured item graph approach.

Intuitively, an instance of the winner determination problem should be easy to solve if we could pick a set of just a few items, such that every bid has an item in this set. The concept of bounded hypertree width of the dual hypergraph captures this situation by allowing clusters of such items arranged in a tree-like shape. Conversely, we could think of the situation where we, in some sense, have a lot of items that tightly constrain the possible packings in the corresponding set packing problem, or, in other words, that the edge set of the dual hypergraph is highly intersecting. Not surprisingly, we can capture this situation by looking at (dual) hypergraphs having bounded fractional edge cover number.

More concretely, we can specialize and apply the decomposition techniques from Chapter 4 together with a dynamic programming algorithm along the lines of Gottlob and Greco’s algorithm for the bounded hypertree width case to show that MaxWSP (and hence the WDP) is tractable on the class $C(\overline{H}, k)$, i.e., the class of instances whose dual hypergraphs have fractional hypertree width bounded by $k$. In particular, the interplay between the fractional edge covers, the tree decompositions involved, and the duality between packings in the hypergraph and independent sets in its dual make the situation quite complex. This result makes bounded fractional hypertree width the strictly most general known property that allows MaxWSP to be solved in polynomial time. To proceed we need to define some notation and define the computational problems involved more formally. For the most part, we will follow the terminology of Gottlob and Greco [GG07].
8.1.1 Combinatorial Auctions and Set Packing Problems

A combinatorial auction is defined as a pair \((I, B)\), where \(I = \{I_1, \ldots, I_m\}\) is the set of items the auctioneer has to sell, and \(B = \{B_1, \ldots, B_n\}\) is the set of bids from the agents interested in the items in \(I\). Each bid \(B_i\) has the form \((\text{item}(B_i), \text{pay}(B_i))\), where \(\text{pay}(B_i)\) is a rational number denoting the price a buyer offers for the items in \(\text{items}(B_i) \subseteq I\). An outcome for \((I, B)\) is a subset \(b\) of \(B\) such that \(\text{item}(B_i) \cap \text{item}(B_j) = \emptyset\), for each pair \(B_i\) and \(B_j\) of bids in \(b\) with \(i \neq j\). The revenue associated with \(b\), denoted \(\text{rev}(b)\), is the rational number \(\sum_{B_i \in b} \text{pay}(B_i)\). Now, the winner determination problem is to find the outcome that maximizes the revenue over all possible outcomes for a combinatorial auction.

Formally, a packing \(h\) for a hypergraph \(H\) is a set of hyperedges of \(H\) such that for each pair \(h, h' \in h\) with \(h \neq h'\), it holds that \(h \cap h' = \emptyset\). Letting \(w\) be a weight function for \(H\), i.e., a polynomial-time computable function from \(E(H)\) to rational numbers, the weight of a packing \(h\) is the rational number \(w(h) = \sum_{e \in h} w(e)\), where \(w(\{}\} = 0\). Then, the maximum weighted set packing problem for \(H\) with respect to \(w\), denoted \(\text{MaxWSP}(H,w)\), is the problem of finding a packing for \(H\) having the maximum weight over all packings for \(H\). As mentioned in the introduction, a combinatorial auction \((I, B)\) can be represented by a hypergraph \(H_{(I, B)}\), where the set of nodes \(V(H_{(I, B)})\) coincides with the set of items \(I\), and the set of edges \(E(H_{(I, B)})\) are exactly the bids of the agents \(\{\text{item}(B_i) \mid B_i \in B\}\). Now it is easy to see that \(\text{MaxWSP}\) is just a different formulation of the winner determination problem — given a combinatorial auction \((I, B)\), we can define the weight function \(w_{(I, B)}(\text{item}(B_i)) = \text{pay}(B_i)\). Then the set of solutions to \(\text{MaxWSP}(H_{(I, B)}, w_{(I, B)})\) is precisely the set of solutions to the winner determination problem on \((I, B)\).

