Masaryk University
Faculty of Informatics

Parameterized Algorithms for Geometric Intersection Graphs

Ph.D. Thesis

RNDr. Deniz Ağaoğlu Çağırıcı

Supervisor: Prof. RNDr. Petr Hliněný, Ph.D.

Brno, September 2023

Declaration

I hereby declare that this thesis is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during the elaboration of this work are properly cited and listed in completed reference to the due source.

RNDr. Deniz Ağaoğlu Çağırıcı

Acknowledgements

I want to take a moment to express my heartfelt gratitude to the wonderful people who have been by my side throughout this thesis journey. Your unwavering support, encouragement, and belief in me have been incredibly meaningful, and I want to express my deepest appreciation to each of you.

To my fantastic supervisor, Petr Hliněný, your guidance, patience and dedication to academic excellence have shaped this thesis, and your commitment to my progress has been a constant source of motivation, even during the most challenging times. You will always be the best boss I will have. Thanks for being my supervisor, which was practically the Olympic gold of my academic adventure in Brno!

To Tomasz Krawczyk and Pascal Schweitzer, I would like to thank both of you for giving me the opportunity to conduct research visits under your guidance and your exceptional hospitality during my stays. Beyond the enriching research experiences, I am truly appreciative of the delightful conversations and their introduction to fellow researchers, which made my time memorable and inspiring.

To my amazing friends, your jokes and hangouts made the tough times easier. Your presence and unwavering belief in my abilities have been a constant source of inspiration. Thanks for being there when I needed a break from all the serious stuff.

To my mom, dad and little brother, you're my rock! Your boundless love, sacrifices and belief in me pushed me forward, even in moments of self-doubt. I owe an immeasurable debt of gratitude for your constant support and encouragement. Thank you for calming my nerves before every presentation, even at those ungodly hours.

To my parent-in-laws, your warm support and belief in my aspirations have meant a lot to me. Your encouraging cheers have made this journey all the more meaningful and made me feel like part of the family from day one. Thank you for embracing me.

To my partner in crime, Onur, you deserve a medal for putting up with my thesis induced craziness and pre-presentation jitters even from miles away as if they were a piece of cake. Your consistent encouragement and endless patience have been my greatest motivators. Your belief in my capabilities has been my guiding light, and I am eternally grateful for your presence in my life. Thanks for being my anchor!

Abstract

This thesis immerses itself in the landscape of graph theory, with a central focus on the parameterized complexity of the graph isomorphism and class recognition problems on H-graphs, introduced in 1992 by Biró, Hujter and Tuza. H-graphs represent a particular type of intersection representations of graphs, and they naturally generalize many graph classes including interval, circular-arc, split and chordal graphs.

By introducing efficient algorithms solving the automorphism problem for colored set families and colored clique families of interval graphs, and using existing group theoretical tools, we show that H-graph isomorphism can be tested in `FPT`-time when H is a tree. We provide complementary and fully combinatorial `FPT`-approaches to test the isomorphism of proper H-graphs when H is a tree or a unicyclic graph.

We further consider H-graphs when H is a unicyclic graph, and prove that H-graphs can be recognized in `XP`-time when the input graphs are non-chordal, and proper H-graphs can be recognized in `FPT`-time in general. We establish polynomial time recognition algorithms for Helly H-graphs for unicyclic H, and H-graphs when H consists of a cycle and a path attached to that cycle.

Abstrakt

Tato práce se zabývá oblastí teorie grafů s ústředním zaměřením na parametrizovanou složitost problému izomorfismu a problému rozpoznávání na třídách H-grafů, které byly zavedené v roce 1992 Biróem, Hujterem a Tuzou. H-grafy představují zvláštní typ reprezentace průnikových grafů a přirozeně zobecňují mnoho tříd grafů včetně intervalových, kruhově intervalových, split a chordálních grafů.

Za použití efektivních algoritmů řešících problém automorfismu pro barvené množinové systémy a barvené systémy klik intervalových grafů a existujících algebraických grupově-teoretických rutin ukážeme, že izomorfismus H-grafu může být testován v čase `FPT`, pokud H je strom. Doplňkově podáváme plně kombinatorické `FPT`-přístupy k testování izomorfismu vlastních H-grafů, když H je strom nebo unicyklický graf.

