# Complexity of Generalised Colourings of Chordal Graphs

*A depth examination report submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the School of Computing Science*

JURAJ STACHO

M.Sc., Comenius University, Slovakia, 2005

Supervised by Dr. Pavol Hell

December 14, 2007

School of Computing Science, Simon Fraser University, Burnaby, BC, Canada

# Abstract

The graph colouring problem and its derivatives have been notoriously known for their inherent intractability. The difficulty seems to stem from the fact that we want a solution for any given graph, however complex it may be. One of the ways how to overcome this difficulty is to restrict possible inputs to the problem; that is, we ask for a solution only for graphs having some special structure which in turn can be used to efficiently solve the problem.

In this report, we focus on a class of graphs with particularly nice structure, namely chordal graphs. We first survey results about colouring problems, homomorphism problems and partitioning problems in general graphs, as well as, describe useful graph decomposition techniques used in efficient algorithms for these problems. Then we explain our solutions to some cases of partitioning and colouring problems on chordal graphs.

# Contents

Chapter 1

# Introduction

The graph colouring problem has been a focal point of graph theory for over a century. Here the task is to label the vertices of a given graph with colours in such a way that no adjacent vertices receive identical labels. The origins of this problem can be traced probably all the way back to the problem of four colours, as formulated by Francis Guthrie in 1852, where we are asked to colour any map of countries using only four colours in such a way that no two countries sharing a boundary use the same colour. Just like the four colour problem, there are many interesting problems that can be easily recast as a graph colouring problem. Unfortunately the problem itself turns out to be virtually intractable (as long as the number of colours exceeds two). Similarly, by a result of Hell and Nešetřil [65], a generalisation of this problem – the problem of existence of a homomorphism (i.e., a mapping of the vertices of a source graph to the vertices of a target graph that preserves adjacencies) is intractable unless the target graph is fixed and bipartite or has a loop. Finally, as follows from a result of Farrugia [37], a very large class of interesting generalisations of the graph colouring problem is intractable. The difficulty seems to be caused by the generality of the input to the problem. (We ask for a solution to the problem for any graph, however complex it may be.) This is one of the reasons why researchers have been trying to restrict the input graphs so as to obtain tractable versions of the aforementioned problems.

In this report, we investigate an instance of this approach, namely a restriction of the inputs to chordal graphs. We focus on chordal graph particularly in later chapters. First, in Chapters 2 and 3, we survey results about partitioning, colouring, and homomorphism problems in general graphs. Then, in Chapter 4, we describe various graph decomposition methods that are useful for constructing efficient algorithms for colourings and other problems, and in particular we discuss efficient algorithms for graph properties definable in Monadic Second Order logic in graphs of bounded treewidth. Finally, in Chapter 5, we turn our attention to chordal graphs and explain our solutions to some cases of partitioning and colouring problems on chordal graphs.

## 1.1 Definitions

A *graph* $G$ is a pair $(V, E)$ where $V$ is a set of *vertices* (also called a *vertex set*), and $E$, called a set of *edges* (or an *edge set*), is a binary relation on $V$, i.e., $E \subseteq V \times V$. For a graph $G$, the set of vertices and the set of edges of $G$ is denoted by $V(G)$ and $E(G)$ respectively.

A graph $G$ is *undirected* if $E(G)$ is symmetric. In case we want to emphasise that $G$ is not necessarily undirected, we say that $G$ is *directed*, or that $G$ is a *digraph*. A graph $G$ is *loopless*, if $E(G)$ is irreflexive. Note that the definition of graph above does not allow multiple edges between two vertices; in other words, our graphs are what is sometimes reffered to as *simple*.

We say that vertices $u$ and $v$ are *adjacent* in $G$ (or that $u$ and $v$ are *neighbours* in $G$), if $uv \in E(G)$ of $vu \in E(G)$. A walk $W$ in $G$ is a sequence of vertices $W = u_1, u_2, \ldots u_k$ such that for each $1 \leq i < k$, the vertices $u_i$ and $u_{i+1}$ are adjacent. A walk $W = u_1, \ldots, u_k$ in $G$ is *closed* if the vertices $u_1$ and $u_k$ are also adjacent. A (closed) walk $W$ in $G$ is a *path* (respectively a *cycle*) if no vertex of $W$ appears on $W$ more than once. We denote by $P_k$ the path on $k$ vertices with $k-1$ edges, and denote by $C_k$ the cycle on $k$ vertices with $k$ edges. We say that $G$ is *connected* if for any two vertices $u, v$ of $G$ there exists a path that connects them, i.e., a path $u = u_1, u_2, \ldots, u_k = v$. Otherwise, we say that $G$ is *disconnected*. We say that a graph is *acyclic* if it contains no cycle. An undirected acyclic graph is called a *forest*. A connected forest is called a *tree*.

We say that a graph $H$ is a *subgraph* of $G$ and denote it by $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We say that a subgraph $H$ of $G$ is *induced* and denote it by $H \leq G$, if $E(H) = E(G) \cap V(H) \times V(H)$. For any subset $S$ of the vertices of $G$, we denote by $G[S]$ the induced subgraph of $G$ whose vertex set is $S$. We also say that $G[S]$ is a subgraph of $G$ *induced on* $S$. For a subset $S$ of vertices (respectively edges), we denote by $G - S$ the subgraph of $G$ that is obtained by removing from $G$ the vertices (edges) of $S$. (Note that with each removed vertex we must also remove all edges adjacent to it.) In the case that $S$ consists only of a single element $x$, we write $G - x$ instead of $G - \{x\}$.

For a connected graph $G$, a vertex $u$ is a *cutpoint* of $G$ if the graph $G - u$ is disconnected. An edge $e = uv$ is a *bridge* of $G$ if the graph $G - e$ is disconnected. A subset $S$ of vertices (edges) of $G$ is a *vertex (edge) separator* of $G$ if $G - S$ is disconnected.

We say that two graphs $G$ and $H$ are *isomorphic* and denote it by $G \cong H$, if there exists a bijective mapping $f : V(G) \to V(H)$ such that $uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$ for any $u, v \in V(G)$. The *complement* of a graph $G$ is the graph $\overline{G}$ with vertices $V(G)$ and edges $uv$ where $u \neq v$ and $uv \notin E(G)$. The *union* $G \cup H$ of two graphs $G$ and $H$ is the graph with vertices $V(G) \cup V(H)$ and edges $E(G) \cup E(H)$. The *disjoint union* $G \uplus H$ of $G$ and $H$ is the union of graphs $G'$ and $H$, where $G \cong G'$ and $V(G') \cap V(H) = \emptyset$. The *join* $G + H$ of two graphs $G$ and $H$ is the complement of $\overline{G} \uplus \overline{H}$.

A graph $G$ is *complete*, if it contains edges between all pairs of distinct vertices. We denote by $K_n$ the complete graph on $n$ vertices. A *clique* of $G$ is a complete subgraph of $G$, and an *independent set* of $G$ is an induced subgraph of $G$ having no edges. For any graph $G$, we denote by $\omega(G)$, and $\alpha(G)$, the size of a maximum clique in $G$, and the size of a maximum independent set in $G$, respectively.

For a subset $S$ of the vertices of $G$, we denote by $N(S)$ the *open neighbourhood* of $S$, i.e., the set of vertices of $G - S$ that are adjacent to at least one vertex of $S$. We denote by $N[S]$ the *closed neighbourhood* of $S$, i.e., $N[S] = N(S) \cup S$. If $S$ contains only one element $u$, we abbreviate $N(\{u\})$ and $N[\{u\}]$ to $N(u)$ and $N[u]$ respectively. We let $\deg(u) = |N(u)|$ be the *degree* of $u$, and let $\Delta(G)$ be the maximum degree among the vertices of $G$.

Unless indicated otherwise, $n$ always denotes the number of vertices of $G$ and $m$ the number of edges of $G$.

A *colouring* of a graph $G$ is a mapping $c : V(G) \to \mathbb{N}$ where $\mathbb{N}$ is the set of all natural numbers. A *$k$-colouring* of $G$ is a mapping $c : V(G) \to \{1, 2, \ldots, k\}$. A colouring $c$ of $G$ is *proper* if for any $u, v \in V(G)$ we have $c(u) \neq c(v)$ whenever $uv \in E(G)$. The *chromatic number* of $G$, denoted by $\chi(G)$, is the smallest integer $k$ such that $G$ admits a proper $k$-colouring.

A *dominating set* $S$ in $G$ is a subset of vertices of $G$ such that any vertex of $G$ not in $S$ has at least one neighbour in $S$. The domination number of $G$, denoted by $\gamma(G)$, is the size of a smallest dominating set in $G$.

For any two vertices $u, v \in V(G)$, we denote $d(u, v)$ the *distance* between $u$ and $v$, i.e., the length of a shortest path between $u$ and $v$ in $G$. We denote by $G^k$ the *$k$-th power* of $G$, i.e., the graph obtained from $G$ by making adjacent any two vertices in distance at most $k$.

For a graph $H$, we say that $G$ is $H$-free is $G$ contains no induced subgraph isomorphic to $H$. For a set of graphs $\mathcal{H}$, we say that $G$ is $\mathcal{H}$-free is $G$ contains no induced subgraph isomorphic to some $H \in \mathcal{H}$.

## 1.2 Graph classes

In this section we briefly describe some properties of chordal graphs, and also mention other graph classes, in particular those that are important in our later investigations.

### 1.2.1 Chordal Graphs

A graph $G$ is *chordal* if it does not contain an induced subgraph isomorphic to $C_k$ for $k \geq 4$. Chordal graphs have been investigated in the literature also under names *triangulated* or *rigid-circuit* graphs [57]. Chordal graphs have been found to possess several interesting structural properties.

An *elimination ordering* $\pi$ is a numbering of vertices of a graph $G$ from 1 to $n$. The *fill-in* $F_\pi$ caused by the ordering $\pi$ is the set of edges defined as follows.

$$F_\pi = \left\{ uv \;\middle|\; \begin{array}{l} u \neq v,\ uv \notin E(G) \text{ and there exists a } u\text{-}v \text{ path } P \text{ in } G \\ \text{with } \pi(w) < \min\{\pi(u), \pi(v)\} \text{ for all } u, v \neq w \in P \end{array} \right\}$$

An elimination ordering $\pi$ is *perfect* if $F_\pi = \emptyset$. Equivalently, one can say that a total ordering $\prec$ on the vertices of $G$ is a perfect elimination ordering of $G$, if whenever $y$ and $z$ are neighbours of $x$ such that $x \prec y$ and $x \prec z$, we have that $y$ and $z$ are adjacent. An elimination ordering $\pi$ is *minimum* if $|F_\pi|$ is minimum over all orderings of $G$, and is *minimal* if there is no ordering $\sigma$ with $F_\sigma \subsetneqq F_\pi$. The graph $G_\pi = (V, E \cup F_\pi)$ is the *fill-in graph* for $\pi$.

Elimination orderings arise in the study of Gaussian elimination of sparse symmetric matrices [97]. The following properties establish connection between perfect elimination orderings and chordal graphs.

**Proposition 1.1.** [25] *Any choral graph $G$ contains a simplicial vertex, i.e., a vertex whose neighbourhood in $G$ induces a clique in $G$.*

**Proposition 1.2.** [98] *Any ordering $\pi$ is a perfect elimination ordering of $G_\pi$.*

**Proposition 1.3.** [25, 98] *A graph $G$ has a perfect elimination ordering if and only if $G$ is chordal.*

Perfect elimination orderings play an important role in many algorithms for chordal graphs. In particular, it can be shown [57] that using a perfect elimination ordering of a chordal graph $G$, one can compute in time $O(n + m)$ the chromatic number $\chi(G)$ of $G$, the size of a maximum clique $\omega(G)$ of $G$, and also the size of a maximum independent set $\alpha(G)$ of $G$. Note that these problems are intractable in general. It also turns out that for any chordal graph $G$ the chromatic number of $G$ and the size of a maximum clique are equal, i.e., $\chi(G) = \omega(G)$. In particular, this is true for any induced subgraph $H$ of $G$ since $H$ must be also chordal. This shows that chordal graphs are a special subclass of so-called perfect graphs which we discuss in the next section.

Now let $T$ be a tree and $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a set of (connected) subtrees of $T$. The *vertex intersection graph* of $\mathcal{T}$ is a graph $G$ with vertex set $V(G) = \{v_1, \ldots, v_n\}$ in which two vertices $v_i$ and $v_j$ are adjacent if and only if the trees $T_i$ and $T_j$ share a vertex, i.e., $V(T_i) \cap V(T_j) \neq \emptyset$. (A similar class of graphs so-called *edge intersection graphs* of paths in a tree has also been studied [58].) We have the following property.

**Proposition 1.4.** [55] *A graph $G$ is chordal if and only if $G$ is the vertex intersection graphs of subtrees of some tree.*

A family of sets $\{X_1, \ldots, X_n\}$ is said to satisfy the *Helly property* if for any index set $I \subseteq \{1, \ldots, n\}$ with $T_i \cap T_j \neq \emptyset$ for any $i, j \in I$, we have that $\bigcap_{i \in I} T_i \neq \emptyset$. The following is easy to observe.

**Proposition 1.5.** *A family of subtrees of a tree satisfies the Helly property.*

**Proposition 1.6.** [52] *Any chordal graph $G$ on $n$ vertices has at most $n$ maximal cliques.*

Lastly, we have the following property.

**Proposition 1.7.** [57] *Every minimal vertex separator in a chordal graph induces a clique.*

This property allows one to construct a decomposition of any chordal graph using its minimal separators. This is captured by the following notion.

A *clique-tree* $T$ of a connected chordal graph $G$ is a tree such that *(i)* each vertex $v \in V(T)$ corresponds to a maximal clique $C_v$ of $G$, *(ii)* for any edge $xy \in E(G)$, there exists a vertex $v \in V(T)$ with $x, y \in C_v$, and *(iii)* for any vertex $x \in V(G)$, the vertices $v \in V(T)$ with $x \in C_v$ induce a connected subgraph $T_x$ in $T$.

The following property explains a connection between vertex intersection graphs of subtrees of a tree and clique-trees of chordal graphs.

**Proposition 1.8.** *Any chordal graph $G$ is the vertex intersection graph of subtrees $T_x$ of a clique-tree $T$ of $G$.*

Finally, we establish a connection between perfect elimination orderings and clique-trees of chordal graphs.

Let $T$ be a fixed clique-tree of a chordal graph $G$. Let us consider $T$ rooted at a vertex $r$ chosen arbitrarily, and let $\sqsubset$ be a strict partial order on the vertices of $T$ defined as follows: $u \sqsubset v \iff u$ is a descendant of $v$. Based on this, one can easily observe that the vertex set of any (connected) subgraph of $T$ has a (unique) maximal element with respect to $\sqsubset$. Hence, for any vertex $x \in V(G)$, let $m_x$ be the unique maximal element of $T_x$ (the connected subgraph of $T$ formed by vertices $v \in V(T)$ with $x \in C_v$) with respect to $\sqsubset$.

Now let $\prec$ be a strict partial order on the vertices of $G$ defined as follows: $x \prec y \iff m_x \sqsubset m_y$. It is easy to see from the definition that the following observation holds.

**Proposition 1.9.** *Any linear extension of $\prec$ is a perfect elimination ordering of $G$.*

(This useful fact appears to not have been explicitly observed previously.)

Now we introduce a fundamental concept for exploring the structure of a graph. *Lexicographic breadth-first search* is an algorithm that, given a graph $G$, constructs a special breadth-first search ordering of the vertices of $G$. In the process of exploring the graph, the algorithm assigns to the vertices labels formed by subsets of $\{1, \dots, n\}$, and using these labels it decides which vertex to explore next. The elements of any label are assumed to be ordered from the largest to the smallest number in the label. The algorithm always chooses an unprocessed vertex with lexicographically largest label among the unprocessed vertices (ties are broken arbitrarily), where lexicographic order is just the usual dictionary order, e.g., $\{9, 7, 6, 1\} < \{9, 8, 5\}$ and $\{6, 4, 3\} < \{6, 4, 3, 2\}$. The algorithm as just described is summarised in Algorithm 1.

We have the following property about the algorithm.

**Proposition 1.10.** [57] *A graph $G$ is chordal if and only if Algorithm 1 on $G$ produces a perfect elimination ordering $\pi$ of $G$.*

Now it follows that one can decide in time $O(n + m)$ whether a given graph $G$ is chordal just by computing an elimination ordering $\pi$ using Algorithm 1 on $G$, and then testing whether $\pi$ is a perfect elimination ordering by simply checking whether the neighbours of any vertex $v$ that appear in $\pi$ after $v$ form a clique; both steps can easily be implemented in time $O(n+m)$.

We remark that Lexicographic breadth-first search algorithm was originally introduced by Rose, Tarjan and Leuker in [98] as a simple linear time algorithm for recognising chordal graphs (as we just described). It has since found numerous applications outside chordal graphs, for instance, efficient recognition of cographs ($P_4$-free graphs), $P_4$-sparse graphs, $P_4$-reducible graphs, AT-free graphs, interval graphs, unit interval graphs, and their powers [20].