Let \(H = (V(H), E(H))\) be a hypergraph. We always assume that hypergraphs have no isolated vertices, that is, for every \(v \in V(H)\) there exists at least one \(e \in E(H)\) such that \(v \in e\). For a given hypergraph \(H\), its dual \(\bar{H} = (V, E)\) is such that nodes in \(V\) are in one-to-one correspondence with hyperedges in \(H\), and for each node \(x \in V(H)\), \(\{h \mid x \in h \wedge h \in E(H)\}\) is in \(E\). Figure 8.1 shows an example hypergraph and its dual. A graph \(G = (V, E)\) is an item graph for \(H\) if \(V = V(H)\) and, for each \(h \in E(H)\), the subgraph of \(G\) induced over the nodes in \(h\) is connected. Note that any item graph for \(H\) can be viewed as a simplification of \(H\), the primal graph of \(H\), obtained by deleting some edges, while preserving the connectivity condition on the nodes included in each hyperedge.

8.1.2 Tractability via Fractional Edge Covers

We will now use the same approach as in the proof of Lemma 4.9 to show a similar result concerning set packings.

**Lemma 8.1.** A hypergraph \(H\) admits at most \(|E(H)|^{|\rho^*(H)|}\) different packings.
8.1. Fractional Edge Covers and Combinatorial Auctions

Proof. Let $H$ be a hypergraph and fix an arbitrary enumeration of its edge set $E(H) = \{e_1, \ldots, e_n\}$. Now, we will define an auxiliary hypergraph whose edges correspond to packings of the instance at hand. We will then use Lemma 4.8, the Generalised Weighted Entropy Lemma, to bound the number of edges in the auxiliary graph. To this end, let $\hat{H}$ be a hypergraph over $E(H) \times \{0, 1\}$, where the edges correspond to the packings of the instance: for each packing $h$ there is an edge $\{(e_1, e_1 \in h), (e_2, e_2 \in h), \ldots, (e_n, e_n \in h)\}$ in $\hat{H}$. Let $\psi$ be a fractional edge cover of $\bar{H}$ (the dual hypergraph of $H$) with $\sum_{e \in E(\bar{H})} \psi(e) = \rho^*(\bar{H})$.

For each edge $e_i$ in $\bar{H}$, we define $\hat{A}_i = e_i \times \{0, 1\}$ and let the weight $\alpha_{e_i}$ of $\hat{A}_i$ be the same as the weight $\psi(e_i)$. Since the nodes of $\bar{H}$ are in one-to-one correspondence with the hyperedges of $H$, the sets $\hat{A}_i$ are now a fractional cover of $\hat{H}$. We let $\hat{E}_i$ be the edge set of $\hat{H}[\hat{A}_i]$ (the dual hypergraph of $H$) with $\sum_{e \in E(\hat{H})} \psi(e) = \rho^*(\hat{H})$.

For each edge $e_j$ in $\hat{H}$, we define $\hat{A}_i = e_j \times \{0, 1\}$ and let the weight $\alpha_{e_j}$ of $\hat{A}_i$ be the same as the weight $\psi(e_j)$. Since the nodes of $\bar{H}$ are in one-to-one correspondence with the hyperedges of $H$, the sets $\hat{A}_i$ are now a fractional cover of $\hat{H}$. We let $\hat{E}_i$ be the edge set of $\hat{H}[\hat{A}_i]$, i.e., $\hat{E}_i = \{e \cap \hat{A}_i | e \in E(\hat{H})\}$, and set the weights $w_i$ of $e_i \in \hat{E}_i$ uniformly to one. By Lemma 4.8, we have

$$\frac{|E(\hat{H})|}{\prod_{i=1}^{\alpha} |\hat{E}_i|^{|A_i|}} \leq \left(\frac{\sum_{e_j \in \hat{E}_i} w_j(e_j)^{1/\alpha_i}}{w_i(e_i)}\right)^{\alpha_i}.$$  

Since the weights $w_i$ are all one, we can conclude that the number of edges of $\hat{H}$ can be bounded by

$$\frac{|E(\hat{H})|}{\prod_{i=1}^{\alpha} |\hat{E}_i|^{|A_i|}} \leq \left(\sum_{e_j \in \hat{E}_i} w_j(e_j)^{1/\alpha_i}\right)^{\alpha_i}.$$  