Dále uvažujeme H-grafy, kde H je unicyklický graf. Dokážeme, že tyto H-grafy lze rozpoznat v čase `XP` za podmínky, že jsou vstupní grafy nechordální, přičemž takové vlastní H-grafy lze obecně rozpoznat v čase `FPT`. Podáme polynomiální algoritmy na rozpoznávání Hellyho H-grafů pro unicyklický H, a H-grafů, jejichž H sestává z cyklu a cesty připojené k tomuto cyklu.

Contents

1 Introduction **3**
- 1.1 Parameterized complexity 3
- 1.2 Basic graph terms 5
- 1.3 Intersection graphs 7
- 1.4 Outline of the thesis 13

2 Algebraic Tools **18**
- 2.1 Automorphism groups of graphs 18
- 2.2 Babai's tower-of-groups 19
- 2.3 Bounded color multiplicity graph isomorphism 20
- 2.4 Partially ordered sets (posets) 21
- 2.5 PQ-trees of interval graphs 24
- 2.6 Some parameterized results on graph isomorphism 26
- 2.7 GI-completeness reduction to $\boldsymbol{H}$-graphs when $\boldsymbol{H}$ contains at least two cycles sharing a vertex 26
- 2.8 $\boldsymbol{S_d}$-graphs, $\boldsymbol{T}$-graphs, $\boldsymbol{L}$-graphs, $\boldsymbol{U}$-graphs, Helly $\boldsymbol{H}$-graphs and Proper $\boldsymbol{H}$-graphs 28

3 Automorphisms of Colored Set Families and Colored Cliques of Interval Graphs **35**
- 3.1 AutomSET$(\boldsymbol{X}; \boldsymbol{\mathcal{U}}^{[m]})$ in FPT 38
- 3.2 AutomMarkedINT$(\boldsymbol{G}; \boldsymbol{\mathcal{A}}^{[m]})$ in FPT 46

4 $\boldsymbol{S_d}$-graph Isomorphism **52**
- 4.1 Isomorphism of $\boldsymbol{S_d}$-graphs of bounded clique size 53
- 4.2 $\boldsymbol{S_d}$-graphs to posets of bounded width 56
- 4.3 Isomorphism of $\boldsymbol{S_d}$-graphs 59

5 Proper $\boldsymbol{S_d}$-graph Isomorphism and Proper $\boldsymbol{T}$-graph Isomor-

phism **66**
5.1 Isomorphism of proper S_d-graphs 66
5.2 Isomorphism of proper T-graphs 67
5.3 T-graph isomorphism in XP-time 71

6 T-graph Isomorphism **74**
6.1 Structure of chordal graphs 74
6.2 Canonical decomposition of T-graphs 76
6.3 Isomorphism of T-graph decompositions 82
6.4 Isomorphism of chordal graphs parameterized by leafage and vertex leafage 87

7 Proper U-graph Recognition and Isomorphism **89**
7.1 NP-hardness of $\mathcal{U}$-graph and proper $\mathcal{U}$-graph recognition . . . 90
7.2 Recognizing proper chordal and proper Helly U-graphs . . . 93
7.3 Recognizing proper U-graphs in general 102
7.4 Isomorphism of proper U-graphs 112
7.5 GI-completeness for H-graphs and proper H-graphs 115

8 Recognizing L-graphs, strictly cyclic U-graphs, and Helly $\mathcal{U}$-graphs **119**
8.1 NP-completeness reduction to H-graphs when H contains at least two cycles sharing a vertex 119
8.2 Recognizing L-graphs 123
8.3 Recognizing strongly cyclic U-graphs and Helly $\mathcal{U}$-graphs . . 139

9 A Purely Combinatorial Reduction for Set-Family Automorphisms **143**

10 Conclusions **148**
10.1 Future work and open problems 148
10.2 Author's publications 152

Bibliography **153**

Chapter 1

Introduction

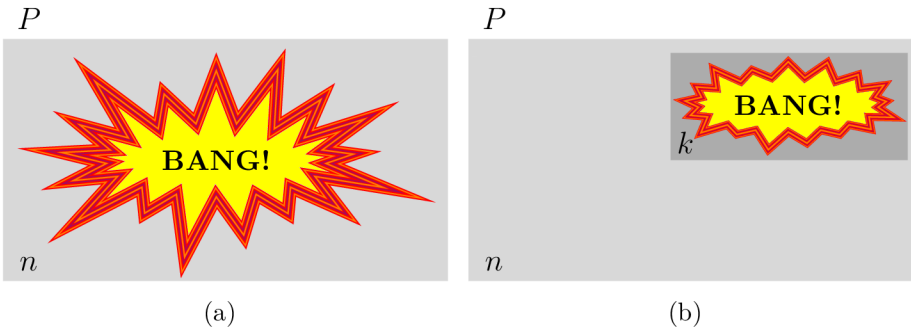
We consider selected intersection graph classes related to geometric objects, and study on them the parameterized complexity of various problems which are assumed hard in general.