We should mention that there exists yet another efficient algorithm by Tarjan [111] for recognition of chordal graph. *Maximum cardinality search* is an algorithm that numbers vertices of a graph from $n$ to 1 where the next vertex to be numbered it one that is adjacent to the most numbered vertices (ties are broken arbitrarily). This algorithm also produces a perfect elimination ordering given a chordal graph, yet it should be pointed out that the orderings produced by Maximum cardinality search and the orderings produced by Lexicographic breadth-first search are not exactly the same, and moreover there are perfect elimination orderings that none of these algorithms can produce.

We close by mentioning that recently a unified approach to graph search algorithms generalising both the above algorithms has been discovered [22].

Algorithm 1: Lexicographic Breadth-First search

```
Input: A graph $G$
Output: An elimination ordering $\pi$
set $label(v) \leftarrow \emptyset$ for all $v \in V(G)$
for $i \leftarrow n$ downto 1 do
    pick an unnumbered vertex $v$ with lexicographically largest label
    $\pi(i) \leftarrow v$ /* the vertex $v$ becomes numbered */
    for each unnumbered $w$ adjacent to $v$ do
        add $i$ to $label(w)$
```

### 1.2.2 Perfect Graphs

A graph $G$ is *perfect* if, for every induced subgraph $H$ of $G$, the chromatic number $\chi(H)$ is equal to the clique number $\omega(H)$. The class of perfect graphs includes such classes of graphs as bipartite graphs, chordal graphs, cographs, comparability graphs, and their complements. It is an interesting and important class of graphs, and has been a focus of attention for more than a half of a century now, which is in part due to the result of Grötschel, Lovász and Schrijver [59] who gave a polynomial time algorithm for all the basic combinatorial problems ($\chi$,$\alpha$,$\omega$) in perfect graphs. There are many interesting results known about perfect graphs. Here we only mention the most notable ones conjectured by Berge [6], namely The Weak Perfect Graph Theorem due to Lovász [24] and The Strong Perfect Graph Theorem due to Chudnovsky, Robertson, Seymour and Thomas [15].

**Theorem 1.11 (The Weak Perfect Graph Theorem).** [24]
*A graph is perfect if and only its complement if also perfect.*

**Theorem 1.12 (The Strong Perfect Graph Theorem).** [15]
*A graph is perfect if and only if it has no induced odd cycle or its complement.*

As we already remarked, some problems that are difficult in general like the graph colouring or the maximum clique problem, admit a polynomial time solution in perfect graphs. Unfortunately, the algorithms for these problems are not always combinatorial, and in addition, there are problems that are not tractable even in perfect graphs. It is therefore natural to ask in which restricted classes of perfect graphs these problems have nice combinatorial solutions and perhaps a polynomial solutions for problems intractable in perfect graphs. We already mentioned such a subclass, namely the class of chordal graphs in which many difficult problems admit even linear time algorithms.

In the following, we briefly describe some other interesting examples of such classes, and summarise the complexities of selected combinatorial problems at the end of this chapter.

#### Split graphs

A graph $G$ is *split* if $G$ can be partitioned into a clique and an independent set with no other restriction on the edges between the two. It can be seen that any split graph is also chordal, and since the complement of a split graph is also split, any split graph is also co-chordal (a graph is *co-chordal* if it is the complement of a chordal graph). In fact, the converse is also true.

**Theorem 1.13.** [57] *A graph $G$ is split if and only if $G$ is both chordal and co-chordal.*

## Comparability graphs

A subset of edges $F \subseteq E(G)$ of an undirected graph $G$ is an *orientation* of $G$ if $F \cap F^{-1} = \emptyset$ and $F \cup F^{-1} = E(G)$. (The set $F^{-1}$ consists of edges $vu$ where $uv \in F$).

A graph $G$ is a *comparability graph* if there exists a transitive orientation of $G$, that is, an orientation $F$ of $G$ with $F^2 \subseteq F$ where $F^2 = \{ac \mid ab, bc \in F \text{ for some } b\}$. We remark that basic facts about comparability graphs go all the way back to the paper of Gallai [54].

## Permutation graphs

For a permutation $\pi$ of $\{1,\ldots,n\}$, let $G[\pi]$ be a graph with vertices $V(G[\pi]) = \{1,\ldots,n\}$ and edges $ij \in E(G[\pi]) \iff (i-j)\big(\pi^{-1}(i) - \pi^{-1}(j)\big) < 0$.

A graph $G$ is called a *permutation graph* if $G \cong G[\pi]$ for some permutation $\pi$. It can be shown that any permutation graph is also a comparability graph. Moreover, one can observe that the complement of a permutation graph $G = G[\pi]$ is the graph $G[\pi^R]$ where $\pi^R$ is the reverse of $\pi$, i.e., $\pi^R(i) = \pi(n+1-i)$ for all $1 \le i \le n$. Hence, $\overline{G}$ is a permutation graph and it follows that any permutation graph is also a co-comparability graph (a graph is *co-comparability* if it is the complement of a comparability graph). In fact, it can be shown that, conversely, any graph that is both a comparability and a co-comparability graph is also a permutation graph.

**Theorem 1.14.** [57] *A graph $G$ is a permutation graph if and only if $G$ is both a comparability and co-comparability graph.*

## Cographs

A graph $G$ is a *cograph* if $G$ can be constructed from single vertex graphs using the operations of join and union (or equivalently the operations of complement and union). This construction can be represented by a tree whose leaves are the vertices of $G$, and inner nodes are labeled either 0 or 1, denoting the operation of union or join respectively. It can be shown [21] that for any cograph $G$ there exists a unique minimal tree that represents the construction of $G$; such a tree is called the *cotree* of $G$. We remark that using the cotree of a cograph $G$, one can solve many graph problems on $G$ efficiently and usually in linear time.

From a structural point of view, in [21] it was shown that the class of cographs is precisely the class of $P_4$-free graph.

**Theorem 1.15.** [21] *A graph $G$ is a cograph if and only if $G$ contains no induced $P_4$.*

Additionally, the authors in [21] show a multitude of equivalent definitions of cographs that independently appeared in the literature. In particular, it can be seen that cographs form a proper subclass of permutation graphs.

#### Interval graphs

For a family of intervals $\mathcal{I} = \{I_1, \ldots, I_n\}$ on the real line, the intersection graph of $\mathcal{I}$ is the graph with vertices $\{v_1, \ldots, v_n\}$ and edges $v_iv_j \iff I_i \cap I_j \neq \emptyset$.

A graph $G$ is an *interval graph* if $G$ is the intersection graph of a family of intervals on the real line. It can be seen that any interval graph is also a chordal graph. Similarly, it is not difficult to argue [57] that any interval graph is also a co-comparability graph, and conversely, any graph that is chordal and co-comparability is also an interval graph.

**Theorem 1.16.** [57] *A graph $G$ is interval if and only if $G$ is both chordal and co-comparability.*

By a result of Lekkerkerker and Boland [84], we have the following characterisation of interval graphs. An *asteroidal triple* of a graph $G$ is a triple of mutually non-adjacent vertices such that for any two vertices of the triple there exists a path in $G$ that avoids the neighbourhood of the third vertex in the triple. We say that a graph $G$ is *AT-free* if $G$ does not contain any asteroidal triple.

**Theorem 1.17.** [84] *A graph $G$ is interval if and only if $G$ is chordal and contains no asteroidal triple (is AT-free).*

We remark that AT-free graphs form a class on their own, and even though they are not a subclass of perfect graphs, several interesting results are known about them [102].

### 1.2.3 Other classes

In this section, we briefly define other classes of graphs we shall deal with in later chapters.

#### Circular-Arc graphs

A graph $G$ is a *circular-arc graph* if $G$ is the intersection graph of arcs of a circle. Circular-arc graphs are not necessarily perfect, e.g., $C_5$ is circular-arc.

#### Bi-arc graphs

A graph $G$ is a *bi-arc graph* if there exist a family of arcs of a circle with two distinguished points $p$ and $q$ where each vertex of $G$ is associated with two arcs $(N_x, S_x)$ such that $N_x$ contains $p$ but not $q$ and $S_x$ contains $q$ but not $p$, and for any two vertices $x, y$ the following holds: $(i)$ if $xy \in E(G)$ then $N_x \cap S_y = \emptyset$ and $N_y \cap S_x = \emptyset$, and $(ii)$ if $xy \notin E(G)$ then $N_x \cap S_y \neq \emptyset$ and $N_y \cap S_x \neq \emptyset$. Note that it is not possible that $N_x \cap S_y = \emptyset$ and $N_y \cap S_x \neq \emptyset$.

### 1.2.4 Complexity Results

In table 1.1 in this section we summarise complexities of the graph colouring and related problems in different graph classes. The results are taken from [102]. Here, $n$ and $m$, as usual, refer to the number of vertices respectively edges of an input graph, and $n^\alpha$ is the complexity of matrix multiplication.

| | recognition | $\chi(G)$ | $\alpha(G)$ | $\gamma(G)$ | $HAM$ |
|---|---|---|---|---|---|
| All graphs | - | $NP$c | $NP$c | $NP$c | $NP$c |
| Perfect | $O(n^9)$ | $\in P$ | $\in P$ | $NP$c | $NP$c |
| Chordal | $O(n+m)$ | $O(n+m)$ | $O(n+m)$ | $NP$c | $NP$c |
| Split | $O(n+m)$ | $O(n+m)$ | $O(n+m)$ | $NP$c | $NP$c |
| AT-free | $O(n^\alpha)$ | *open* | $O(n^3)$ | $O(n^6)$ | *open* |
| Interval | $O(n+m)$ | $O(n)$ | $O(n)$ | $O(n)$ | $O(n)$ |
| Circular-arc | $O(n+m)$ | $NP$c | $O(n)$ | $O(n)$ | $O(n^2 \log n)$ |
| Comparability | $O(n^\alpha)$ | $O(n+m)$ | $O(n^2m)$ | $NP$c | $NP$c |
| Co-comparability | $O(n^\alpha)$ | $O(n^2m)$ | $O(n+m)$ | $O(nm^2)$ | $O(n^3)$ |
| Permutation | $O(n+m)$ | $O(n \log\log n)$ | $O(n \log\log n)$ | $O(n)$ | $O(n+m)$ |
| Cographs | $O(n+m)$ | $O(n)$ | $O(n)$ | $O(n)$ | $O(n)$ |

**Table 1.1**: A summary of complexities of selected graph problems in different graph classes. Note that all problems except the recognition assume that an appropriate representation of a graph is given, e.g., a set of intervals for an interval graph.

Chapter 2

# Graph Properties

In this chapter, we investigate complexity of generalised colouring problems cast as colourings by graph properties. We will mostly follow the notation used in [37].

For a set $\mathcal{U}$, a *property* $\mathcal{P}$ of $\mathcal{U}$ is a subset of $\mathcal{U}$. A property $\mathcal{P}$ is *trivial* if $\mathcal{P} = \emptyset$ or $\mathcal{P} = \mathcal{U}$; otherwise $\mathcal{P}$ is *non-trivial.* A *graph property* $\mathcal{P}$ is an isomorphism closed property of the set of all graphs, i.e., for each $G \in \mathcal{P}$, any $G'$ isomorphic to $G$ is also in $\mathcal{P}$. Unless indicated otherwise, we shall assume that the properties we consider are non-trivial.

## 2.1 Closure properties

We say that $\mathcal{P}$, a property of $\mathcal{U}$, is *closed* under a $(k+1)$-ary relation $R \subseteq \mathcal{U} \times \mathcal{U}^k$, $k \geq 0$, if, whenever $(s, t_1, t_2, \ldots, t_k) \in R$ and $t_1, t_2, \ldots t_k \in \mathcal{P}$, then also $s \in \mathcal{P}$. For instance, by definition any graph property $\mathcal{P}$ is closed under isomorphism, i.e., the relation $\alpha(G_1, G_2) \equiv$ "$G_1$ is isomorphic to $G_2$". We say that a graph property $\mathcal{P}$ is *additive*, if it is closed under taking disjoint unions of graphs, i.e., the relation $\uplus(G, G_1, G_2) \equiv$ "$G$ is the disjoint union of $G_1$ and $G_2$"; we say that $\mathcal{P}$ is *co-additive* if it is closed under join.

For a partial order $\preceq$ on the set $\mathcal{U}$, we say that a property $\mathcal{P}$ is *$\preceq$-hereditary*, if $\mathcal{P}$ is closed under $\preceq$. In particular, for the case of graphs, we shall be interested in two partial orders on graphs, namely the subgraph order $\subseteq$ and the induced subgraph order $\leq$. We shall say that a graph property $\mathcal{P}$ is *hereditary*, respectively *induced-hereditary*, if $\mathcal{P}$ is $\subseteq$-hereditary, respectively $\leq$-hereditary. We denote by $\mathbb{L}$, $\mathbb{L}_{\leq}$, $\mathbb{L}^a$, and $\mathbb{L}^a_{\leq}$ the class of hereditary, induced hereditary, additive hereditary and additive induced hereditary properties respectively.

We now briefly mention some examples of hereditary graph properties. In chapter 3, we define the problem $HOM(H)$ of existence of a homomorphism from a given graph $G$ to the graph $H$. This problem clearly defines a graph property $(\to H) = \{G \mid G \to H\}$, that is, $(\to H)$ contains the graphs $G$ homomorphic to $H$. It is easy to see that for any $H$ the property $(\rightarrow H)$ is additive and hereditary, hence also induced hereditary. Below, we list more examples of (induced) hereditary graph properties.

$$\begin{array}{ccc}
\mathcal{O} = \{\overline{K}_r \mid r \geq 0\} & \mathcal{K} = \{K_r \mid r \geq 0\} & \mathcal{S}_k = \{G \mid \Delta(G) \leq k\} \\
\mathcal{P}_k = \{G \mid P_k \not\leq G\} & \mathcal{K}_k = \{G \mid K_k \not\leq G\} & H_{\subseteq} = \{G \mid G \subseteq H\} \\
\overline{\mathcal{P}}_k = \{G \mid \overline{P}_k \not\leq G\} & \overline{\mathcal{K}}_k = \{G \mid \overline{K}_k \not\leq G\} & H_{\leq} = \{G \mid G \leq H\} \\
(\rightarrow H) = \{G \mid G \rightarrow H\} & \mathcal{B} = \{\text{perfect graphs}\} & \mathcal{X} = \{\text{chordal graphs}\}
\end{array}$$

The properties $\mathcal{O}, \mathcal{S}_k, \mathcal{K}_k, (\rightarrow H), H_{\subseteq}$ are all hereditary, whereas $\mathcal{K}_r, \mathcal{P}_k, \overline{\mathcal{P}}_k, \overline{\mathcal{K}}_k, H_{\leq}, \mathcal{B}, \mathcal{X}$ are only induced hereditary and not hereditary. Similarly the properties $\mathcal{O}, \mathcal{S}_k, \mathcal{P}_k, \overline{\mathcal{P}}_k, \mathcal{K}_k, \overline{\mathcal{K}}_k$, $(\rightarrow H)$, $\mathcal{B}, \mathcal{X}$ are additive, whereas $\mathcal{K}, H_{\subseteq}, H_{\leq}$ are not. Furthermore, $\mathcal{K}, \mathcal{B}, \overline{\mathcal{P}}_k, \overline{\mathcal{K}}_k$ and $\mathcal{P}_k$ for $k \geq 4$ are the only co-additive properties from the list above.

An element $s \in \mathcal{U}$ is a *$\preceq$-predecessor* of $t \in \mathcal{U}$ if $s \preceq t$; if moreover $s \neq t$, then $s$ is a *proper* $\preceq$-predecessor of $t$. For a property $\mathcal{P}$, a *minimal forbidden* $\preceq$-predecessor is an element $t \notin \mathcal{P}$ such that each proper $\preceq$-predecessor of $t$ is in $\mathcal{P}$. We denote $\mathcal{F}_{\preceq}(\mathcal{P})$ the set of all minimal forbidden $\preceq$-predecessors for $\mathcal{P}$, i.e., $\mathcal{F}_{\preceq}(\mathcal{P}) = \min_{\preceq}\{t \mid t \notin \mathcal{P}\}$.

For a set $\mathcal{S} \subseteq \mathcal{U}$, we define the property $\mathrm{Forb}_{\preceq}(\mathcal{S}) = \{t \mid \forall s \in \mathcal{S},\ s \not\preceq t\}$. If $\mathcal{S}$ is a single element $s$, we write $\mathrm{Forb}_{\preceq}(s)$ instead of $\mathrm{Forb}_{\preceq}(\{s\})$. For instance, we have $\mathrm{Forb}_{\leq}(P_k) = \mathcal{P}_k$ and $\mathrm{Forb}_{\subseteq}(K_k) = \mathcal{K}_k$, where $\mathcal{F}_{\leq}(\mathcal{P}_k) = \{P_k\}$ and $\mathcal{F}_{\subseteq}(\mathcal{K}_k) = \{K_k\}$. In general, if $\mathcal{S}$ is an antichain under $\preceq$, then $\mathcal{F}_{\preceq}(\mathrm{Forb}_{\preceq}(\mathcal{S})) = \mathcal{S}$.

Note that, if $\preceq$ is the minor relation (a graph $H$ is a *minor* of $G$, $H \preceq G$, if $H$ can be obtained from $G$ by a series of edge contractions and edge removals), then by The Graph Minor Theorem of Robertson and Seymour [96], any antichain under $\preceq$ is finite; hence $\mathcal{F}_{\preceq}(\mathcal{P})$ is finite for any minor closed graph property $\mathcal{P}$.