In a now familiar fashion, the hypergraph $\hat{H}[\hat{A}_i]$ describes what the packings look like if we consider only the hyperedges (from $H$) in $A_i$: each edge of $\hat{H}[\hat{A}_i]$ describes a possible combination of hyperedges from $A_i$ that a packing can contain. More precisely, assume that $A_i$ consists of the hyperedges $\{e'_1, \ldots, e'_k\}$. The hypergraph $\hat{H}[\hat{A}_i]$ contains the edge $\{(e'_1, b_1), \ldots, (e'_k, b_k)\}$ if and only if there is a packing $h$ such that $e_i \in h = h_i$ holds. But, $A_i$ is an edge in the dual hypergraph $H$, which means that all the $e_i$ have a node
8. Final Thoughts

in $V(H)$ in common, so at most one of the $e_i$ can occur in any packing at the same time. This leads us to the conclusion that there can be at most $|E(H)|$ edges in $\hat{E}_i$. Hence the number of edges of $\hat{H}$ and thereby the number of packings can be bounded by

$$|E(H)| \prod_{i=1}^{|E(H)|} |E(H)| = |E(H)| \prod_{i=1}^{|E(H)|} \alpha_i = |E(H)| \sum_{e \in E(\hat{H})} \psi(e) = |E(H)| \rho^\ast(\hat{H}).$$

Since the proof of Lemma 4.8 is not algorithmic, there is no immediate way to turn the upper bound of Lemma 8.1 into an algorithm for enumerating all packings. Instead, our algorithm will use a simple enumeration of the independent sets of the dual hypergraph. Generating all independent sets of a hypergraph is, of course, a problem with numerous application areas, whose polynomial time solvability is an important open question [EG95; EGM03]. Fortunately, Lemma 8.1 puts an effective bound on how many independent sets there can be, making the enumeration of the solution space terminate quickly.

Theorem 8.2. The packings of a hypergraph $H$ can be enumerated in time $|E(H)| \rho^\ast(\hat{H}) + O(1)$.

Proof. If $E = \{e_1, \ldots, e_n\}$ is an ordering of the hyperedges of $H$, then let $E_i = \{e_1, \ldots, e_i\}$. It is not hard to check that independent sets in $\hat{H}$ are in one-to-one correspondence with packings for $H$. Hence, a call to Algorithm 1 with $\hat{H}$ as input, solves our problem.

Algorithm 1: FindIndependentSets

Data: Hypergraph $H^\ast = (\{v_1, \ldots, v_n\}, \{e_1, \ldots, e_m\})$.

$\mathcal{N} \leftarrow \{v_1\},$ $L_1 \leftarrow \text{list containing } \emptyset \text{ and } \{v_1\}$

for $i = 1 \text{ to } n - 1$ do

$\mathcal{N} \leftarrow \mathcal{N} \cup \{v_{i+1}\},$ $L_{i+1} \leftarrow L_i$

foreach $\alpha$ in $L_i$ do

$\alpha' \leftarrow \alpha \cup \{v_{i+1}\}$

if $\alpha'$ is an independent set in $H^\ast[\mathcal{N}]$ then

add $\alpha'$ to $L_{i+1}$

add $\emptyset$ to $L_{i+1}$

return $L_n$

The algorithm creates lists $L_i$ containing the independent sets of $H^\ast[\{v_1, \ldots, v_i\}]$. To understand the correctness of Algorithm 1, the crucial observation is that an independent set of $H^\ast[\{v_1, \ldots, v_{i+1}\}]$ induces an independent set of $H^\ast[\{v_1, \ldots, v_i\}]$. This means that the list $L_{i+1}$ can be constructed in the manner of Algorithm 1 and that $L_n$ will hold all independent
8.2 Constraint Representation and Tractability

sets of \(H^*\). Clearly, the \(i\)th step can be done in \(|L_i| \cdot |H^*[\{v_1, \ldots, v_{i+1}\}]|^{O(1)}\) time.

By the above discussion, an upper bound on the running time of the algorithm in our case is \(\sum_{i=1}^{n-1} |L_i| \cdot |H[E_{i+1}]|^{O(1)}\). Furthermore, observing that any fractional edge cover of \(\bar{H}\) gives a fractional edge cover of \(H[E_i]\), we see that \(\rho^*(\bar{H}[E_i]) \leq \rho^*(\bar{H})\), for \(1 \leq i \leq n\). Hence, by Lemma 8.1, \(|L_i| \leq |E(H)|^{\rho^*(\bar{H})}\), and we can conclude that the total running time is \(|E(H)|^{\rho^*(\bar{H})} + O(1)\).

**Corollary 8.3.** \(\text{MaxWSP}\) can be solved in polynomial time on the class of all hypergraphs whose dual hypergraphs have bounded fractional edge cover number.