In this chapter, we first introduce parameterized complexity and explain why the parameterization is useful. Then, we define the geometric intersection graphs which are the subject of this thesis, and review literature for two main problems that we focus on, namely the class recognition and the graph isomorphism problems. We assume that the readers have basic knowledge of algorithms and classical complexity, on the level of standard bachelor courses.

1.1 Parameterized complexity

The complexity class NP represents the set of all decision problems whose positive solutions can be verified in polynomial time. A problem is called NP-hard if it is at least as hard as any other problem in NP. Besides efficiently solvable problems, many common problems are NP-hard and it is believed that (though unknown) there exists no polynomial time algorithm for any NP-hard problem. To cope with hardness of such problems, several approaches have been introduced such as designing heuristics, approximation and randomized algorithms. The parameterized complexity theory is another approach motivated by the hard problems and it aims to analyse NP-hard problems in a finer detail. The main idea is to investigate the tractable instances (or parameters) of NP-hard problems and express the complexity

as a function of the input size and some parameter of the input instead of only measuring the complexity by the input size. Figure 1.1a shows an NP-hard problem P which suffers from combinatorial explosion leading to usually exponential running time of any algorithm solving P correctly, and Figure 1.1b shows the goal behind parameterized complexity which is to confine the combinatorial explosion to a parameter (or multiple parameters) which might be small and dealt efficiently with parameterized algorithms in practice.



(a) (b)

Figure 1.1: An illustration of the key idea behind parameterized complexity[1].

Two important classes of parameterized problems are FPT and XP which we briefly explain here. Let n be the input size, k be the parameter, f and g be arbitrary computable functions depending only on k, and c be a constant. Then, a problem is called *fixed parameter tractable*, shortly FPT, if there exists an algorithm (called an FPT-algorithm) solving this problem in time $O(f(k) \cdot n^c)$. Likewise, a problem is called XP if there exists an algorithm (called an XP-algorithm) solving this problem in time $O(f(k) \cdot n^{g(k)})$. Note that both these classes are asymptotically strictly better than the algorithms that are exponential in n for bounded parameter values. Since the exponent of n depends on the parameter in XP-algorithms in contrast to FPT-algorithms, XP-algorithms are arguably less efficient and thus, we would like to design FPT-algorithms in favour of XP ones.

When comparing parameterized algorithms, the efficiency depends on the parameter, i.e., the complexity of the functions f and g of the parameter in addition to the constant c. For instance, choosing a parameter such that f and k grow slowly and c is as small as possible results in a more

[1]This figure was inspired by the logo of the website fpt.wikidot.com.

efficient parameterized algorithm. Moreover, for the same problem, there can be different parameters leading to different complexities and not all parameters lead to efficient algorithms. Therefore, it is important to select a good parameterization. For instance, if the chosen parameter is not small in practice, then the parameterization becomes too restrictive. It is also possible to consider multiple parameters instead of a single parameter, and then, the complexity depends on a combination of several parameters. The perhaps most common and useful parameters are the expected solution value (in case of optimization problems), and structural parameters such as tree-width, rank-width, maximum degree, clique size and/or vertex cover.

1.2 Basic graph terms

A *graph* is a pair $G = (V, E)$ where $V = V(G)$ is a finite vertex set and $E = E(G) \subseteq V(G) \times V(G)$ is the edge set. The *order* $|V(G)|$ of a graph G is the number of vertices in $V(G)$, and similarly, the number of edges is denoted by $|E(G)|$. For an edge $e = \{u, v\}$, u and v are called the *end-vertices*, or shortly the *ends* of e.

A graph is called *directed* if its edges are ordered (bidirectional), and *undirected*, otherwise. A *weighted graph* is a graph in which each edge is assigned a numerical value called its *weight*, and an *unweighted graph* is a graph where no edge has a weight.