## 2.2 Composition of properties

For two properties $\mathcal{P}, \mathcal{Q}$ the property $\mathcal{P} \circ \mathcal{Q}$ contains all graphs $G$ whose vertex set $V(G)$ can be partitioned into sets $X, Y$ where $G[X] \in \mathcal{P}$ and $G[Y] \in \mathcal{Q}$. We shall call the property $\mathcal{P} \circ \mathcal{Q}$ the *composition* of the properties $\mathcal{P}$ and $\mathcal{Q}$. More generally, for properties $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$, the property $\mathcal{Q}_1 \circ \ldots \circ \mathcal{Q}_k$ consists of all graphs $G$ whose vertex set $V(G)$ can be partitioned into sets $V_1, \ldots, V_k$ where $G[V_i] \in \mathcal{Q}_i$ for all $1 \leq i \leq k$. If $\mathcal{P} = \mathcal{Q}_1 \circ \ldots \circ \mathcal{Q}_k$, then each $\mathcal{Q}_i$ is a *factor* or *divisor* of $\mathcal{P}$, while $\mathcal{P}$ is their *product*. For a property $\mathcal{R}$, we usually abbreviate

$\mathcal{R} \circ \ldots \circ \mathcal{R}$ to $\mathcal{R}^k$. For a class of graph properties $\mathbb{P}$, we say that a property $\mathcal{P}$ is *reducible* or *factorisable* over $\mathbb{P}$, if there exist properties $\mathcal{R}, \mathcal{Q} \in \mathbb{P}$ such that $\mathcal{P} = \mathcal{R} \circ \mathcal{Q}$. Otherwise, we say that $\mathcal{P}$ is *irreducible* over $\mathbb{P}$.

## 2.3 Uniqueness of factorisation

For graphs $G_1, G_2$ and a partial graph ordering $\preceq$, a *(disjoint) $\preceq$-composition* of $G_1$ and $G_2$ is a graph $H$ containing (vertex disjoint) subgraphs $H_1$ and $H_2$ where $G_1 \preceq H_1$ and $G_2 \preceq H_2$.

We say that a hereditary graph property $\mathcal{P}$ is *hereditary (disjoint) compositive* if $\mathcal{P}$ is closed under (disjoint) $\subseteq$-composition. We say that an induced-hereditary graph property $\mathcal{P}$ is *induced-hereditary (disjoint) compositive* if $\mathcal{P}$ is closed under (disjoint) $\leq$-composition.

It turns out that any (additive) hereditary compositive property has a factorisation into (additive) hereditary compositive properties which is unique, up to the order of factors; similarly for (additive) induced hereditary disjoint compositive properties.

**Theorem 2.1.** [37] *An (induced) hereditary (disjoint) compositive property has a unique factorisation into irreducible (induced) hereditary (disjoint) compositive factors; an additive (induced) hereditary property has a unique factorisation into irreducible additive (induced) hereditary factors.*

The proof of this theorem is quite non-trivial and utilises partial results spanning several papers [5, 35, 36, 76, 90, 91, 92]. The result itself serves as a tool in characterising uniquely partitionable graphs which are used in the complexity characterisation which we discuss in the next section.

## 2.4 Complexity

For properties $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$, the problem of deciding, given a graph $G$, whether $G \in \mathcal{Q}_1 \circ \ldots \circ \mathcal{Q}_k$ is called a $(\mathcal{Q}_1, \ldots, \mathcal{Q}_k)$-COLOURING or $(\mathcal{Q}_1, \ldots, \mathcal{Q}_k)$-PARTITIONING or $(\mathcal{Q}_1 \circ \ldots \circ \mathcal{Q}_k)$-RECOGNITION problem. For instance the problem $\mathcal{O}^k$-RECOGNITION is the well known proper $k$-colouring problem.

Interestingly, the problem of $\mathcal{P}$-RECOGNITION can be arbitrarily hard, even when $\mathcal{P}$ is additive and induced-hereditary [37]. (Note that this does not happen with minor closed properties $\mathcal{P}$ since then $\mathcal{F}_{\preceq}(\mathcal{P})$ is always finite [96].) On the other hand, if both $\mathcal{P}$ and $\mathcal{Q}$ are in $NP$ then their composition $\mathcal{P} \circ \mathcal{Q}$ is also in $NP$, whereas if both $\mathcal{P}$ and $\mathcal{Q}$ are in $P$, then $\mathcal{P} \circ \mathcal{Q}$ can be $NP$-complete. It turns out [34, 37] that for reducible additive induced-hereditary properties each problem is either in $P$ or $NP$-hard.

**Theorem 2.2.** [34, 37] *Let $\mathcal{P}$ and $\mathcal{Q}$ be additive induced-hereditary properties, $\mathcal{P} \circ \mathcal{Q} \neq \mathcal{O}^2$. Then $(\mathcal{P} \circ \mathcal{Q})$-RECOGNITION is $NP$-hard. Moreover, it is $NP$-complete iff $\mathcal{P}$- and $\mathcal{Q}$-RECOGNITION are both in $NP$.*

## 2.5 Mixed Properties

For a property $\mathcal{Q}$, we define the property $\overline{\mathcal{Q}}$ as the graphwise complement of $\mathcal{Q}$, i.e., $\overline{\mathcal{Q}} = \{\overline{G} \mid G \in \mathcal{Q}\}$. Unfortunately, not all interesting graph properties can be expressed as a composition of additive properties. This is for instance the case for the class of polar graphs. (A graph $G$ is *polar* if $G$ can be partitioned into a complete multipartite graph and a disjoint union of cliques.). However, some graph properties can be expressed as a composition $\mathcal{P} \circ \overline{\mathcal{Q}}$ where $\mathcal{P}$ and $\mathcal{Q}$ are additive properties. For example, polar graphs precisely correspond to the property $\mathcal{P}_3 \circ \overline{\mathcal{P}}_3$. It is easy to see that a property $\mathcal{Q}$ is additive induced-hereditary if and only if $\overline{\mathcal{Q}}$ is co-additive induced-hereditary. Note that similar claim does not hold for hereditary properties. In [37], the following complexity result is proved.

**Theorem 2.3.** [37] *Let $\mathcal{P}$ be a composition of an additive and a co-additive induced hereditary property. Then $\mathcal{P}$-RECOGNITION is $NP$-hard except, possibly, if there exist efficiently recognisable irreducible properties $\mathcal{Q}$ and $\mathcal{R}$ where $\mathcal{Q}$ is additive and $\mathcal{R}$ is co-additive such that $\mathcal{P}$ is $\mathcal{Q}, \mathcal{R}, \mathcal{Q} \circ \mathcal{R}$, $\mathcal{Q} \circ \mathcal{K}^2$, $\mathcal{O}^2 \circ \mathcal{R}$, or $\mathcal{O}^2 \circ \mathcal{K}^2$.*

On the other hand, it is known that $(\mathcal{O} \circ \mathcal{K})$-, $(\mathcal{O}^2 \circ \mathcal{K})$-, $(\mathcal{O} \circ \mathcal{K}^2)$- and $(\mathcal{O}^2 \circ \mathcal{K}^2)$-RECOGNITION are polynomial time solvable [46], whereas the complexity of the cases $\mathcal{Q} \circ \mathcal{R}$, $\mathcal{Q} \circ \mathcal{K}^2$ and $\mathcal{O}^2 \circ \mathcal{R}$ is largely unknown. However, for hereditary properties there is a complete characterisation due to Alekseev and Lozin [3].

**Theorem 2.4.** [3] *For any properties $\mathcal{P}$ and $\mathcal{Q}$ where $\mathcal{P} \subseteq \mathrm{Forb}_{\leq}(K_p)$, and $\mathcal{Q} \subseteq \mathrm{Forb}_{\leq}(\overline{K_q})$ for some integers $p, q$, there exists a polynomial time algorithm for $(\mathcal{P} \circ \mathcal{Q})$-RECOGNITION.*

The running time of this algorithm is roughly $O(n^{2R(p,q)})$ where $R(p,q)$ is the Ramsey number of $p$ and $q$. It should be noted that this result is a particular instance of a more general approach called sparse-dense partitions [46] which is described in Section 3.3.1 of Chapter 3. Finally, since each hereditary property must be $K_p$-free for some $p$, unless it contains all graphs, we have the following consequence.

**Theorem 2.5.** [3] *For any additive hereditary properties $\mathcal{P}$ and $\mathcal{Q}$, the problem of $(\mathcal{P} \circ \overline{\mathcal{Q}})$-*RECOGNITION *is NP-hard, unless both $\mathcal{P}$ and $\mathcal{Q}$ are in P, in which case it is polynomial time solvable.*

## 2.6 Properties of graph classes

Finally, in the light of results from the previous sections, we can naturally ask what is the complexity of $\mathcal{P}$-RECOGNITION when restricted to a particular class of graphs. That is, for a graph property $\mathcal{P}$ and a class of graphs $\mathcal{D}$, we can consider the problem of $\mathcal{P}$-RECOGNITION on instances from $\mathcal{D}$. In case $\mathcal{D}$ is efficiently recognisable, we can equivalently consider the problem of $\mathcal{D} \cap \mathcal{P}$-RECOGNITION. Notice that unfortunately Theorem 2.2 does not apply here, since for properties $\mathcal{P}$ and $\mathcal{Q}$, we do not necessarily have $\mathcal{D} \cap (\mathcal{P} \circ \mathcal{Q}) = (\mathcal{D} \cap \mathcal{P}) \circ (\mathcal{D} \cap \mathcal{Q})$. Generally, the complexity characterisation like Theorem 2.2 in this case is largely unknown even for nicely structured graph classes $\mathcal{D}$.

Chapter 3

# Graph Homomorphisms

In this chapter, we investigate the complexity of graph homomorphism properties which form a significant subclass of all additive hereditary properties. We explain the connection with Constraint Satisfaction and well known Dichotomy Conjecture.

## 3.1 Constraint Satisfaction

A *vocabulary* is a set $\sigma = \{(R_1, k_1), \ldots, (R_t, k_t)\}$ of relation names and their arities. A *relational structure* $\mathcal{H}$ over the vocabulary $\sigma$ is a tuple $\mathcal{H} = (V, R_1, \ldots, R_t)$ where $V$ is a set and $R_i \subseteq V^{k_i}$ for all $1 \leq i \leq t$. For two relational structures $\mathcal{G}$ and $\mathcal{H}$ over the same vocabulary, a *homomorphism* from $\mathcal{G}$ to $\mathcal{H}$ is a mapping $h : V(\mathcal{G}) \to V(\mathcal{H})$ where for any $1 \leq i \leq t$, whenever $(x_1, \ldots, x_{k_i}) \in R_i(\mathcal{G})$, then also $(h(x_1), \ldots, h(x_{k_i})) \in R_i(\mathcal{H})$. Note that (di)graphs are precisely relational structures over the vocabulary $\{(E, 2)\}$.

A *constraint satisfaction problem* is the problem of deciding, given two relational structures $\mathcal{G}$ and $\mathcal{H}$ over the same vocabulary (with possibly $\mathcal{G}$ or $\mathcal{H}$ fixed), whether there exists a homomorphism $h$ from $\mathcal{G}$ to $\mathcal{H}$. In case $h$ exists, we also say that $\mathcal{G}$ is homomorphic to $\mathcal{H}$ and write $\mathcal{G} \to \mathcal{H}$, otherwise we write $\mathcal{G} \not\to \mathcal{H}$. We observe that any constraint satisfaction problem is in $NP$, since given a mapping $h$, one can easily check whether it is a homomorphism from $\mathcal{G}$ to $\mathcal{H}$, and the description of $h$ is clearly polynomial in size. In case $\mathcal{H}$ is fixed, we denote such problem $CSP(\mathcal{H})$. If $\mathcal{H}$ is a (directed) graph $H$, we also call it *$H$-colouring problem*, and denote it $HOM(H)$. The structure $\mathcal{G}$ is usually called *an instance* or *a source*, and $\mathcal{H}$ is usually called *a template* or *a target*. The *$H$-graph retract* problem $RET(H)$ is a problem of deciding, given a (di)graph $G$ containing a fixed copy $H'$ of $H$, whether there exists a homomorphism from $G$ to $H$ that bijectively maps $H'$ to $H$. It is a particular example of a constraint satisfaction problem. We say that a graph $G$ is a *core* if $G$ is not homomorphic to any proper induced subgraph of $G$, or equivalently, $G$ cannot be retracted to some proper induced subgraph of $G$. We call $CSP$ the class of problems $CSP(\mathcal{H})$, and similarly call $HOM$ respectively $RET$ the class of problems $HOM(H)$ respectively $RET(H)$.

The class $CSP$ is a very poweful and rich class of problems. There are countless examples of problems expressible as a $CSP$. For instance, we find uses of $CSP$ in machine vision, belief maintenance, scheduling, temporal reasoning, type reconstruction, graph theory and satisfiability [28, 49, 77, 89].

For two decision problems $A$ and $B$, we say that $A$ is *polynomially reducible* to $B$ if there exists a polynomially computable function $f$ transforming instances of $A$ to instances of $B$ such that $X \in A \iff f(X) \in B$ for any instance $X$ of $A$. We say that $A$ and $B$ are *polynomially equivalent* if $A$ is polynomially reducible to $B$ and $B$ is polynomially reducible to $A$. For two classes of decision problems $\mathcal{A}$ and $\mathcal{B}$, we say that $\mathcal{A}$ is *polynomially equivalent* to $\mathcal{B}$, if for any problem $A \in \mathcal{A}$ there exists a polynomially equivalent problem $B_A \in \mathcal{B}$, and for any problem $B \in \mathcal{B}$ there exists a polynomially equivalent problem $A_B \in \mathcal{A}$

### 3.1.1 Dichotomy Conjecture

Constraint satisfaction problems have recently gained considerable interest due to the famous paper of Feder and Vardi [49]. The result of Ladner [80] says that assuming $P \neq NP$, there must exist problems that are neither in $P$ nor $NP$-complete, also called *intermediate problems.* In the light of this result, Feder and Vardi investigated which subclasses of $NP$ have the same computational power as $NP$, and which do not (and hence might not contain intermediate problems). They define the class $MMSNP$, monotone monadic strict $NP$ without inequality, and show that for this class Ladner's argument does not immediately apply, however, removing either of 'monotone', 'monadic' or 'without inequality' restrictions gives the full computational power of $NP$. Furthermore, they show that $MMSNP$ is polynomial time equivalent to the class $CSP$, where the reduction from $MMSNP$ to $CSP$ is deterministic, whereas the reduction from $CSP$ to $MMSNP$ is randomised. (Later Gabor Kun [78] derandomised their construction.) Hence they conjectured the following.

**Conjecture 3.1 (Dichotomy Conjecture for CSP).**
*Every problem in the class $CSP$ is either in $P$ or $NP$-complete.*

It should be mentioned that this dichotomy conjecture was largely motivated by a result of Shaeffer [100], who showed a dichotomy for boolean $CSP$, and a result of Hell and Nešetřil, who showed a dichotomy for undirected graphs. (See Theorem 3.11.)

Feder and Vardi in [49] also investigated which classes of $NP$ problems have the same computational power as $CSP$, and in particular showed the following.

**Theorem 3.2.** [49] *$CSP$ is polynomially equivalent to the classes $RET$ and $HOM$.*

This implies that showing a dichotomy for either of $RET$ or $HOM$ implies dichotomy for $CSP$. Hence, the following conjectures.

**Conjecture 3.3** (**Dichotomy Conjecture for (di)graph homomorphism problem).**
*Every problem in the class $HOM$ is either in $P$ or $NP$-complete.*

**Conjecture 3.4** (**Dichotomy Conjecture for graph retract problem).**
*Every problem in the class $RET$ is either in $P$ or $NP$-complete.*

The last problem we are going to deal with that has a connection to the conjecture is the following. The *list constraint satisfaction problem* for a relational structure $\mathcal{H}$, denoted by $LCSP(\mathcal{H})$, is the problem of deciding, given a relational structure $\mathcal{G}$ with lists $\ell(v) \subseteq V(\mathcal{H})$ for any $v \in V(\mathcal{G})$, whether there exists a homomorphism $h$ from $\mathcal{G}$ to $\mathcal{H}$ with $h(v) \in \ell(v)$ for any $v \in V(\mathcal{G})$. Such homomorphism $h$ is called a *list homomorphism.* We denote $LCSP$ the class of problems $LCSP(\mathcal{H})$. For (directed) graph $H$, we write $LHOM(H)$ for the problem $LCSP(H)$, and denote $LHOM$ the class of problems $LHOM(H)$. (Note that here we deviate from the notation used in [10] by using $LHOM$ strictly for (di)graph list homomorphism, and using $LCSP$ in place of $LHOM$ for structures.)

In the literature [10, 64], list $CSP$'s are also called *conservative* $CSP$'s or $CSP$'s for *conservative structures.* (A structure $\mathcal{H}$ is *conservative* if $\mathcal{H}$ contains all unary (arity one) relations $U_S(\mathcal{H}) = S \subseteq V(\mathcal{H})$.) Indeed, any instance $(\mathcal{G}, \ell)$ of $LCSP(\mathcal{H})$ can be transformed to an instance $\mathcal{G}'$ of $CSP(\mathcal{H}')$ where $\mathcal{H}'$ is the structure $\mathcal{H}$ augmented with all unary relations $U_S$ (i.e., $\mathcal{H}'$ is conservative), and $\mathcal{G}'$ is constructed from $\mathcal{G}$ by setting $U_S = \{v \in V(\mathcal{G}) \mid \ell(v) = S\}$. By a similar argument, any instance $\mathcal{G}'$ of $CSP(\mathcal{H}')$ for conservative $\mathcal{H}'$ can be transformed to an instance $(G, \ell)$ for $LCSP(\mathcal{H})$ where $\mathcal{H}$ has no (or may have some) unary relations.