**Proof.** Immediate. Note that the algorithm of Theorem 8.2 does not need an actual fractional edge cover.

We are now in the position to prove that the maximum weighted set packing problem can be solved in polynomial time on the class \(\mathcal{C}(\bar{H}, k)\) of hypergraphs whose dual hypergraphs have fractional hypertree width bounded by \(k\). The algorithm behind Theorem 8.4 uses a given fractional hypertree decomposition to calculate all possible packings for each bag in the decomposition. By then using a dynamic programming step to filter out inconsistent partial packings, we can transform the instance to an acyclic instance of the problem, making the algorithm of Gottlob and Greco \([CG07]\), for such instances, applicable. This algorithm then chooses maximum weight partial solutions to finally arrive at an optimal solution for the whole instance.

**Theorem 8.4.** Let \(r \geq 1\), \(H\) be a hypergraph, and \(w\) a weight function for \(H\). Then there is a polynomial time algorithm that, given a fractional hypertree decomposition of \(H\) of width at most \(r\), computes an optimal solution to \(\text{MaxWSP}(H, w)\).

It would be interesting to investigate the practical implications of our results. How would implementations of our algorithms perform on real-world or realistic instances of the WDP (such as those generated by, e.g., CATS \([LBPS00]\)), when compared to state-of-the-art solvers? How do such instances behave with respect to the hypergraph invariants discussed in this section of the thesis?

### 8.2 Constraint Representation and Tractability

All the results in the thesis, so far, concern the situation when the constraints are represented by explicitly listing all tuples in their respective relations. As we mentioned in the introduction to Chapter 4, Marx \([Mar11]\) has studied the more verbose truth table representation. Here, a constraint of arity \(r\) is represented by having one bit for each possible \(r\)-tuple that can appear on the \(r\) variables of the constraint, indicating whether this particular \(r\)-tuple satisfies the constraint or not. To increase the flexibility of the representation and make it more natural we allow that the variables have different domains, making the size of the truth table of an \(r\)-ary constraint proportional...
to the size of the direct product of the domains of the \( r \) variables. While this type of representation does not have as strong motivation as listing all tuples does, investigating it is an important theoretical problem and, hopefully, the ideas can be useful in the study of more natural representations. By using Marx' \[\text{Mar11}\], highly inventive, new hypergraph width measure, \textit{adaptive width}, we can now identify a large class of tractable counting CSPs with truth table representation.

\subsection{Truth Tables and Bounded Adaptive Width}

We will let instances of \#\text{CSP}_t, the counting CSP problem with truth table representation, be quadruples \((V, D, \text{Dom}, C)\), where \(\text{Dom} : V \to 2^D\) assigns a domain \(\text{Dom}(v) \subseteq D\) to each variable \(v \in V\), and each constraint \(C = ((x_1, \ldots, x_k)R) \in C\) of arity \( k \) is represented by the truth table of the constraint relation \(R\), i.e., by a sequence of \(\prod_{i=0}^{k} |\text{Dom}(x_i)|\) bits that describe this subset \(R\) of \(\prod_{i=0}^{k} \text{Dom}(x_i)\).

The dual of covering is, of course, independence. Let \(H\) be a hypergraph. An \textit{independent set} of \(H\) is set \(X \subseteq V(H)\) of vertices such that \(|X \cap e| \leq 1\) for every \(e \in E(H)\). The \textit{independence number} of \(H\), denoted by \(\alpha(H)\) is the size of the largest independent set of \(H\). In a fashion similar to the treatment of edge covers, we can consider linear relaxations of independent sets, so that a function \(\phi : V(H) \to [0,1]\) is a \textit{fractional independent set} of \(H\) if \(\sum_{e \in E} \phi(v) \leq 1\) for every \(e \in E(H)\). The \textit{fractional independence number} \(\alpha^*(H)\) of \(H\) is the maximum of \(\sum_{e \in V(H)} \phi(v)\) taken over all fractional independent sets \(\phi\) of \(H\), and it is well known that \(\alpha(H) \leq \alpha^*(H) = \rho^*(H) \leq \rho(H)\) for every hypergraph \(H\). This means we could replace fractional edge covers with fractional independent sets in the definition of fractional hypertree width, so that \(\text{fhw}(H) \leq w\) means that there exists a tree decomposition \((T, (B_t)_{t \in V(T)})\) such that for all fractional independent sets \(\phi\), \(\phi(H[B_t]) \leq w\) for each \(t \in V(T)\).