A *loop* in a graph is an edge which joins a vertex to itself. *Parallel edges* in a graph are the edges joining the same end-vertices and a graph containing parallel edges is called a *multigraph*. On the other hand, a graph containing no loops and no parallel edges is called *simple*. A *finite graph* is a graph which has finite number of vertices and edges, and unless stated otherwise, we focus on unweighted, undirected and finite simple graphs in this thesis.

Two vertices $u, v \in V(G)$ are called *adjacent* if and only if $\{u, v\} \in E(G)$, and the edge $e = \{u, v\}$ is called *incident* to u and v.

For a vertex $v \in V(G)$, the set of all vertices that v is adjacent to is called the *(open) neighborhood* of v, does not contain v itself and denoted by $N_G(v)$. The set $N_G(v) \cup v$ is called the *closed neighborhood* of v and denoted by $N_G[v]$. The number of vertices in $N_G(v)$ is called the *degree* of v.

A *subgraph* S of a graph G, denoted shortly by $S \subseteq G$, is another graph such that $V(S) \subseteq V(G)$ and $E(S) \subseteq E(G)$ where every edge in $E(S)$ has its both ends in $V(S)$. The subgraph $S \subseteq G$ is called *induced* if, for every

$u, v \in V(S)$, $\{u, v\} \in E(S)$ if and only if $\{u, v\} \in E(G)$. This is denoted by $G[V(S)]$.

For a subgraph $S \subseteq G$, the *attachment* of a vertex $v \in V(G) \setminus V(S)$ in S is the neighborhood of v in S and denoted by $N_S[v]$.

A *path* denoted by P_n is a finite sequence of edges connecting a sequence of n distinct vertices where each pair of consecutive vertices are adjacent. Then, a graph is *connected* if and only if there exists a path between every pair of vertices, and otherwise *disconnected*.

A *(connected) component* of a graph G is a connected subgraph S of G in which no vertex $v \in V(S)$ is adjacent to some $u \in V(G) \setminus V(S)$. Analogous to the attachment of a vertex, the *attachment* of a connected component X in a subgraph $S \subseteq G \setminus X$ is the set of neighborhoods of the vertices $v \in X$ in S.

The *complement* $\overline{G}$ of a graph G is a graph on the same vertex set where a pair of vertices $u, v \in V(G)$ are adjacent in $\overline{G}$ if and only if they are not adjacent in G.

A *subdivision* of an edge $\{u, v\}$ is to replace $\{u, v\}$ with a new vertex w and two new edges $\{u, w\}$ and $\{w, v\}$. A *subdivision* of a graph is the result of a sequence of subdivisions of its edges.

A *complete graph* denoted by K_n is a graph of order n in which any pair of vertices are adjacent. A *clique* in a graph G is a subgraph of G which induces a complete graph, and the order of such a subgraph is called the *size* of that clique. A clique is called *maximal* if it can not be extended to a larger clique by adding new vertices, and a *maximum* clique is a maximal clique of the largest size.

A *cycle* of length n in a graph, denoted by C_n, is defined analogously to a path with the addition that the first and the last vertices are adjacent. Then, a graph G is called *cyclic* if there exists at least one cycle in G. In contrast, a graph which contains no cycle is called *acyclic*.

An edge joining two nonconsecutive vertices of a cycle is called a *chord*. *Chordless cycles* of a graph G are exactly the induced cycles in G, and such cycles of length at least four are also called the *holes*.

A *tree* which we denote by T is a connected acyclic graph. A tree with n vertices has $n-1$ edges, and every pair of vertices of a tree is connected by a unique path. In a tree, the vertices of degree 1 are called the *leaves*, and the

others are called the *branches* or the *branching vertices*. A *star* denoted by S_n is a tree on n vertices which contains one branch also called the *central vertex*.

1.3 Intersection graphs

We consider subclasses of geometric intersection graphs, and mainly focus on two fundamental and well-studied problems in computer science, the class recognition problem and the graph isomorphism problem. We first define the geometric intersection graphs and their subclasses which are related to this thesis. Then, we introduce these two problems together with the complexity overview for mentioned subclasses.

We informally say that a *geometric graph* is a graph which is defined by the structure of certain geometric objects. We now define the intersection graphs.

Definition 1.1. *The* intersection graph *of a family of sets is an undirected simple graph where each set is associated with a vertex of the graph and each pair of vertices is joined by an edge, if and only if the corresponding sets have a non-empty intersection.*

Now, we briefly mention some intersection graphs related to our research.