It turns out that for $LCSP$ a dichotomy is already known [10]. (We remark this result does not immediately imply a dichotomy for any of $HOM$, $RET$ or $CSP$).

**Theorem 3.5 (Dichotomy for list homomorphism problem (LCSP)).** [10] *Every problem in the class $LCSP$ is either in $P$ or $NP$-complete.*

We should mention that this result was strongly motivated by the results of Feder, Hell and Huang [38, 43, 44] who proved a dichotomy for $LHOM$ in the case of undirected graphs. (See Theorems 3.12, 3.13, and 3.14.)

We close by noting that there also exists a dichotomy for a related problem of counting homomorphisms $\#CSP$ defined formally in [11].

**Theorem 3.6 (Dichotomy for counting homomorphism problem (#CSP)).** [11] *Every problem in the class $\#CSP$ is either in $FP$ (polynomial) or $\#P$-complete.*

### 3.1.2 Polymorphisms

Let $\mathcal{H}$ be a fixed relational structure. It turns out that the complexity of $CSP(\mathcal{H})$ can be determined from particular algebraic properties of $\mathcal{H}$.

We say that a mapping $\varphi : V(H)^k \to V(H)$ where $k \geq 1$ is a *polymorphism* of $H$ if, whenever $u_iv_i \in E(H)$ for all $1 \leq i \leq k$, also $\varphi(u_1, \ldots, u_k)\varphi(v_1, \ldots, v_k) \in E(H)$. Similarly, for a relational structure $\mathcal{H}$ over the vocabulary $\sigma = \{(R_1, k_1), \ldots, (R_t, k_t)\}$, a mapping $\varphi : V(\mathcal{H})^k \to V(\mathcal{H})$ where $k \geq 1$ is a *polymorphism* of $\mathcal{H}$ if, for all $1 \leq j \leq t$, whenever $(u_{1i}, \ldots, u_{k_ji}) \in R_j(\mathcal{H})$ for all $1 \leq i \leq k$, also $(\varphi(u_{11}, \ldots, u_{1k}), \ldots, \varphi(u_{k_j1}, \ldots, u_{k_jk})) \in R_j(H)$. We denote by $\mathcal{P}ol(\mathcal{H})$ the set of all polymorphisms of $\mathcal{H}$.

In [74], the following interesting fact is shown.

**Theorem 3.7.** [74] *Assuming $V(\mathcal{H}) = V(\mathcal{H}')$, if $\mathcal{P}ol(\mathcal{H}') \subseteq \mathcal{P}ol(\mathcal{H})$, then $CSP(\mathcal{H})$ is polynomially reducible to $CSP(\mathcal{H}')$.*

It follows that for structures $\mathcal{H}$ having many polymorphisms, $CSP(\mathcal{H})$ is likely to be polynomial, whereas for structures with few polymorphisms, $CSP(\mathcal{H})$ is expected to be $NP$-complete. In particular, this is true for so-called projective structures. A structure $\mathcal{H}$ is *projective* if the only polymorphisms of $\mathcal{H}$ are $f \circ \pi_{i,k}$ where $f$ is an automorphism of $\mathcal{H}$ and $\pi_{i,k}$ is a projection $(u_1, \ldots, u_k) \mapsto u_i$.

**Theorem 3.8.** [74] *If $\mathcal{H}$ is projective, then $CSP(\mathcal{H})$ is $NP$-complete.*

For example, it turns out that $K_3$ is projective, and so is the structure $N$ over elements $\{0,1\}$ with one ternary relation $E(N) = \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$. Hence, the corresponding $CSP$ problems are $NP$-complete which is no surprise, since $CSP(K_3)$ is precisely 3-colouring and $CSP(N)$ is $NAE$-3-$SAT$ (not all equal 3-$SAT$); both are already known to be $NP$-complete.

On the other hand, having a particular polymorphism in $\mathcal{P}ol(\mathcal{H})$, allows one to solve $CSP(\mathcal{H})$ polynomially. We say that a polymorphism $\varphi$ is *conservative* if $\varphi(v_1,\ldots,v_k) \in \{v_1,\ldots,v_k\}$ for any $v_1,\ldots,v_k \in V(\mathcal{H})$. A *majority* operation is a ternary polymorphism $\varphi$ satisfying $\varphi(u,u,v) = \varphi(u,v,u) = \varphi(v,u,u) = u$ for all $u,v \in V(\mathcal{H})$. A *Mal'tsev* operation is a ternary polymorphism $\varphi$ satisfying $\varphi(u,u,v) = \varphi(v,u,u) = v$ for all $u,v \in V(\mathcal{H})$. A *semilattice* operation is a binary polymorphism $\varphi$ satisfying $\varphi(u,u) = u$, $\varphi(u,v) = \varphi(v,u)$ and $\varphi(u,\varphi(v,w)) = \varphi(\varphi(u,v),w)$ for all $u,v,w \in V(\mathcal{H})$.

**Theorem 3.9.** [12, 49, 64] *If $\mathcal{H}$ admits a majority, or a Mal'tsev, or a semilattice operation, then $CSP(\mathcal{H})$ is polynomial time solvable.*

It should be noted that unlike in the general $CSP$, these three polymorphisms are precisely those that lead to all polynomial cases for the classes $LCSP$ and $\#CSP$ (in both cases a dichotomy is known [10, 11]), namely for $LCSP$ polynomiality is connected with existence of either majority, or Mal'tsev, or semilattice polymorphism, whereas the polynomial cases for $\#CSP$ are roughly those with Mal'tsev operation (which corresponds to solving systems of linear equations over a finite field).

Finally, we should mention that all of the above operations are instances of so-called Taylor operations, which usually give rise to a polynomial time algorithm for $CSP$, and are conjectured to be the only polymorphism that guarantee polynomiality. We say that a $k$-ary polymorphism $\varphi$ is *inclusive in position $i$* if there exists a choice of $u_j, v_j \in \{u,v\}$ for all $1 \leq j \leq k$ where $u_i \neq v_i$ such that $\varphi(u_1,\ldots,u_k) = \varphi(v_1,\ldots,v_k)$ holds for any $u,v \in V(\mathcal{H})$. A $k$-ary polymorphism $\varphi$ is a *Taylor operation* if $\varphi$ is inclusive in each position $1 \leq i \leq k$.

**Conjecture 3.10.** [64] *If $\mathcal{H}$ admits a Taylor operation, then $CSP(\mathcal{H})$ is polynomial time solvable, otherwise $NP$-complete.*

We close by noting that recently other conjectured dichotomy characterisations have appeared in the literature [83, 86, 94]; they all turned out to be equivalent to the one above.

## 3.2 Undirected Graphs

We begin by mentioning a result of Hell and Nešetřil [65] which shows a dichotomy for $HOM$ for the case of undirected graphs.

**Theorem 3.11.** [65] *If $H$ is bipartite or has a loop, then $H$-colouring problem is in $P$, otherwise $H$-colouring problem is $NP$-complete.*

Recently, this result has been reproved by Bulatov [9] using algebraic methods.

### 3.2.1 List Homomorphisms

The complexity characterisation for $LHOM$ problems for undirected graph split into several cases; namely it was first shown for reflexive, then for irreflexive graphs, and finally for graphs allowing both vertices with and without loops.

**Theorem 3.12.** [38] *Let $H$ be a reflexive graph. If $H$ is an interval graph, then $LHOM(H)$ is polynomial time solvable, otherwise $NP$-complete.*

Actually, it turns out that interval graphs $H$ are the only reflexive graphs with so-called *$\underline{X}$-property* (read "$X$-underbar-property"), also called a *min-ordering.* This also coincides with existence of a conservative semilattice polymorphism for corresponding $CSP(\mathcal{H})$ [74]. (The ordering $\prec$ defines a semilattice polymorphism $f(x, y) = x \iff x \prec y$, and $f$ is conservative since $\prec$ is a total ordering; the converse is also true.) On the other hand, one can prove using forbidden induced subgraph characterisation of interval graphs due to Lekkerkerker and Boland [84] (see Theorem 1.17), that for any minimal non-interval graph $H$, the problem $LHOM(H)$ is $NP$-complete, and hence is for all non-interval graphs $H$. (Note that this type of argument generally does not work for $HOM$ properties.)

Similarly, one can show a dichotomy for irreflexive graphs. Note that by Theorem 3.11, for any irreflexive non-bipartite graph $H$, we have that $HOM(H)$ is already $NP$-complete.

**Theorem 3.13.** [43] *Let $H$ be an irreflexive graph. If $H$ is bipartite and the complement of $H$ is a circular-arc graph, then $LHOM(H)$ is polynomial time solvable, otherwise $NP$-complete.*

Finally, in [44], a dichotomy is proved for the general case.

**Theorem 3.14.** [44] *The list homomorphism problem $LHOM$ is polynomial time solvable if $H$ is a bi-arc graph, and $NP$-complete otherwise.*

We conclude by mentioning that a similar dichotomy was proved for a related *minimum cost homomorphism* problem $MinHOM$, where the task is to find a homomorphism of least cost [60]. That is, given an instance $(G, c)$ of $MinHOM(H)$ for a fixed (di)graph $H$ where $G$ is a (di)graph and $c$ a mapping $c : V(G) \times V(H) \to \mathbb{R}$, the cost of a homomorphism $h : G \to H$ with respect to $c$ is defined as $\sum_{v \in V(G)} c(v, h(v))$. Then a solution to $MinHOM(H)$ for $(G, c)$ is a homomorphism from $G$ to $H$ that minimises the cost with respect $c$.

A graph $G$ is a *proper interval graph* if $G$ is the intersection graph of an inclusion-free family intervals on the real line, i.e., no interval of the family is properly contained in another interval of the family. The *intersection bigraph* of two families of sets $\mathcal{I} = \{I_1, \ldots, I_k\}$ and $\mathcal{J} = \{J_1, \ldots, J_\ell\}$ is the graph with vertices $\{v_1, \ldots, v_k, u_1, \ldots, u_\ell\}$ and edges $v_i u_j \iff I_i \cap J_j \neq \emptyset$. A graph $G$ is a *proper interval bigraph* if $G$ is the intersection graph of two inclusion-free families of intervals on the real line.

**Theorem 3.15.** [60] *Let $H$ be a connected graph. If $H$ is a proper interval graph or a proper interval bigraph, then $MinHOM(H)$ is polynomial time solvable, otherwise $NP$-complete.*

## 3.3 Matrix Partitions

Let $M$ be a $k \times k$ (symmetric) matrix with entries from $\{0, 1, *\}$. An *$M$-matrix partition* of a (di)graph $G$, or just *$M$-partition* of $G$, is a partition of the vertices $V(G)$ into sets $V_1, \ldots, V_k$ such that for any $1 \leq i, j \leq k$ and vertices $x \in V_i$ and $y \in V_j$, we have $xy \in E(G)$ if $M_{i,j} = 1$ and $x \neq y$, and $xy \notin E(G)$ if $M_{i,j} = 0$. For a fixed $k \times k$ matrix $M$, the *$M$-partition problem* is a problem of deciding, given a graph $G$, whether $G$ admits an $M$-partition. Similarly, the *list $M$-partition problem* is a problem of deciding, given a graph $G$ with lists $\ell(v) \subseteq \{1 \ldots k\}$ for all $v \in V(G)$, whether $G$ admits an $M$-partition $V_1, \ldots, V_k$ such that for all $v \in V(G)$, we have $v \in V_i$ if and only if $i \in \ell(v)$.

It is easy to see that (list) matrix partitions generalise (list) homomorphisms, namely (list) $H$-colouring is precisely (list) $M$-partition where $M$ is the adjacency matrix of $H$ with entries 1 replaced with $*$. Also, (list) matrix partitions generalise (list) full homomorphisms; here the matrix $M$ is exactly the adjacency matrix $H$. (A *full homomorphism* is a mapping $h : V(G) \to V(H)$ such that $xy \in E(G)$ if and only if $h(x)h(y) \in E(H)$ for all $x, y \in V(G)$.)

On the other hand, there are multiple examples of interesting problems [46] that are expressible as an $M$-partition, but are not expressible as a (full) homomorphism, e.g., a clique cutset, a homogeneous set, and a skew cutset problems. It should be noted that any $M$-matrix partition problem can be cast as an instance of a special $CSP$ called *trigraph homomorphism* which is a homomorphism between structures with two binary relations $E$ and $N$. An instance $G$ of the $M$-partition problem where $M$ is a $k \times k$ matrix corresponds precisely to the instance $\mathcal{G}$ of $CSP(\mathcal{H})$ where $E(\mathcal{G}) = E(G)$, $N(\mathcal{G}) = V(G) \times V(G) \setminus E(G)$, $V(\mathcal{H}) = \{1, \ldots, k\}$, $E(\mathcal{H}) = \{ij \mid M_{i,j} = 1 \text{ or } *\}$, and $N(\mathcal{H}) = \{ij \mid M_{i,j} = 0 \text{ or } *\}$.

Now we investigate the complexity of the (list) $M$-partition problems for fixed symmetric matrix $M$. First we should mention that it follows from Theorems 3.12, 3.13, 3.14 that for matrices with no zeroes or no ones, one can obtain a dichotomy for the list $M$-partition problem for symmetric matrices $M$. (Similarly a dichotomy for the $M$-partition problem for such matrices follows from Theorem 3.11.)

**Theorem 3.16.** [46] *If a symmetric matrix $M$ contains no 0 or no 1 entry, then the (list) $M$-partition problem is $NP$-complete or polynomial time solvable.*

Apart from trying to prove a dichotomy, the following weaker *quasi-dichotomy* for list $M$-partition problem was conjectured in [46] and proved in [40]. (The problem is *quasipolynomial* is it has a solution with running time $O(n^{c \cdot \log^t n})$ for some positive constants $c, t$.)

**Theorem 3.17.** [40] *All list $M$-partition problems are quasipolynomial, or $NP$-complete.*

In fact, in [40] a more general result about so-called three-inclusive $W$-full $CSP$ is given. (See also [64].) Note that while, of course, a dichotomy would be preferred, the existence of a quasipolynomial algorithm for a problem suggests that the problem is not likely $NP$-complete, since no $NP$-complete problem is known to have a quasipolynomial time solution. Unfortunately, a dichotomy in this case is not expected strongly, since there exists a certain 4x4 "stubborn" matrix [13] (see Figure 3.1) for which the problem is quasipolynomial but no polynomial solution is known as of yet, and it is believed that this matrix is not just a single exception but many other such matrices may exist.

$$\begin{pmatrix} 0 & * & 0 & * \\ * & 0 & * & * \\ 0 & * & * & * \\ * & * & * & 1 \end{pmatrix}$$

**Figure 3.1**: The “stubborn” matrix.

### 3.3.1 Sparse-Dense partitions

One of the tools used successfully in solving matrix partition problems is the *sparse-dense partition* algorithm [46]. Let $\mathcal{S}$ and $\mathcal{D}$ be two induced-hereditary classes (or properties) of graphs for which there exists a constant $c$ such that any graph in $\mathcal{S} \cap \mathcal{D}$ has at most $c$ vertices. ($\mathcal{S}$ can be thought of as a sparse class and $\mathcal{D}$ as a dense class.) A *sparse-dense* partition of a graph $G$ is a partition of $V(G)$ into two sets $V_{\mathcal{S}}, V_{\mathcal{D}}$ where $G[V_{\mathcal{S}}] \in \mathcal{S}$ and $G[V_{\mathcal{D}}] \in \mathcal{D}$.

**Theorem 3.18.** [46] *Any graph $G$ on $n$ vertices has at most $n^{2c}$ sparse-dense partitions. Furthermore, all sparse-dense partitions of $G$ can be enumerated in time $O(n^{2c+2}T(n))$ where $T(n)$ is the time needed to recognise an $n$-vertex graph in $\mathcal{S}$ or an $n$-vertex graph in $\mathcal{D}$.*

For example, if $\mathcal{S} = \mathcal{O}$ and $\mathcal{D} = \mathcal{K}$, i.e., sparse graphs are independent sets and dense graphs are cliques, then we have $c = 1$ implying a polynomial time algorithm for recognising split graphs. Similarly, for $\mathcal{S} = \{\text{planar graphs}\}$ and $\mathcal{D} = \mathcal{K}$, we have $c = 4$ since $K_5$ is not planar, and again $\mathcal{S} \circ \mathcal{D}$ has a polynomial time solution.

Applying this in the context of matrix partitions, one can show that the list $M$-partition problem is polynomial time solvable for matrices with no $*$ entries. Note that this result complements Theorem 3.16.

**Theorem 3.19.** [46] *If $M$ contains no $*$ entry, then the (list) $M$-partition problem is polynomial time solvable.*

### 3.3.2 Small cases

For symmetric matrices of small sizes, a complete characterisation of complexity is known.