The main conceptual contribution in Marx' \[\text{Mar11}\] treatment of CSPs with truth table representation is that we should look at the instance first and use different tree decompositions for different instances, instead of just blindly using the same fixed decomposition for each hypergraph. More concretely, Marx defines the \textit{adaptive width}, \(\text{adw}(H)\), of a hypergraph \(H\) by exchanging the two quantifiers in the alternative definition of fractional hypertree width above, so that \(\text{adw}(H) \leq w\) means that for all fractional independent sets \(\phi\), there exists a tree decomposition \((T, (B_t)_{t \in V(T)})\) such that \(\phi(H[B_t]) \leq w\) for each \(t \in V(T)\). Marx \[\text{Mar11}\] also demonstrates a class of \(\mathcal{H}\) of hypergraphs with bounded adaptive width such that \(\mathcal{H}\) has unbounded fractional hypertree width.

With this new tool, we can look at the distribution of domain sizes in an instance of \#\text{CSP}_t and derive a fractional independent set \(\phi\) based on these sizes. Then bounded adaptive width guarantees that there is a tree decomposition where this particular \(\phi\) is bounded on each bag. Marx \[\text{Mar11}\] also shows that we can find such a tree decomposition in polynomial time.
and that there is only a bounded number of solutions in each bag of this decomposition, which makes our dynamic programming algorithm from Lemma 4.1 applicable, thus proving the following theorem.

**Theorem 8.5.** Let \( A \) be a class of relational structures of bounded adaptive width. Then \( \#\text{CSP}_{tt}(A, \_\text{\_}) \) is solvable in polynomial time.

We remark that under a technical conjecture by Marx [Mar11], concerning the efficiency of the tree-width based algorithm for binary CSPs, this class of structurally restricted \( \#\text{CSP}_{tt} \) problems is the largest possible tractable class.

The truth table representation is a much more verbose representation of constraint relations than the explicit listing of all satisfying tuples. Naturally, you could also consider more compact representations. Chen and Grohe [CG10] has precisely characterised the tractable structural restriction for CSPs where the relations are represented in a type of generalised DNF format. This GDNF representation can be exponentially more succinct than the explicit representation. Furthermore, Chen and Grohe also characterise the structural restrictions of an even more succinct representation based on extending the ordered binary decision diagram representation of boolean functions to allow relations over arbitrary domains. As a first step, trying to extend these results to the counting setting should prove worthwhile. Secondly, trying to think of sensible ways of representing instances of optimisation problems, such as \( \text{MINHOM} \) and \( \text{VCSP} \), more compactly, in a similar fashion, seems like an interesting direction to take. In this context, we also want to mention the so called mixed representation of Cohen et al. [CGH09], which takes a place strictly between the GDNF and explicit representations with regards to succinctness.

### 8.3 Separation; Approximability, and as a Graph Parameter

Part III of the thesis is concerned with an exciting new approach to studying the approximability of optimisation problems. What started out as a very simple idea has diverged in a number of directions, with plenty of room in each for further investigation and improvements. In Chapters 6.4 and 7.3 we have given a rather thorough discussion of our contributions, as well as concrete open problems and research directions. Here, we single out two topics relating to the application of our approach to the approximability of the problem \( \text{MAX } H\text{-COL} \) (and more generally to \( \text{MAX CSP}(\Gamma) \)), and to the computation and interpretation of the separation parameter. We also want to mention the possibility of important new insights that could arise from applying our separation parameter to other optimisation problems than \( \text{MAX } H\text{-COL} \) and \( \text{MAX CSP} \). For instance, Thapper [Tha10] has worked out the basic details of adapting separation to the \( \text{MAX } \text{SO}\text{L} \) problem.

In the applications of the separation parameter in Section 6.3 we have used the complete graphs as our base set of known problems. For the purpose of extending the applicability of our method, a possible direction to
8. Final Thoughts

take is to find a larger base set of Max H-Col problems. We suggest two candidates for further investigation: the circular complete graphs, for which we have obtained partial results for the parameter $s$ in Section 7.1 and the Kneser graphs, see for example [HN04]. Both of these classes generalise the complete graphs, and have been subject to substantial previous research. The Kneser graphs contain many examples of graphs with low clique number, but high chromatic number. They could thus prove to be an ideal starting point for studying this phenomenon in relation to our parameter.