A graph G is an *interval graph* if the vertex set of G can be mapped into some set of intervals on the real line such that any two vertices of G are adjacent if and only if the corresponding intervals intersect. Interval graphs are thus the intersection graphs of segments on a common line. In Figure 1.2a and Figure 1.2b, we see an example interval graph and its interval representation on the dashed real line, respectively.

(a) (b)

Figure 1.2

A graph is said to be a *chordal* if it has no chordless cycle of length more than three. It is also known that a graph G is chordal if and only if G is the intersection graph of subtrees of some suitable tree [36]. Interval graphs

are a subclass of chordal graphs. In Figure 1.3a and Figure 1.3b, we see a chordal graph and a chordal representation on the dashed tree, respectively.

(a) (b)

Figure 1.3

A graph G is a *circular-arc graph* if G is an intersection graph of a finite set of arcs around a circle. They form a natural superclass of interval graphs. While every interval graph is chordal, circular-arc graphs are not always chordal. For instance, Figure 1.4a and Figure 1.4b, we see a non-chordal circular-arc graph and its intersection representation around the dashed circle, respectively.

(a) (b)

Figure 1.4

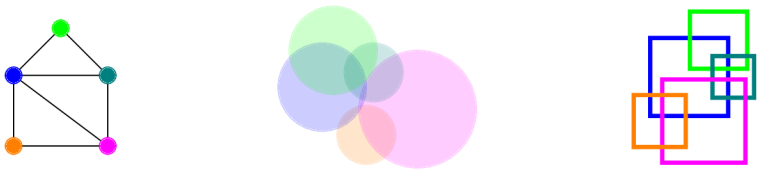
An *independent set* is a set of vertices where no pair of them are adjacent. *Split graphs* are a subclass of chordal graphs of which vertex set can be partitioned into a clique and an independent set. They can be characterized as chordal graphs whose complements are also chordal [19, 33, 37]. As chordal graphs are the intersection graphs of connected subtrees of a tree, split graphs are intersection graphs of subtrees of a subdivision of a suitable star S_d (but the converse is not true). In Figure 1.5a and Figure 1.5b, we see a split graph whose clique vertices are colored orange, each vertex of the independent set is colored differently, and its intersection representation on the dashed star, respectively.

A *disk graph* is the intersection graph of a family of disks in the Euclidean plane. In Figure 1.6a and Figure 1.6b, we see a disk graph and an intersection representation of this graph, respectively. *Square graphs* are defined

(a) (b)

Figure 1.5

analogous to disk graphs on a family of axis-aligned squares instead of disks, and Figure 1.6c is an intersection representation of Figure 1.6a as a square graph.



(a) (b) (c)

Figure 1.6

A *string graph* is an intersection graph of a family of simple curves in the plane where each curve is called a *string*. In Figure 1.7a and Figure 1.7b, we see a string graph and one of its intersection representations, respectively.

(a) (b)

Figure 1.7

A *segment graph* is an intersection graph of a family of straight line segments in the plane. In Figure 1.8a and Figure 1.8b, we see a segment graph and one of its intersection representations, respectively.

A *bipartite graph* $G = (V, E)$ is a graph where V can be partitioned into two sets A and B such that no pair of vertices in A (and in B) are adjacent. Therefore, bipartite graphs contain no odd length cycles (shortly, *odd cycles*). The *grid intersection graphs* are defined as the bipartite segment

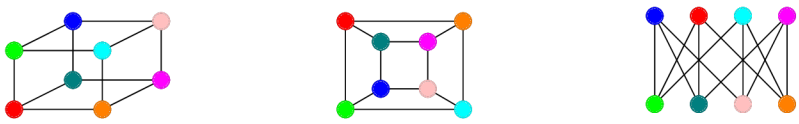
(a) (b)

Figure 1.8

graphs on vertical and horizontal line segments in the plane. Since the graph given in Figure 1.8a contains an odd cycle on blue, green and teal vertices, it does not form a grid intersection graph.

Definition 1.2. *Given a graph G and a particular graph class $\mathcal{C}$, the* recognition problem *of the class $\mathcal{C}$ is to decide whether G belongs to $\mathcal{C}$.*

The complexity of this problem for several geometric intersection graph classes has been shown to be solvable efficiently. For instance, interval [17], chordal [59], circular-arc [54] and split graphs [40] can be recognized in linear time. However, there are some examples such as disk [20], grid intersection [44], segment [45], string [43] and square graphs [55] for which the recognition problem is `NP`-hard. Moreover, for some classes, it is not even clear whether the recognition problem is in `NP` (e.g., for segment graphs [45]).