**Theorem 3.20.** [46] *Suppose that the size of $M$ is $k = 3$. Then the list $M$-partition problem is $NP$-complete if $M$ or its complement is the matrix of 3-colouring or the stable cutset problem, and polynomial time solvable otherwise.*

**Theorem 3.21.** [13, 46] *Suppose that the size of $M$ is $k = 4$, and $M$ is not the "stubborn" matrix or its complement. Then the list $M$-partition problem is $NP$-complete if $M$ contains a submatrix corresponding to the problem of 3-colouring or its complement, and polynomial time solvable otherwise.*

Note that if $M$ contains a $*$ on the main diagonal, the $M$-partition problem (without lists) is trivial, hence since the "stubborn" matrix contains a star on the main diagonal, it follows that Theorem 3.21 provides a dichotomy for the $M$-partition problem.

Note that by simultaneously permuting rows and columns, we can assume that a symmetric square matrix $M$ of size $k+l$ with no $*$'s on the main diagonal is arranged as follows: $M = \left(\begin{array}{c|c} A & C^T \\ \hline C & B \end{array}\right)$ where $A$ is a symmetric $k \times k$ matrix with all zeroes on the main diagonal, and $B$ is symmetric a $l \times l$ matrix with all ones on the diagonal. We write that $M$ is an $(A, B, C)$-block matrix.

## 3.3.3 Partitioning chordal graphs

In this section, we mention results about the complexity of $M$-partition problems on chordal graphs. As one would expect, the situation is much more pleasing than in the general case. In particular, the following claims are shown in [47]. A matrix is said to be *crossed* if each non-$*$ entry belongs to a row or a column of non-$*$ entries.

**Theorem 3.22.** [47] *If all diagonal entries of $M$ are zero or all diagonal entries of $M$ are one, then the chordal list $M$-partition problem can be solved in polynomial time.*

**Theorem 3.23.** [47] *Suppose that $M$ is an $(A, B, C)$-block matrix. If $C$ is crossed, then the chordal list $M$-partition can be solved in polynomial time.*

Note that by Theorem 3.22, all $HOM$ properties and all $LHOM(H)$ properties for irreflexive $H$ in chordal graphs are polynomial time solvable. On the other hand, there are instances of matrices $M$ for which the list $M$-partition problem is $NP$-complete already on split graphs, and matrices $M$ for which the chordal $M$-partition problem (without lists) is $NP$-complete [47].

Finally, a special case of chordal $M$-partition problem, namely the property $\mathcal{O}^k \circ \mathcal{K}^l$, i.e., $M = (A, B, C)$ where all entries in $C$ and all off-diagonal entries in $A$ and $B$ are $*$, has been shown to admit a simple linear time algorithm [68] and it is characterised by a single minimal forbidden induced subgraph, namely $(l+1)K_{k+1}$ [67].

We conclude by mentioning that for cographs, all $M$-partition problems have been shown to admit a polynomial time solution [42].

### 3.3.4 Bounded size obstructions

Let $M$ be a fixed symmetric matrix. In this section, we focus on *minimal obstructions* (also called *forbidden induced subgraphs*) for the $M$-partition problem, that is, graphs that do not admit an $M$-partition, but whose all proper induced subgraph do. Clearly, any graph containing a minimal obstruction for the $M$-partition problem as an induced subgraph cannot admit an $M$-partition; this follows from the fact that the class of graphs that admit an $M$-partition is induced hereditary (closed under taking induced subgraphs). Actually, the previous argument is true for any induced hereditary property. It follows that if for the matrix $M$, the $M$-partition problem (or any induced hereditary property, for that matter) has only a finite number of minimal obstructions, it can be solved in polynomial time; one only needs to check whether the input graph contains one of the obstructions. It turns out that in number of polynomially solvable problems, some of which we already described, this is precisely the case; for instance, for any matrix $M$ which has all off-diagonal entries $*$, we have a single minimal obstruction for the $M$-partition problem in chordal graphs [67].

Unfortunately, even though we might know that for a particular matrix $M$ the number of obstructions is finite, the actual number of obstructions can be very large. Hence, instead, we shall settle for showing an upper bound on the size of any minimal obstruction; clearly, if this bound is a constant for $M$ (i.e., depending only on $M$), the number of obstructions will be finite.

First we mention the following result from [48]. A matrix $M = (A, B, C)$ is *friendly* if $A$ and $B$ contain no $*$ entries.

**Theorem 3.24.** [48] *If $M$ is not a friendly matrix, then there are infinitely many minimal obstructions for the $M$-partition problem.*

Note that the converse of this theorem is not true [48], that is, there are friendly matrices with infinitely many obstructions for the $M$-partition problem. However, in a special case when $M$ has no $*$ entries, it turns out to be true [41]. We again assume that $M = (A, B, C)$ where $A$ is a $k \times k$ matrix and $B$ is a $l \times l$ matrix.

**Theorem 3.25.** [41] *If $M$ has no $*$ entries then a minimal obstruction for the $M$-partition problem has at most $(k+1)(l+1)$ vertices, and moreover, there are at most two minimal obstructions with precisely $(k+1)(l+1)$ vertices.*

Note that these results concern all minimal obstructions for the $M$-partition problem, whereas if we consider only those minimal obstructions that are for instance perfect, or chordal, the number of them can be finite, even though the total number of minimal obstructions is infinite. Clearly, for the $M$-partition problem in perfect, respectively chordal graphs, one only needs to consider minimal obstructions that are perfect, respectively chordal. For instance, the two by two matrix $M$ with zeroes on the main diagonal and $*$ off the diagonal has (by Theorem 3.24) infinitely many obstructions, namely all odd cycles, whereas it has only one obstruction $K_3$ in perfect and in chordal graphs.

Now we shall restrict our attention to matrices $M = (A, B, C)$ where all off-diagonal entries in $A$ are in $a$, all off-diagonal entries in $B$ are in $b$, and all entries in $C$ are in $c$ where $a, b, c \subseteq \{0, 1, *\}$. The following results describe different cases for $a, b, c$ in which the number perfect graphs that are minimal obstructions to the $M$-partition problem is finite.

**Theorem 3.26.** [39] *Any perfect graph which is a minimal obstruction to the $M$-partition problem has a bounded number of vertices for the following cases of $a, b, c$.*

*i)* $|a| = |b| = |c| = 1$ *and* $c \neq \{*\}$,

*ii)* $|c| = 1$, $a = \{0, 1\}$ *or* $a = \{*\}$, *and* $b = \{0, 1\}$ *or* $b = \{*\}$.

Note that in the exact bounds on the size of obstructions can be exponential in $k$ and $l$, in particular they are exponential in some cases of *ii*). Finally, we remark that it is possible to improve these bound when considering only chordal graphs instead of perfect graphs [46].

Chapter 4

# Decompositions

In this chapter, we investigate particular techniques for decomposing graphs into atomic subgraphs and discuss related complexity issues. Namely we study decomposition by clique separators, modular decomposition, tree decomposition, and graph grammars with relation to logic of graphs.

## 4.1 Clique separators

By a result of Whitesides [116], for any graph $G$, it is possible to efficiently identify a *clique separator* of $G$ (i.e., a clique whose removal disconnects $G$) if one exists.

**Theorem 4.1.** [116] *One can find in time $O(nm)$ a separating clique in a graph if one exists.*

In many combinatorial (optimization) problems, the notion of a separating clique plays an important role. For instance, if a graph $G$ contains a clique $C$ that separates $G$ to $G[A]$ and $G[B]$, and if one can colour $G[A \cup C]$ and $G[B \cup C]$, then a colouring of $G$ is obtained easily by permuting the colours in the colourings of $G[A \cup C]$ and $G[B \cup C]$ so that they match on $C$. This, in particular, makes the graph colouring problem on chordal graphs easy, since by Theorem 1.7 any chordal graph is separable by a clique (unless the graph itself is a clique).

Clearly, as we just described, if a graph contains a separating clique $C$, one can decompose it into subgraphs $G[A \cup C]$ and $G[B \cup C]$ and apply this rule to $G[A \cup C]$ and $G[B \cup C]$ independently. This way one can obtain a special tree decomposition $(T, X)$ of $G$ (defined later in the chapter) in which for any edge $uv \in E(T)$, the set $X(u) \cap X(v)$ induces a clique in $G$, and for all $u \in V(T)$ the graph $G[X(u)]$ does no contain any clique separators. (Such a decomposition is called a *clique decomposition* of $G$.) This was already pointed out by Tarjan in [109] where he describes an efficient algorithm to do so. (Note that the complexity of his algorithm is better than the naïve application of Theorem 4.1 alone.)

**Theorem 4.2.** [109] *One can in time $O(nm)$ compute a clique decomposition of a graph.*

## 4.2 Modular decomposition

A *module* $M$ (also called a *homogeneous set*) of a graph $G$ is a set of vertices $M \subseteq V(G)$ such that each vertex of $G - M$ is either adjacent to every vertex in $M$ or non-adjacent to every vertex in $M$. (That is, no two vertices in $M$ are distinguishable by their neighbourhoods in $G - M$.) A module $M$ is called *trivial* if $M = V(G)$ or $M$ contains a single vertex. Note that, if two modules $N$ and $M$ of $G$ intersect, then both $N \cup M$ and $N \cap M$ are also modules of $G$. A module $M$ is *strong* if for any module $N$ with $N \cap M \neq \emptyset$, we have $N \subseteq M$ or $M \subseteq N$. A strong module $M \subseteq N$ is a *maximal submodule* of a module $N$, if there is no strong module $M'$ with $M \subsetneqq M' \subsetneqq N$. The following fact was first shown in [101].

**Proposition 4.3.** [101] *Every vertex of a non-trivial module $M$ belongs to a unique maximal submodule of $M$.*

Hence, applying this to $M = V(G)$ we have that one can uniquely decompose any graph $G$ to a collection of strong modules of $G$. This decomposition is called the *modular decomposition* of $G$; it is also known as the *substitution decomposition*. The decomposition of $G$ into its strong modules can be captured by the following *modular decomposition tree* $T_G$. The nodes of $T_G$ correspond to the strong modules of $G$, the root node is $V(G)$, the leaves are the vertices of $G$, and the children of every internal node $M$ are the maximal submodules of $M$. Note that the tree $T_G$ is unique.

The complexity of computing modular decomposition was first shown to be $O(n^2)$ [93], then later improved on chordal graphs to $O(n + m)$ [72], and finally shown to be $O(n + m)$ [87, 23] in general graphs. All these results concern undirected graphs. For directed graphs, also an $O(n + m)$ time algorithm was recently discovered [88].

Modular decomposition has been discovered independently by researchers in graph theory, network theory, game theory, and other areas [54, 93]. In particular it was used in efficient recognition of comparability graphs, efficient solutions to chromatic number and minimum clique problems for comparability graphs [87, 102], and efficient recognition of chordal comparability and interval graphs [72].

## 4.3 Graph Grammars

Graph grammars originated as a natural generalisation of formal language theory to graphs and since found their applications in numerous areas of computer science such as VLSI layout schemes, database design, modeling of concurrent systems, pattern recognition, compiler construction and others. Several different flavours of grammars for graphs have been studied; some are based on replacing vertices, and some based on replacing (hyper)-edges, while others replace whole subgraphs; here, additionally, mechanisms for "gluing" graphs can vary. In all of these models, nodes or edges of graphs are usually labeled, and it is the labels that control the way graphs are transformed by the grammar.

Here, we focus only on one particular model of graph grammars, hyperedge replacement grammars (HRG). These grammars are particularly interesting since they are a direct counterpart to the context-free grammars (CFG) on strings. (As we shall see, many algorithms for CFG carry over to HRG.)

For a set $S$, we denote by $S^*$ the set of all finite sequence of elements of $S$. Let $\mathcal{C}$ be an arbitrary (fixed) set of *labels* and let $\mathit{type} : \mathcal{C} \to \mathbb{N}$ be a *typing* function. A *hypergraph* $H$ over $\mathcal{C}$ is a tuple $(V_H, E_H, \mathit{att}_H, \mathit{lab}_H, \mathit{ext}_H)$ where $V_H$ is a finite set of nodes, $E_H$ is a finite set of hyperedges, $\mathit{att} : E_H \to V_H^*$ is a mapping assigning a sequence of pairwise distinct *attachment nodes* $\mathit{att}_H(e)$ to each $e \in E_H$, $\mathit{lab}_H : E_H \to \mathcal{C}$ is a mapping that *labels* each hyperedges such that $\mathit{type}(\mathit{lab}_H(e)) = |\mathit{att}_H(e)|$, and $\mathit{ext}_H \in V^*$ is a sequence of pairwise distinct external nodes. Note that in this model hyperedges are ordered subsets of vertices and are labeled according to their cardinality. External nodes $\mathit{ext}_H$, in general, do not have to be specified, they are only used in grammar specification to describe how hyperedges are replaced. We denote by $\mathcal{H}_\mathcal{C}$ the set of hypergraphs over $\mathcal{C}$.

Now we define *hyperedge replacement* mechanism. Let $H \in \mathcal{H}_\mathcal{C}$ and let $e_1, \ldots, e_k \subseteq E(H)$ be a set of hyperedges to be replaced by hypergraphs $H_1, \ldots, H_k \in \mathcal{H}_\mathcal{C}$ where $\mathit{type}(e_i) = |\mathit{ext}_{H_i}|$ for $1 \leq i \leq k$. The graph $H[e_1/H_1, \ldots, e_k/H_k]$ is constructed from the disjoint union of $H$ and the graphs $H_1, \ldots, H_k$ by identifying the vertices of $\mathit{att}_H(e_i)$ with the vertices of $\mathit{ext}_{H_i}$ in their respective orders, for each $1 \leq i \leq k$, and then removing hyperedges $e_1, \ldots, e_k$. It is not difficult to observe that this mechanism is "context-free", that is, we have $H[e_1/H_1, \ldots, e_k/H_k] = H[e_1/H_1] \ldots [e_k/H_k]$ and $H[e_1/H_1][e_2/H_2] = H[e_2/H_2][e_1/H_1]$, and $H[e_1/H_1][e_2/H_2] = H[e_1/H_1[e_2/H_2]]$ for edges $e_i$ in appropriate hypergraphs.

A *hyperedge replacement grammar* $HRG$ is a tuple $(N, T, P, S)$ where $N \subseteq \mathcal{C}$ is a set of *nonterminals*, $T \subseteq \mathcal{C}$ with $T \cap N = \emptyset$ is a set of *terminals*, $P \subseteq N \times \mathcal{H}_{\mathcal{C}}$ is a finite set of *productions* with $type(A) = |ext_R|$ whenever $(A, R) \in P$, and $S \in N$ is the *start symbol.*

A *derivation step* in $HRG$ is the binary relation $\underset{P}{\Longrightarrow}$ on $\mathcal{H}_{\mathcal{C}}$ with $H \underset{P}{\Longrightarrow} H'$ whenever $H' = H[e/R]$ where $lab_H(e) = A$ and $(A, R) \in P$. We denote by $\underset{P}{\Longrightarrow}^*$ the transitive closure of $\underset{P}{\Longrightarrow}$. For $A \in \mathcal{C}$, let $A^\bullet$ denote the hypergraph with a single hyperedge $A$ attached to $type(A)$ vertices. ($A^\bullet$ is usually called a *handle.*) The *hypergraph language* $L(HRG)$ *generated* by the grammar $HRG$ is $L_S(HRG)$, where for $A \in N$, the set $L_A(HRG)$ consists of all hypergraphs in $\mathcal{H}_T$ derivable from $A^\bullet$ by applying productions of $P$. That is,

$$L_A(HRG) = \{H \in \mathcal{H}_T \mid A^\bullet \underset{P}{\Longrightarrow}^* H\}.$$

The context-free nature of hyperedge replacement allows one to define derivation trees for hyperedge replacement grammars similar to the ones defined for context-free languages. (For simplicity we deviate from the standard definition given in [99].)

A *derivation tree* $\mathcal{T}$ in $HRG = (N, T, P, S)$ is a tree in which each node is labeled either by a terminal symbol $A \in T$, a nonterminal symbol $A \in N$, or by a production rule $(A, R) \in P$ such that $(i)$ a vertex $v$ labeled by $A \in T$ has no children and we define $res(v) = A^\bullet$, $(ii)$ a vertex $v$ labeled by $A \in N$ either has no children and $res(v) = A^\bullet$ or has exactly one child $w$ labeled by a production rule $(A, R) \in P$ and $res(v) = res(w)$, and $(iii)$ a vertex $v$ labeled by a production rule $(A, R) \in P$ has a child $v_i$ labeled by $lab_R(e_i)$ for each hyperedge $e_i \in E(R)$ where $1 \leq i \leq k = |E(R)|$ and we define $res(v) = R[e_1/H_1, \ldots, e_k/H_k]$ where $H_i = res(v_i)$. The graph defined by $\mathcal{T}$ is $res(r)$ where $r$ is the root of $\mathcal{T}$. Now it is easy to see that for any hypergraph $H \in L_A(HRG)$ one can construct a derivation tree $\mathcal{T}$ whose root $r$ is labeled by $A$ such that $res(r) = H$.

A hyperedge replacement grammar $HRG = (N, T, P, S)$ is said to be of *order* $k$ if $type(A) \leq k$ for all $A \in N$. An example of a grammar of order 3 generating all partial 3-trees (cf. Section 4.4) can be seen in Figure 4.1.