In recent work, Šámal and coauthors have shown that the cubical chromatic number $\chi_q$ can be approximated within $a_{GW}$ [Šám12]. This suggests the interesting possibility of a close connection between the approximability of $mc$ and that of $s(H,G)$, with $H$ fixed. Let $M$, $N$, and $H$ be graphs.

A function $g : E(M) \to E(N)$ is said to be H-cut continuous if, for any H-cut $C \subseteq E(N)$ in $N$, we have that $g^{-1}(C) \subseteq E(M)$ is an H-cut in $M$. For any homomorphism $h$, the edge map $h^*$ is H-cut continuous for every $H$. Šámal [Šám05] used cut continuous maps ($H = K_2$) to show that certain non-homomorphic graphs have the same cubical chromatic number. Here we show how general H-cut continuous maps can be used to generalise the implication in Lemma [GM06].

**Lemma 8.6.** Let $M$, $N$, and $H$ be graphs in $\mathcal{G}$. If there exists an M-cut continuous map from $H$ to $N$, then $s(M,H) \geq s(M,N)$.

**Proof.** Let $f : E(H) \to E(N)$ be an M-cut continuous function. It suffices to show that for any graph $H \in \mathcal{G}$ and $w \in W(H)$, we have

$$mc_M(H,w) \geq mc_M(N,w_N),$$

where $w_N(e) = \sum_{e' \in f^{-1}(e)} w(e')$. Let $g : V(N) \to V(M)$ be an optimal solution to $(N,w_N)$. Then, $C = (g^*)^{-1}(E(M))$ is an M-cut in $N$, so $f^{-1}(C)$ is an M-cut in $H$. Hence, there exists a solution to $(H,w)$ which contains precisely the edges in $f^{-1}(C)$. The measure of this solution is given by

$$\sum_{e \in f^{-1}(C)} w(e) = \sum_{e \in (g^*)^{-1}(E(M))} \sum_{e' \in f^{-1}(e)} w(e') = m_M(g).$$

Since $g$ is optimal, $m_M(g) = mc_M(N,w_N)$, and inequality (8.1) holds. 

The possibility of efficiently computing (bounds on) $s(M,N)$ have an immediate application: Lemma 8.13 and Lemma 8.6 give necessary conditions for the existence of a homomorphism $N \to M$. As noted by Šámal [Šám12], this can be used as a no-homomorphism lemma, proving the absence of a homomorphism between two given graphs. Needless to say, establishing such properties is often a non-trivial task.

To illustrate what we mean by gaining new insights through the lens of our separation parameter, we finish the thesis by mentioning that there is an interesting convergence between concepts from Part II and Part III in the Max SOL setting: The key lemma of Grohe and Marx [GM06] says that if the
hypergraph associated with a CSP instance $I$ has fractional edge cover number $\rho$, then the number of solutions to $I$ is bounded by $||I||^\rho$. If the relational vocabulary only contains one relation of arity $k$, it turns out it is possible to express this fractional edge cover number as the separation $s(M, T)$ (for MAX SOL), where $M$ is the CSP instance with uniform weight 1 on the variables; and $T$ is a structure on a $k$-element domain with exactly one $k$-tuple with uniform weight 1 on the domain elements. One important point is that, in the linear programming formulation for $s$, we weight the homomorphisms $T \to M$, which corresponds exactly to the tuples in $M$ (the edges in the constraint hypergraph). The classes of instances with bounded fractional edge cover number can now be seen as subsets of $\{M \mid s(M, T) \leq c\}$, for some fixed constant $c$. 
Bibliography


BIBLIOGRAPHY


BIBLIOGRAPHY


162
Bibliography


Bibliography


### Linköping Studies in Science and Technology

<table>
<thead>
<tr>
<th>No</th>
<th>Title</th>
<th>Year</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Anders Haraldsson: A Program Manipulation System Based on Partial Evaluation</td>
<td>1977</td>
<td>ISBN 91-7372-144-1</td>
</tr>
<tr>
<td>297</td>
<td>Mariam Kamkari: Interprocedural Dynamic Slicing with Applications to Debugging and Testing</td>
<td>1993</td>
<td>ISBN 91-7871-065-0</td>
</tr>
</tbody>
</table>
Linköping Studies in Information Science