We now briefly mention some properties of hyperedge replacement grammars. In terms of their generative power, they can generate all context-free string languages as well as some context-sensitive languages (for appropriately defined string graphs). On the other hand, similarly to context-free grammars, analogues of pumping lemma and Parikh's theorem hold for HRG's. Furthermore, as a consequence of the hyperedge replacement rule, graphs generated by a hyperedge replacement grammar of order $k$ are always at most $k$-connected. Then, since the description of HRG is finite, no HRG can generate graphs of arbitrary large connectivity. Also, for any $k$, there exists a HRG of order $k$ generating all partial $k$-trees (see Figure 4.1 for an example), whereas no HRG of order $k-1$ can generate them. It follows that graphs generated by HRG's of different orders form a proper infinite hierarchy. Finally, there is a characterisation of string graphs generated by HRG's in terms of so-called deterministic tree walking transducers [99].

Now we discuss complexity of the membership problem for HRG. It is not difficult to see that the problem is in $NP$. One has to guess the derivation and test whether it generates the input graph; the derivation can be described by a polynomial number of bits since it is not possible to erase vertices. (Actually [99], for every $HRG$, there exists a linear function $f$ such that, if an $n$-vertex hypergraph $H \in L(HRG)$, then $H$ has a derivation of length at most $f(n)$.) In general the membership problem turns out to be $NP$-hard, and in particular the following is true. (Here, 'linear' and 'edge replacement' means that the rules contain only one nonterminal on the right hand side, and that the grammar is of order two.)

**Theorem 4.4.** [81] *There exists a linear edge replacement grammar that generates an $NP$-complete graph language of maximum degree 2. There exists an edge replacement grammar that generates an $NP$-complete graph language of connected graphs.*

Note that in this theorem the first grammar generates disconnected graphs with unbounded number of connected components, whereas the second grammar generates graphs of unbounded degree. This is in particular important since, if the grammar only generates hypergraphs of bounded degree and of bounded number of connected components (or more precisely, hypergraphs whose $k$-separability is $O(\log n)$), the membership problem is polynomial time solvable. (By $k$-separability of a hypergraph $H$ we mean the maximum number of connected components of $H - X$ where $X \subseteq V_H$ and $|X| = k$.) It should be noted that the algorithm is a variant of the famous Cocke-Younger-Kasami algorithm for recognising context-free languages.

**Theorem 4.5.** [99, 82] *Let $HRG$ be a hyperedge replacement grammar of order $k$. If for each $n$-vertex hypergraph $H \in L(HRG)$, $k$-separability of $H$ is $O(\log n)$, then the membership problem for $HRG$ is polynomial time solvable. In case $k$-separability is $O(1)$ for all $H \in L(HRG)$, the membership problem is in $LOGCFL$.*

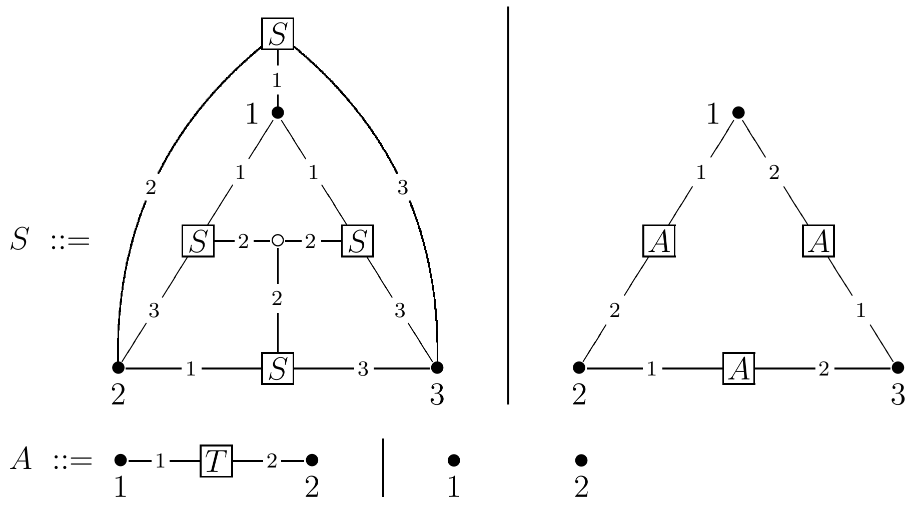

**Figure 4.1**: An example of a hyperedge replacement grammar generating all partial 3-trees. The grammar uses two nonterminal labels $A$ and $S$, where $S$ is the starting nonterminal, and one terminal label $T$. The vertices marked as $\circ$ and $\bullet$ represent nodes where the latter are the external nodes $ext_H$ for particular hypergraph $H$ (the labels indicate their ordering in $ext_H$). The vertices in boxes represent hyperedges where the edges in the diagram connect each hyperedge $e$ with its attachment nodes $att_H(e)$ (the labels on edges indicated their ordering in $att_H(e)$). The diagram is a modification of a diagram from [99].

We conclude by mentioning the following results. A $k$-*uniform* hypergraph or simply a $k$-*hypergraph* is a hypergraph whose all hyperedges are of cardinality $k$. In [26], it was shown that if an $HRG$ of order $k$ generates only $k$-hypergraphs, then there exists a cubic time algorithm (for any $k$) recognising graphs in $L(HRG)$, whereas [27] if the grammar of order $k$ is allowed to use edges of all cardinalities, it can generate an $NP$-complete graph language. These result improve Theorems 4.5 and 4.4 respectively.

**Theorem 4.6.** [26] *Let $HRG$ be a hyperedge replacement grammar of order $k$. Then there exists a cubic time algorithm to decide, given a $k$-hypergraph $H$, whether $H \in L(HRG)$.*

**Theorem 4.7.** [27] *For $k \geq 3$, there exists a hyperedge replacement grammar of order $k$ that generates an $NP$-complete set of $k$-connected hypergraphs.*

## 4.4 Treewidth and Monadic Second Order Logic

A *tree decomposition* $(T, X)$ of a connected graph $G$ is a pair $(T, X)$ where $T$ is a tree and $X$ is a mapping from $V(T)$ to the subsets of $V(G)$, such that *(i)* for any edge $ab \in E(G)$, there exists $u \in V(T)$ with $a, b \in X(u)$, and *(ii)* for any vertex $a \in V(G)$, the vertices $u \in V(T)$ with $a \in X(u)$ induce a connected subgraph in $T$. For a tree decomposition $(T, X)$, the sets $X(u)$ are called *bags*. Note that a clique-tree of a chordal graph $G$ that we defined earlier is precisely a tree decomposition $(T, X)$ of $G$ where $\{X(u) \mid u \in V(T)\}$ is the set of all maximal cliques of $G$. The *width* of a tree decomposition $(T, X)$ is defined as one less than the size of a largest bag $X(u)$ among $u \in V(T)$. The *treewidth* of $G$, denoted by $tw(G)$, is the smallest integer $k$ such that $G$ admits a tree decomposition $(T, X)$ of width $k$.

A *k-tree* is any graph defined recursively as follows: $(i)$ the complete graph on $k + 1$ vertices is a $k$-tree, and $(ii)$ a graph obtained from a $k$-tree $G$ by the addition of a new vertex that is adjacent to a $k$-clique of $G$ is a $k$-tree. A *partial k-tree* is any (not necessarily induced) subgraph of a $k$-tree. It can be seen that the treewidth of any $k$-tree is exactly $k$. Moreover, the partial $k$-trees are precisely the graphs of treewidth at most $k$. In particular, 1-trees are precisely trees, and partial 1-trees correspond to forests.

Now we introduce different types of logics we shall discuss here briefly in connection with bounded treewidth graphs. Let $\mathcal{V}$ be a countable alphabet of *variables* (denoted by $x$,$y$,$z\ldots$). Let $\mathcal{R}$ be a (finite) vocabulary of relational symbols $R$ and their arities $\rho(R)$. A *first order formula* (FO) is any (finite) formula that can be constructed from atomic formulas using binary operations $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$ and quantification symbols $\forall x, \exists x$ where atomic formulas are $x = y$ and $R(x_1, \ldots, x_k)$ for $x, y, x_1, \ldots, x_k \in \mathcal{V}$ and $R \in \mathcal{R}$ with $\rho(R) = k$.

Let $\mathcal{X}$ be a countable alphabet of *relational variables* (denoted by $X$,$Y$,$Z\ldots$) and their arities $\rho(X)$. A *second order formula* (SO) is a (finite) formula that can be constructed from atomic formulas using binary operations $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$ and quantification symbols $\forall x, \exists x, \forall X, \exists X$ where atomic formulas are $x = y$, $R(x_1, \ldots, x_k)$ and $X(x_1, \ldots, x_k)$ for $x, y, x_1, \ldots, x_k \in \mathcal{V}$, $R \in \mathcal{R}$ and $X \in \mathcal{X}$ with $\rho(R) = \rho(X) = k$.

An SO formula $\varphi$ is called a *monadic second order formula* (MS) if $\varphi$ contains only relational variables $X \in \mathcal{X}$ of arity one (so-called *set variables*); the arities of relational symbols $R \in \mathcal{R}$ in $\varphi$ are unrestricted.

The semantics of FO, SO and MS is defined as usual (we omit the formal details). Now we turn our attention to graphs. A (di)graph $H$ with vertex set $V(H)$ and edge set $E(H)$ defines the structure $|H|_1 = (V(H); adj)$ with one binary relational symbol *adj* representing pairs of adjacent vertices. Similarly $H$ defines the structure $|H|_2 = (V(H) \cup E(H); inc)$ with one ternary relational symbol *inc* representing triples, each consisting of an edge and vertices incident to it. In addition, one can introduce into $|H|_2$ a (finite) collection of relational symbols $lab_a$ of arity one which describe labels of vertices and edges of $H$. Also, the structure $|H|_2$ can be defined for $k$-hypergraphs $H$ (fixed $k$) using $(k+1)$-ary relational symbol *inc* (similarly for $|H|_1$).

We shall refer to $MS$ logic of hypergraphs as $MS_1$, respectively $MS_2$ relative to the representation of hypergraphs $H$ by structures $|H|_1$, respectively $|H|_2$.

In general, all $MS_1$-expressible properties are $MS_2$-expressible, whereas not every $MS_2$-expressible property is $MS_1$-expressible (a notable example is Hamiltonicity [99]). However, when restricted to $k$-trees, this is no longer the case and the converse is also true.

**Theorem 4.8.** [19] *The same properties of finite simple hypergraphs over (a finite alphabet) $A$ of treewidth at most $k$ (fixed $k$) are expressible in $MS_1$ and in $MS_2$.*

Now finally, we are ready to describe the main theorem of this section. The theorem deals with hypergraph replacement grammars and their relation to properties of hypergraphs definable by MS. (We refer the reader to Section 4.3 for definitions related to graph grammars.)

**Theorem 4.9.** [16, 17, 18] *Let $L$ be a $MS_2$-definable set of (labeled) hypergraphs (i.e., the set of (labeled) hypergraphs satisfying some $MS_2$ formula $\varphi$). Let $HRG$ be any hyperedge replacement grammar.*

*i) One can construct a hypergraph replacement grammar $HRG'$ generating $L \cap L(HRG)$.*

*ii) For every derivation tree $\mathcal{T}$ of $HRG$, one can decide in time $O(size(\mathcal{T}))$ whether the graph $G$ in $L(HRG)$ defined by $\mathcal{T}$ is in $L$.*

*iii) One can construct an algorithm, that, for any given oriented graph $G$, gives in time $O(size(G)^2)$ the following possible answers:*

*a) $G \notin L(HRG)$,*

*b) $G \in L$ without saying whether $G \in L(HRG)$,* *c) $G \notin L$ without saying whether $G \in L(HRG)$.*

Now we remark that the set of partial $k$-trees is definable by a hypergraph replacement grammar $HRG_k$. (See Figure 4.1 for a concrete example for $k = 3$; for other $k$ the grammar $HRG_k$ is defined similarly.) Furthermore, any derivation tree $\mathcal{T}$ of a graph $H$ in $HRG_k$ has size $O(|V(H)|)$ which can be shown as follows. First starting from $K_k$ we derive in $HRG_k$ a suitable $k$-tree $G \supseteq H$ by adding at most $|V(H)|$ vertices. Then we remove edges of $G$ that are not in $H$ one by one. Each time we either add a vertex or remove an edge; hence at most $|V(H)| + |V(H)|k + k^2$ steps, since in each step in which we add a vertex, we add exactly $k$ edges adjacent to this vertex and $K_k$ we start with has $\leq k^2$ edges. Finally, note that each derivation step produces at most $k+1$ new nodes in the derivation tree $\mathcal{T}$, and the description of each node in $\mathcal{T}$ is of constant size. This gives us the following fundamental corollary.

**Theorem 4.10.** [16, 17, 18] *Any property of graphs of treewidth at most k that is expressible in Monadic Second Order Logic (in either $MS_1$ or in $MS_2$) is decidable in linear time.*

We close by remarking that using only Theorem 4.9 *i*) and Theorems 4.5, and 4.6 does not give us a comparable result to the above (that is, a polynomial time algorithm for the membership problem for $HRG'$ from Theorem 4.9), since the structure of $HRG'$ is not guaranteed to satisfy the conditions of either of Theorems 4.5, 4.6.

Chapter 5

# Problems and Initial solutions

In this chapter, we briefly state the results we obtained so far, and discuss possible extensions.

An *$H$-transversal* of a graph $G$ is a set of vertices $S$ of $G$ that meets all induced copies of $H$ in $G$. An *$F$-free $H$-transversal* of $G$ is an $H$-transversal $S$ which induces an $F$-free graph in $G$. Any $F$-free $H$-transversal $S$ of $G$ can be simply viewed as a colouring of $G$ with two colours (one for $S$ and one for $G - S$) such that one colour induces an $F$-free graph and the other colour an $H$-free graph. Using the terminology of Chapter 2, a graph $G$ having an $F$-free $H$-transversal is a graph with the property $\text{Forb}_{\leq}(F) \circ \text{Forb}_{\leq}(H)$. For instance, if $F = K_2$ and $H = K_2$ then this is precisely the (proper) two colouring. We remark that most of the problems mentioned in this chapter are of this type.

## 5.1 $P_4$-transversals of Chordal Graphs

Recall that a graph is *perfect* if for every induced subgraph $H$ of $G$, the chromatic number $\chi(H)$ is equal to the clique number $\omega(H)$. The Strong Perfect Graph Theorem states that a graph is perfect if and only if it has no induced odd cycle or its complement [15]. This result had been conjectured by Berge [6]. In the long history of this conjecture, the study of the structure of $P_4$'s in a graph has been found to play an important role. In [70], the authors define the notion of a $P_4$-transversal. They show that if a graph has a $P_4$-transversal with certain properties, it is guaranteed to be perfect. They also investigate the complexity of finding a $P_4$-transversal with various properties. In particular they investigate stable $P_4$-transversals, i.e., $P_4$-transversals which form an independent set. They show that for comparability graphs (and therefore also for perfect graphs) it is $NP$-complete to decide whether a graph has a stable $P_4$-transversal. In [71], the authors consider a related problem of $P_4$-colourings (corresponding in our terminology to $\mathcal{P}_4 \circ \ldots \circ \mathcal{P}_4$-RECOGNITION). They show that finding a $P_4$-free $P_4$-transversal (a "$P_4$-free 2-colouring") is $NP$-complete for comparability graphs, $P_5$-free graphs and $(C_4, C_5)$-free graphs.

In [103], we show that the problem of finding a stable $P_4$-transversal remains $NP$-complete when restricted to chordal graphs:

**Theorem 5.1.** [103] *It is $NP$-complete to decide whether a given chordal graph has a stable $P_4$-transversal.*

Using this result, we derive the following consequences:

**Theorem 5.2.** [103] *It is $NP$-complete to decide whether a given chordal graph has a $P_3$-free $P_4$-transversal.*

**Theorem 5.3.** [103] *It is $NP$-complete to decide whether a given chordal graph has a $P_4$-free $P_4$-transversal.*

Additionally, using the same proofs as in the above results, we prove the following strengthening. (A graph $G$ is *strongly chordal* if it is chordal and there exists a perfect elimination ordering $\prec$ of the vertices of $G$ such that if $u \prec v \prec w \prec z$ and $(u, z)$, $(u, w)$ and $(v, w)$ are edges of $G$ then also $(v, z)$ is an edge.)

**Theorem 5.4.** [103]

*(i) It is $NP$-complete to decide whether a strongly chordal graph has a stable $P_4$-transversal.*

*(ii) It is $NP$-complete to decide whether a strongly chordal graph has a $P_3$-free $P_4$-transversal.*

*(iii) It is $NP$-complete to decide whether a strongly chordal graph has a $P_4$-free $P_4$-transversal.*

Furthermore, we have a different proof [104] showing the following for the class of chordal comparability graphs (the graphs that are both chordal and comparability) which is a subclass of strongly chordal graphs.

**Theorem 5.5.**

*(i) It is $NP$-complete to decide whether a chordal comparability graph has a stable $P_4$-transversal.*

*(ii) It is $NP$-complete to decide whether a chordal comparability graph has a $P_3$-free $P_4$-transversal.*

*(iii) Every chordal comparability graph has a $P_4$-free $P_4$-transversal.*

In the theorem above, claims (*i*) and (*ii*) follow the same structure as in Theorems 5.1, 5.2, 5.3, 5.4, while claim (*iii*) is based on the following observation. Let $G$ be an undirected graph, $F$ be an orientation of the edges of $G$, and $\pi = (v_1, v_2, \ldots, v_n)$ be a permutation of the vertices of $G$. We shall call a vertex $v_i$ *transitional* with respect to $F$ and $\pi$, if $v_i$ is neither a source nor a sink in $G[v_1, v_2, \ldots, v_i]$. (That is $v_i$ has both an incoming edge from some $v_j$, $j < i$ and an outgoing edge to some $v_k$, $k < i$.). Let $X_\pi(F)$ be the number of vertices in $G$ that are transitional with respect to $F$ and $\pi$. If $X_\pi(F) = 0$, we call $F$ a *uniform* orientation of $G$ with respect to $\pi$.

**Proposition 5.6.** [104] *There exists a transitive orientation $F$ of a chordal comparability graph $G$ which is uniform with respect to some perfect elimination ordering $\pi$ of $G$.*

Finally, using the same approach as in the above results, we have the following generalisation.

**Theorem 5.7.** [105] *For any $i \leq j$ where $j \geq 4$, it is $NP$-complete to decide whether a given chordal graph has a $P_i$-free $P_j$-transversal.*

## 5.2 Polar Chordal Graphs

A graph $G$ is *polar* if it admits a partition of its vertex set into two subsets $V_r$ and $V_b$, where $V_b$ induces a $P_3$-free graph, and $V_r$ induces a $\overline{P}_3$-free graph. As remarked earlier, we can view this as a colouring of $G$ with two colours, one inducing a $P_3$-free graph and the other inducing a $\overline{P}_3$-free graph, or view it simply as the property $\mathcal{P}_3 \circ \overline{\mathcal{P}}_3$. A graph is $P_3$-free, that is, it has no induced $P_3$ if and only if it is a disjoint union of cliques, with no other edges. (The binary relation 'adjacent or equal' becomes an equivalence relation in this situation.) Thus equivalently, $V_b$ induces a disjoint union of cliques and $V_r$ induces a complete multipartite graph. (As usual, the edges between the two parts are not restricted.) A graph $G$ is *monopolar* if it admits a partition into a $P_3$-free graph and an independent set (the property $\mathcal{O} \circ \mathcal{P}_3$). A graph $G$ is *unipolar* if it admits a partition into a $P_3$-free graph and a clique (the property $\mathcal{K} \circ \mathcal{P}_3$).

For general graphs, the problem of recognising polar graphs [14] and the problem of recognising monopolar graphs are both $NP$-complete [34], whereas there exists a polynomial time algorithm for recognising unipolar graphs [114].

On the other hand, in [31], the authors show that for cographs, the problem of recognising polar graphs, and the problem of recognising graphs that are either monopolar or unipolar are both polynomial time solvable. In fact, in both cases, they give a complete list of minimal forbidden induced subgraphs, and the list is finite.

In [30], we prove similar results for chordal graphs. Note that a chordal graph $G$ is polar if it admits a partition into a $P_3$-free graph and a join of a clique and an independent set. (A complete multipartite graph is chordal iff all but one of the parts are of size one.)

**Theorem 5.8.** [30] *There exists an $O(nm)$ time algorithm to decide, for a given chordal graph, whether it admits a unipolar partition.*

In fact earlier in [53], it was shown that unipolar chordal graphs are precisely the class of $2P_3$-free graph. ($2P_3$ is the disjoint union of two copies of $P_3$.)

**Theorem 5.9.** [30] *There exists an $O(n + m)$ time algorithm to decide, for a given chordal graph, whether it admits a monopolar partition.*

Additionally in [106], we show the complete list of minimal forbidden induced subgraphs for monopolar chordal graphs. This list is not finite, but it can be described using a simple graph grammar (see Section 4.3 and [99]).

**Theorem 5.10.** [106] *There exists a hyperedge replacement grammar $HRG$ that generates all minimal forbidden induced subgraphs for monopolar chordal graphs. There exists a polynomial algorithm to test the membership in $L(HRG)$, the set of graphs generated by $HRG$.*

**Theorem 5.11.** [30] *There exists an $O(n^5)$ time algorithm to decide, for a given chordal graph, whether it admits a polar partition.*

## 5.3 Subcolourings of Chordal Graphs

A *$k$-subcolouring* of a graph $G$ is a partition of the vertices of $G$ into $k$ subsets $V(G) = V_1 \cup \ldots \cup V_k$, such that each $V_i$ induces a disjoint union of cliques, i.e., each $V_i$ induces a $P_3$-free graph. A graph $G$ is called *$k$-subcolourable* if there exists a $k$-subcolouring of $G$. The smallest integer $k$ such that $G$ is $k$-subcolourable is called the *subchromatic number* of $G$, and is denoted by $\chi_s(G)$. We remark that $k$-subcolouring corresponds to the property $\underbrace{\mathcal{P}_3 \circ \ldots \circ \mathcal{P}_3}_{k}$.

The $k$-subcolourings and the subchromatic number were first introduced by Albertson, Jamison, Hedetniemi and Locke in [4]. Initially, the main focus was on bounds for $\chi_s(G)$. More recently, the complexity of recognizing $k$-subcolourable graphs has become a focus of attention. It follows from the result in [1] that for $k \geq 2$ this problem is $NP$-complete for general graphs. In [51] (and also in [56]) the authors show that for $k \geq 2$, it remains $NP$-complete in triangle-free graphs of maximum degree four. On the other hand, there are several natural classes of graphs for which the problem has a polynomial time solution for any fixed $k$, e.g., graphs of bounded treewidth [51]. In another paper [7], the authors show that the problem is $NP$-complete for $k \geq 2$ when restricted to the class of comparability graphs, whereas for interval graphs there is an $O(n^{2k+1})$ time algorithm. They also give an $O(n^{3k+1})$ time algorithm for $k$-subcolouring of permutation graphs, and prove that for chordal graphs, the subchromatic number $\chi_s(G)$ of $G$ is at most $\lfloor \log_2(n+1) \rfloor$.

Finally, the authors of [7] – Broersma, Fomin, Nešetřil and Woeginger – ask the following question: *What is the complexity of the $k$-subcolouring problem in chordal graphs?*

We answer their open question by completely characterising the complexity of the $k$-subcolouring problem (for fixed $k$) in chordal graphs into two possible cases, namely polynomial time solvable for $k \leq 2$ and $NP$-complete for $k \geq 3$.

**Theorem 5.12.** [107] *There exists an $O(n^3)$ time algorithm for deciding (list) 2-subcolourability on chordal graphs; the algorithm also constructs a desired 2-subcolouring if one exists.*

**Theorem 5.13.** [108] *For any $k \geq 3$, it is $NP$-complete to decide whether a chordal graph admits a $k$-subcolouring.*

## 5.4 Injective colourings of Chordal Graphs

An *injective colouring* of a graph $G$ is a colouring $c$ of the vertices of $G$ that assigns different colours to any pair of vertices that have a common neighbour. (That is, for any vertex $v$, if we restrict $c$ to the (open) neighbourhood of $v$, this mapping will be injective; whence the name.) Note that injective colouring is not necessarily a proper colouring, i.e., it is possible for two adjacent vertices to receive the same colour. The injective chromatic number of $G$, denoted $\chi_i(G)$, is the smallest integer $k$ such that $G$ can be injectively coloured with $k$ colours.

Injective colourings were introduced by Hahn, Kratochvíl, Širáň and Sotteau in [62]. They attribute the origin of the concept to complexity theory on Random Access Machines. They prove several interesting bounds on $\chi_i(G)$, and also show that, for $k \geq 3$, it is $NP$-complete to decide whether the injective chromatic number of a graph is at most $k$. Recently, the problem received more attention and several papers dealing particularly with bounds on $\chi_i(G)$ appeared in the literature [29, 63, 85].

In [69], we investigate the complexity of injective colourings in chordal graphs. We have the following results.

**Theorem 5.14.** [69] *It is $NP$-complete for a given split (and hence chordal) graph $G$ and an integer $k$, to decide whether the injective chromatic number of $G$ is at most $k$.*

**Theorem 5.15.** [69] *Unless $NP = ZPP$, for any $\epsilon > 0$, it is not possible to efficiently approximate $\chi(G^2)$ and $\chi_i(G)$ within a factor of $n^{1/3-\epsilon}$, for any split (and hence chordal) graph $G$.*

**Theorem 5.16.** [69] *There exists a polynomial time algorithm that given a split graph $G$ approximates $\chi(G^2)$ and $\chi_i(G)$ within a factor of $\sqrt[3]{n}$.*

In fact, this results corrects a result of Agnarsson et al. [79], who claimed that the chromatic number of the square of a split graph is not $(n^{1/2-\epsilon})$-approximable for all $\epsilon > 0$. Recently, we discovered a result of Král'[75], which implies a generalisation of Theorem 5.16 for all chordal graphs.

**Theorem 5.17.** [75] *Let $G$ be a chordal graph with maximum degree $\Delta \geq 1$. Then:*

$$\chi(G^k) \leq \left\lfloor \sqrt{\frac{91k-118}{384}} (\Delta+1)^{(k+1)/2} \right\rfloor + \Delta + 1 = O(\sqrt{k}\Delta^{(k+1)/2})$$

On the positive side, we prove the following results. Let $\mathcal{B}(G)$ denote the set of bridges of $G$. A graph $G$ is *power chordal*, if all powers of $G$ are chordal. Note that strongly chordal graphs are power chordal.

**Theorem 5.18.** [69] *The injective chromatic number in chordal graphs is fixed parameter tractable. That is, given a chordal graph $G$ and a fixed integer $k$, one can decide in time $O\big(n \cdot k \cdot k^{(k/2+1)^2}\big)$ whether $\chi_i(G) \leq k$ and also whether $\chi(G^2) \leq k$.*

**Theorem 5.19.** [69] *There exists an $O(n+m)$ time algorithm that computes $\chi_i(G)$ given a chordal graph $G$ and $\chi((G-\mathcal{B}(G))^2)$. Using this algorithm one can also construct an optimal injective colouring of $G$ from an optimal colouring of $(G-\mathcal{B}(G))^2$ in time $O(n+m)$.*

**Corollary 5.20.** [69] *The injective chromatic number of a power chordal (strongly chordal) graph can be computed in polynomial time.*

## 5.5 Summary and future directions

In Table 5.1, we summarise the results from the previous sections with other known results about the complexity of partitioning general and chordal graphs in some small cases. The entry in column $X$ and row $Y$ in the table indicates the complexity of graph partitioning into two parts, where one part is $X$-free and the other $Y$-free, both in general graph and in chordal graphs. Here $P$ stands for polynomial time and $N$ stands for $NP$-complete. A fraction $\frac{P}{N}$ indicates that the problem is $NP$-complete in general but polynomial time solvable in chordal graphs, whereas an entry with $P$ respectively $N$ indicates a polynomial time solvable respectively $NP$-complete problem in both classes.

As one would expect, some problems difficult in general graphs are polynomial time solvable in chordal graphs, while other ones remain $NP$-complete. As in the case of general

| $X\backslash Y$ | $K_2$ | $\overline{K_2}$ | $P_3$ | $\overline{P_3}$ | $K_k$ | $\overline{K_k}$ | $P_4$ | $P_i$ |
|---|---|---|---|---|---|---|---|---|
| $K_2$ | $P$ | | | | | | | |
| $\overline{K_2}$ | $P$ | $P$ | | | | | | |
| $P_3$ | $\frac{{}^{1}P}{N}$ | ${}^{2}P$ | $\frac{{}^{3}P}{N}$ | | | | | |
| $\overline{P_3}$ | ${}^{2}P$ | $\frac{{}^{4}P}{N}$ | $\frac{{}^{1}P}{N}$ | $\frac{{}^{4}P}{N}$ | | | | |
| $K_l$ | $\frac{{}^{5}P}{N}$ | ${}^{6}P$ | $\frac{{}^{7}P}{N}$ | $\frac{{}^{5}P}{{}^{b}N}$ | $\frac{{}^{5}P}{N}$ | | | |
| $\overline{K_l}$ | ${}^{6}P$ | $\frac{{}^{5}P}{N}$ | $\frac{{}^{a}P}{{}^{b}N}$ | $\frac{{}^{5}P}{N}$ | ${}^{6}P$ | $\frac{{}^{5}P}{N}$ | | |
| $P_4$ | ${}^{8}N$ | $\frac{{}^{a}P}{N}$ | ${}^{8}N$ | ${}^{b}N$ | ${}^{8}N$ | $\frac{{}^{a}P}{N}$ | ${}^{8}N$ | |
| $P_j$ | ${}^{9}N$ | $\frac{{}^{a}P}{N}$ | ${}^{9}N$ | ${}^{b}N$ | ${}^{9}N$ | $\frac{{}^{a}P}{N}$ | ${}^{9}N$ | ${}^{9}N$ |

$P \equiv$ polytime, $N \equiv NP$-complete, $\frac{P}{N} \equiv \frac{\text{P in chordal graphs}}{\text{NPc in general graphs}}$

Notes: $k, l \geq 3$ and $i, j \geq 4$
[1] [30] Theorems 5.9, 5.11
[2] [114] Theorem 5.29
[3] [107] Theorem 5.12
[4] [46] Theorem 3.21
[5] [47] Theorem 3.23
[6] [3] Theorem 2.4
[7] Theorem 5.25
[8] [103] Theorems 5.1, 5.2, 5.3
[9] [105] Theorem 5.7
[a] Theorem 5.27
[b] Theorem 5.28

**Table 5.1**: Summary of complexity results of $\{X\text{-free}\}\circ\{Y\text{-free}\}$-RECOGNITION in chordal graphs and general graphs

graphs (cf. Theorems 2.2 and 2.3), one would hope to obtain a complete characterisation of complexity of partitioning problems for chordal graphs, namely for reducible additive induced-hereditary properties, respectively for compositions of additive and co-additive induced-hereditary properties. In this section, we try to propose some possible generalisations leading to a dichotomy for the aforementioned types of properties in chordal graphs. The evidence we gathered suggests the following conjectures.

**Conjecture 5.21.** *For any reducible additive induced hereditary property $\mathcal{P}$, the problem $\mathcal{P}$-RECOGNITION in chordal graphs is in $P$ or it is $NP$-hard.*

**Conjecture 5.22.** *For any property $\mathcal{P}$ which is a composition of an additive and a co-additive induced hereditary property, the problem $\mathcal{P}$-RECOGNITION in chordal graphs is in $P$ or it is $NP$-hard.*

For instance, one could easily see, by examining the proof of Theorem 5.1, that this proof heavily relies on the fact that $P_4$-free graphs are closed under the operation of join. (Actually, in chordal graphs, this is equivalent to be closed under adding a dominating vertex.) It appears that this property should be sufficient to render some partitioning problems $NP$-complete, namely the following.

**Conjecture 5.23.** *For any connected graph $H$ without a dominating vertex, the problem of stable $H$-transversal in chordal graphs is $NP$-complete.*

The evidence suggests also the following strengthening.

**Conjecture 5.24.** *Let $H$ be a connected graph. Then stable $H$-transversal of chordal graphs is $NP$-complete if $H$ itself is chordal and contains a $P_4$, and polynomial time solvable otherwise.*

On the other hand, we can prove the following.

**Theorem 5.25.** *For any fixed 0-diagonal, or 1-diagonal matrix $M$ with entries $\{0, 1, *\}$, the problem of $M$-partitionable $P_3$-transversal in chordal graphs is polynomial time solvable.*

By a similar argument, it should be possible to prove the following.

**Conjecture 5.26.** *For any fixed 0-diagonal, or 1-diagonal matrix $M$ with entries $\{0, 1, *\}$, the problem of partitioning the vertex set of a chordal graph $G$ into sets $V_1, V_2, V_3$ where $G[V_1]$ is $M$-partitionable and $G[V_2]$ and $G[V_3]$ are both $P_3$-free is polynomial time solvable.*

Finally, for the sake of completeness and to justify Table 5.1, we include the following claims which are easy to observe.

**Theorem 5.27.** *For any $k \geq 0$ and for any graph property $\mathcal{P}$, which is polynomial time recognisable in chordal graphs, one can decide in polynomial time, whether a given chordal graph admits a partition into a $\overline{K_k}$-free graph and a graph in $\mathcal{P}$.*

A special case of the following result has been shown in [3].

**Theorem 5.28.** *For any additive (induced) hereditary property $\mathcal{P}$ and any graph $H$, if it is $NP$-complete to decide whether a graph admits a partition into an $H$-free graph and a graph from $\mathcal{P}$, it is also $NP$-complete to decide whether a graph admits a partition into a $(H \uplus K_1)$-free graph and a graph from $\mathcal{P}$.*

**Theorem 5.29.** [114] *There exists a polynomial time algorithm that decides, for a given graph $G$, whether $G$ admits a partition into a clique and a $P_3$-free graph.*

# Bibliography

[1] D. Achlioptas, *The complexity of G-free colorability*, Discrete Mathematics 165/166 (1997) 21–30

[2] G. Agnarsson, R. Greenlaw, M. Halldorsson: *On Powers of Chordal Graphs and Their Colorings*, Congress Numerantium 100 (2000) 41–65.

[3] V. E. Alekseev, A. Farrugia, V. V. Lozin: *New results on generalized graph colorings*, Discrete Mathematics and Theoretical Computer Science 6 (2004), 215–222.

[4] M. O. Albertson, R. E. Jamison, S. T. Hedetniemi, S. C. Locke, *The subchromatic number of a graph*, Discrete Mathematics 74 (1989) 33–49

[5] G. Arkit, A. Farrugia, P. Mihók, G. Semanišin, R. Vasky: *Additive induced-hereditary properties and unique factorization*, arXiv:math/0306166v1 [math.CO] `http://arxiv.org/abs/math/0306166`

[6] C. Berge: *Färbung von Graphen deren sämtliche bzw. deren ungerade Kreise starr sind*, Wiss. Zeitschrift, Martin-Luther-Univ. Halle-Wittenberg, Math.-Natur. Reihe 10 (1961), 114–115.

[7] H. Broersma, F. V. Fomin, J. Nešetřil, G. J. Woeginger: *More about subcolorings*, Computing 69 (2002) 187–203.

[8] J. I. Brown, D. G. Corneil: *On generalized graph colorings*, Journal of Graph Theory 11 (1987) 87–99.

[9] A. A. Bulatov: *H-Coloring Dichotomy Revisited*, Theoretical Coputer Science 349 (2005), 31–39.

[10] A. A. Bulatov: *Tractable Conservative Constraint Satisfaction problems*, ACM Transactions on Computational Logic, to appear.

[11] A. A. Bulatov: *The complexity of the Counting Constraint Satisfaction Problem*, manuscript.

[12] A. A. Bulatov, V. Dalmau: *Mal'tsev constraints are tractable*, SIAM Journal on Computing 36 (2006), 16–27.

[13] K. Cameron, E. M. Eschen, C. T. Hoàng, R. Sritharan: *The List Partition Problem for Graphs*, In: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms (SODA 2004), 391–399.

[14] Z. A. Chernyak, A. A. Chernyak: *About recognizing (a,b)-classes of polar graphs*, Discrete Mathematics 62 (1986) 133–138.

[15] M. Chudnovsky, N. Robertson, P. Seymour, R. Thomas: *The Strong Perfect Graph Theorem*, Annals of Mathematics 164 (2006), 51–229.

[16] B. Courcelle: *On the expression of monadic second-order graph properties without quantifications over sets of edges*, In: Fifth Annual IEEE Symposium on Logic in Computer Science (LICS), 1990, 190–196

[17] B. Courcelle: *The Monadic Second-Order Logic of Graphs. I: Recognizable Sets of Finite Graphs*, Information and Computation 85 (1990), 12–75.

[18] B. Courcelle: *The Monadic Second-Order Logic Of Graphs III: Tree-decompositions, Minor and Complexity Issues*, Informatique Théoretique et Applications 26 (1992), 257–286.

[19] B. Courcelle, J. Engelfriet: *A logical characterization of the sets of hypergraphs defined by hyperedge replacement grammars*, Mathematical Systems Theory 28 (1995), 515–522.

[20] D. G. Corneil: *Lexicographic Breadth First Search – A Survey*, In: Graph-Theoretic Concepts in Computer Science (WG 2004), LNCS 3353, Springer, 2004, 1–19.

[21] D. G. Corneil, H. Lerchs, L. Stewart: *Complement Reducible Graphs*, Discrete Applied Mathematics 3 (1981), 163–174.

[22] D. G. Corneil, R. Krueger: *A Unified View of Graph Searching*, SIAM Journal on Discrete Mathematics (2007), to appear.

[23] A. Cournier, M. Habib: *A new linear algorithm for modular decomposition*, In: Trees in algebra and programming (CAAP 94), 19th International Colloquium, Edinburgh, U.K., Lecture Notes in Computer Science 787, Springer, 1994, 68–84.

[24] R. Diestel: **Graph Theory**, 3rd Edition, Graduate Texts in Mathematics 173, Springer-Verlag, Heidelberg, 2005.

[25] G. A. Dirac: *On rigid circuit graphs*, Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg 25 (1961), 71–76.

[26] F. Drewes: *Recognising k-connected hypergraphs in cubic time*, Theoretical Computer Science 109 (1993), 83–122.

[27] F. Drewes: *NP-completeness of k-connected hyperedge-replacement languages of order k*, Information Processing Letters 45 (1993), 89–94.

[28] R. Dechter: *Constraint networks*, In: Encyclopedia of Artificial Intelligence, 1992.

[29] A. Doyon, G. Hahn, A. Raspaud: *On the injective chromatic number of sparse graphs*, preprint 2005.

[30] T. Ekim, P. Hell, J. Stacho, D. de Werra: *Polarity of Chordal Graphs*, to appear in Discrete Applied Mathematics.

[31] T. Ekim, N.V.R. Mahadev, D. de Werra: *Polar cographs*, to appear.

[32] M. Farber: *Characterizations of strongly chordal graphs*, Discrete Mathematics 43 (1983), 173–189.

[33] M. Farber: *Domination, independent domination, and duality in strongly chordal graphs*, Discrete Applied Mathematics 7 (1984), 115–130.

[34] A. Farrugia: *Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard*, Electronic Journal on Combinatorics 11 (2004), #R46.

[35] A. Farrugia, R. B. Richter: *Unique factorisation of additive induced-hereditary properties*, Discussiones Mathematicae Graph Theory 24 (2004), 319–343.

[36] A. Farrugia, P. Mihók, R. B. Richter, G. Semanišin: *Factorizations and characterizations of induced-hereditary and compositive properties*, Journal of Graph Theory 49 (2005), 11-27.

[37] A. Farrugia: *Uniqueness and complexity in generalised colouring*, Ph.D. Thesis, University of Waterloo, 2003, `http://etd.uwaterloo.ca/etd/afarrugia2003.pdf`

[38] T. Feder, P. Hell: *List homomorphisms to reflexive graphs*, Journal of Combinatorial Theory B 72 (1998), 236 - 250.

[39] T. Feder, P. Hell: *Matrix partitions of perfect graphs*, Discrete Mathematics 306 (2006), 2450–2460.

[40] T. Feder, P. Hell: *Full Constraint Satisfaction Problems*, SIAM Journal on Computing 36 (2006), 230–246.

[41] T. Feder, P. Hell: *On realizations of point determining graphs, and obstructions to full homomorphisms*, Discrete Mathematics (2007), to appear.

[42] T. Feder, P. Hell, W. Hochstättler: *Generalized Colourings (Matrix Partitions) of Cographs*, manuscript.

[43] T. Feder, P. Hell, J. Huang: *List homomorphisms and circular arc graphs*, Combinatorica 19 (1999), 487–505

[44] T. Feder, P. Hell, J. Huang: *Bi-arc graphs and the complexity of list homomorphisms*, Journal of Graph Theory 42 (2003), 61–80

[45] T. Feder, P. Hell, S. Klein, R. Motwani: *Complexity of graph partition problems*, 31st Annual ACM STOC (1999) 464–472.

[46] T. Feder, P. Hell, S. Klein, R. Motwani: *List Partitions*, SIAM Journal on Discrete Mathematics 16 (2003), 449–478.

[47] T. Feder, P. Hell, S. Klein, L.T. Nogueira, F. Protti: *List matrix partitions of chordal graphs*, Theoretical Computer Science 349 (2005) 52–66.

[48] T. Feder, P. Hell, W. Xie: *Matrix Partitions with Finitely Many Obstructions*, manuscript.

[49] T. Feder, M. Y. Vardi: *The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory*, SIAM Journal on Computing 28 (1999), 57–104.

[50] U. Feige, J. Killian: *Zero Knowledge and the Chromatic Number*, Journal of Computer and System Sciences 57 (1998), 187–199.

[51] J. Fiala, K. Jansen, V. B. Le, E. Seidel: *Graph subcoloring: Complexity and algorithms*, In: Graph-Theoretic Concepts in Computer Science (WG 2001), LNCS 2204, Springer, 2001, 154–165.

[52] D. R. Fulkerson, O. A. Gross: *Incidence matrices and interval graphs*, Pacific Journal of Mathematics 15 (1965), 835–855.

[53] A.V. Gagarin: *Chordal* $(1,\beta)$*-polar graphs*, Vestsi Nats. Akad. Navuk Belarusi Ser. Fiz.-Mat. Navuk (1999) 115–118.

[54] T. Gallai: *Transitiv orientierbare Graphen*, Acta Mathematica Academiae Scientiarum Hungaricae 18 (1967), 25–66.

[55] F. Gavril: *The intersection graphs of subtrees in trees are exactly the chordal graphs*, Journal of Combinatorial Theory B 16 (1974), 47–56.

[56] J. Gimbel, C. Hartman: *Subcolorings and the subchromatic number of a graph*, Discrete Mathematics 272 (2003) 139–154.

[57] M. C. Golumbic: **Algorithmic Graph Theory and Perfect Graphs**, Academic Press, New York, 1980.

[58] M. C. Golumbic, R. E. Jamison: *The edge intersection graphs of paths in a tree*, Journal of Combinatorial Theory B 38 (1985), 8–22.

[59] M. Grötschel, L. Lovász, A. Schrijver: *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica 1 (1981), 169–197; Corrigendum Combinatorica 4 (1984), 291–295.

[60] G. Gutin, P. Hell, A. Rafiey, A. Yeo: *A Dichotomy for Minimum Cost Graph Homomorphisms*, manuscript, to appear in European Journal of Combinatorics.

[61] M. Habib, Ch. Paul: *A simple linear time algorithm for cograph recognition*, Discrete Applied Mathematics 145 (2005), 183–197.

[62] G. Hahn, J. Kratochvíl, J. Širáň, D. Sotteau: *On the injective chromatic number of graphs*, Discrete Mathematics 256 (2002) 179–192.

[63] G. Hahn, A. Raspaud, W. Wang: *On the injective coloring of $K_4$-minor free graphs*, manuscript.

[64] P. Hell: *From Graph Colouring to Constraint Satisfaction: There and Back Again*, In: Topics in Discrete Mathematics, Dedicated to Jarik Nešetril on the Occasion of his 60th birthday, Algorithms and Combinatorics, Vol. 26 (2006), 407–432

[65] P. Hell, J. Nešetřil: *On the complexity of $H$-colouring*, Journal of Combinatorial Theory B 48 (1990), 92–110.

[66] P. Hell, J. Nešetřil: **Graphs and Homomorphisms**, Oxford University Press, 2004.

[67] P. Hell, S. Klein, L.T. Nogueira, F. Protti: *Partitioning chordal graphs into independent sets and cliques*, Discrete Applied Mathematics 141 (2004) 185 – 194.

[68] P. Hell, S. Klein, L.T. Nogueira, F. Protti: *Packing r-cliques in weighted chordal graphs*, Annals of Operations Research 138 (2005), 179–187.

[69] P. Hell, A. Raspaud, J. Stacho: *On Injective Colourings of Chordal Graphs*, manuscript.

[70] C. T. Hoàng, V. B. Le: *On $P_4$-transversals of perfect graphs*, Discrete Mathematics 216 (2000), 195–210.

[71] C. T. Hoàng, V. B. Le: *$P_4$-free Colorings and $P_4$-Bipartite Graphs*, Discrete Mathematics and Theoretical Computer Science 4 (2001), 109–122.

[72] W. L. Hsu, T. H. Ma: *Fast and Simple Algorithms for Recognizing Chordal Comparability Graphs and Interval Graphs*, SIAM Journal on Computing 28 (1999), 1004–1020.

[73] T. R. Jensen, B. Toft.: **Graph coloring problems**, Wiley Interscience Series in Discrete Mathematics, Wiley, New York, 1995.

[74] P. Jeavons: *On the algebraic structure of combinatorial problems*, Theoretical Computer Science 200 (1998), 185–204.

[75] D. Král̆: *Coloring Powers of Chordal Graphs*, SIAM Journal on Discrete Mathematics 18 (2005), 451–461.

[76] J. Kratochvíl, P. Mihók: *Hom-properties are uniquely factorizable into irreducible factors*, Discrete Mathematics 213 (2000), 189–194.

[77] V. Kumar: *Algorithms for constraint-satisfaction problems*, AI Magazine 13 (1992), 32–44.

[78] G. Kun: *Constraints, MMSNP and expander relational structures*, arXiv:0706.1701v1 [math.CO] `http://arxiv.org/abs/0706.1701v1`

[79] R. Laskar, D. Shier: *On powers and centers of chordal graphs*, Discrete Applied Mathematics 6 (1983), 139–147.

[80] R. E. Ladner: *On the structure of polynomial time reducibility*, Journal of the ACM 22 (1975), 155–171.

[81] K.-J. Lange, E. Welzl: *String grammars with disconnecting or a basic root of the difficulty in graph grammar parsing*, Discrete Applied Mathematics 16 (1987), 17–30.

[82] C. Lautemann: *The complexity of graph languages generated by hyperedge replacement*, Acta Informatica 27 (1990), 399–421.

[83] B. Larose, L. Zádori: *Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras*, International Journal of Algebra and Computation 16 (2006), 563–581.

[84] C. G. Lekkerkerker, J. Ch. Boland: *Representation of a finite graph by a set of intervals on the real line*, Fundamenta Mathematicae 51 (1962), 45–64.

[85] B. Lužar, Riste Škrekovski, M. Tancer: *Injective colorings of planar graphs with few colors*, manuscript.

[86] M. Maróti and R. McKenzie: *Existence theorems for weakly symmetric operations*, Algebra Universalis 2007, to appear.

[87] R. M. McConnell, J. Spinrad: *Linear-time modular decomposition and efficient transitive orientation of comparability graphs*, In: Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1994), 536–545.

[88] R. M. McConnell, F. de Montgolfier: *Linear-time modular decomposition of directed graphs*, Discrete Applied Mathematics 145 (2005), 189–209.

[89] P. Meseguer: *Constraint satisfaction problem: an overview*, AICOM 2 (1989), 3–16.

[90] P. Mihók, G. Semanišin: *Reducible properties of graphs*, Discussiones Mathematicae Graph Theory 15 (1995), 11–18.

[91] P. Mihók, G. Semanišin, R. Vasky: *Additive and hereditary properties of graphs are uniquely factorizable into irreducible factors*, Journal of Graph Theory 33 (2000), 44–53.

[92] P. Mihók: *Unique Factorization Theorem*, Discussiones Mathematicae Graph Theory 20 (2000), 143–153.

[93] J. H. Muller, J. P. Spinrad, *Incremental Modular Decomposition*, Journal of the ACM 36 (1989), 1–19.

[94] J. Nešetřil, M. H. Siggers: *A new combinatorial approach to the constraint satisfaction problem dichotomy classification*, submitted.

[95] N. Robertson, D. P. Sanders, P. Seymour, R. Thomas: *Efficiently four-coloring planar graphs*, Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (STOC), 571–575.

[96] N. Robertson, P. Seymour: *Graph Minors. XX. Wagner's conjecture*, Journal of Combinatorial Theory B 92 (2004), 325–357.

[97] D. J. Rose: *Triangulated graphs and the elimination process*, Journal of Mathematical Analysis and Applications 32 (1970), 579–609.

[98] D. J. Rose, R. E. Tarjan, G. S. Leuker: *Algorithmic aspects of vertex elimination on graphs*, SIAM Journal on Computing 5 (1976), 266–283.

[99] G. Rozenberg: **Handbook of Graph Grammars and Computing by Graph Transformations**, Foundations World Scientific, vol. 1, 1997.

[100] T. J. Shaefer: *The complexity of satisfiability problems*, Proceedings of 10th ACM Symposium on Theory of Computing (1978), 216–226.

[101] J. P. Spinrad: *Two dimensional partial orders*, Ph. D. thesis, Department of Computer Science, Princeton University (1982).

[102] J. P. Spinrad: **Efficient Graph Representations**, American Mathematical Society, 2003.

[103] J. Stacho: *On $P_4$-transversals of Chordal Graphs*, Discrete Mathematics, *to appear.*

[104] J. Stacho: *On $P_4$-transversals of Chordal Comparability Graphs*, manuscript.

[105] J. Stacho: *On $H$-transversals of Chordal Graphs*, manuscript.

[106] J. Stacho: *Forbidden induced subgraph characterisation of polarity in Chordal Graphs*, manuscript.

[107] J. Stacho: *On 2-Subcolourings of Chordal Graphs*, manuscript.

[108] J. Stacho: *On k-Subcolourings of Chordal Graphs*, manuscript.

[109] R. E. Tarjan: *Decomposition by clique separators*, Discrete Mathematics 55 (1985), 221–232.

[110] R. E. Tarjan: *Depth first search and linear graph algorithms*, SIAM Journal on Computing, 1 (1972) 146–160.

[111] R. E. Tarjan: *Maximum cardinality search and chordal graphs*, Stanford University, Unpublished Lecture Notes CS 259.

[112] R. E. Tarjan, M. Yannakakis: *Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM Journal on Computing 13 (1984) 566–579.

[113] R. E. Tarjan, M. Y. Yannakakis: *Addendum: simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM Journal on Computing 14 (1985) 254–255.

[114] R. I. Tyshkevich, A. A. Chernyak: *Algorithms for the canonical decomposition of a graph and recognizing polarity*, Izvestia Akad. Nauk BSSR, ser. Fiz. Mat. Nauk. 6 (1985) 16–23, (in Russian).

[115] D. West: **Introduction to Graph Theory**, Prentice Hall, 1996.

[116] S. H. Whitesides: *An algorithm for finding clique cut-sets*, Information Processing Letters 12 (1981), 31–32.