(a) (b) (c)

Figure 1.9

Definition 1.3. *Two graphs G and H are called* isomorphic*, denoted by $G \simeq H$, if there exists a bijection f from $V(G)$ to $V(H)$ such that $\{u, v\} \in E(G)$ if and only if $\{f(u), f(v)\} \in E(H)$. The* graph isomorphism problem *is to determine whether two graphs are isomorphic.*

While the isomorphism problem has been shown to be solvable in polynomial and even in linear time for many special graph classes, the complexity of this problem in general case is still open. Despite this, it is not assumed to be `NP`-hard. As a result, the complexity class `GI` was defined, and a problem is called *`GI`-complete* if it can be reduced to the graph isomorphism problem in polynomial time.

Among the mentioned intersection graph classes, isomorphism problem can be solved in linear time for interval graphs [17]. The complexity status of the isomorphism problem for circular-arc graphs was unknown for over 40 years, and Krawczyk has recently shown that it can be solved in polynomial time [47]. On the other hand, this problem is `GI`-complete for chordal [63], grid [61] and therefore segment, split [24], string [63], square [57] graphs. This problem is still open for disk graphs.

Many of the mentioned graph classes have a common abstract generalization which is the concept of H-graphs.

Definition 1.4 ([16])**.** *An* H-graph *is the intersection graph of connected subgraphs of a suitable subdivision of a fixed graph* H*.*

H-graphs generalize mentioned intersection graphs as follows. Interval graphs are H-graphs when H is K_2. Circular-arc graphs are H-graphs when H is K_3. Chordal graphs are the union of T-graphs where T ranges over all trees. Split graphs can be represented as H-graphs where H is a star S_d, for suitable d, and this type of H-graph is called an S_d-graph. Moreover, every graph G is an H-graph when $H = G$ by simply representing each vertex v of G by v and all incident edges e to v, see the graph in Figure 1.10a and its H-representation in Figure 1.10b.

(a) (b)

Figure 1.10

Some recognition and isomorphism results specifically on H-graphs can be listed as follows.

- *Recognition.* A *cactus* is a connected graph in which no pair of cycles share an edge. In 2017, Chaplick, Töpfer, Voborník and Zeman showed that the recognition problem for H-graphs is `NP`-complete if fixed H is not a cactus [22]. Even though the recognition of chordal graphs takes linear time [59], in 2015, Klavík, Kratochvíl, Otachi and Saitoh proved that when T is on the input, deciding whether a given chordal graph is a T-graph is `NP`-complete [42] and it can be solved in `XP`-time parameterized by the size of the fixed tree T [22]. On the other

hand, the recognition of S_d-graphs can be solved in polynomial time even with d on the input [22]. While H-graphs for $H = K_3$ can be recognized in linear time [54], the complexity of this problem is unknown for H-graphs when H contains one cycle and several paths (or trees) connected to these cycles.

The recognition problem for H-graphs when H is a cactus and for so-called Helly H-graphs is open (see Section 2.8 for the definition of Helly intersection graphs). In [12], we have shown that the recognition of proper $\mathcal{U}$-graphs is NP-complete when $\mathcal{U}$ is the class of all unicyclic graphs U where U contains exactly one cycle and $U \neq K_3$, and given an FPT-approach to recognize U-graphs parameterized by the size of U. In [8], we first showed that the recognition of H-graphs is NP-complete when H contains two cycles intersecting in precisely one vertex. We then moved on to L-graphs where L contains exactly one cycle and a single path attached to that cycle, and showed that they can be recognized in polynomial time [8]. We finally considered general U-graphs, and showed that, for a fixed unicyclic graph U, U-graph recognition is in XP parameterized by the size of U when the input graphs are strictly cyclic, i.e. non-chordal. We also showed that Helly $\mathcal{U}$-graphs can be recognized in polynomial time where $\mathcal{U}$ is the class of all unicyclic graphs [8].

- *Isomorphism.* The isomorphism problem for H-graphs is GI-complete if H contains a *double-triangle* as a minor [22] where a double triangle is a graph on three vertices u, v, w such that between each pair $\{i, j\} \subsetneq \{u, v, w\}$, there exist two edges. One example is when H is a cactus containing two cycles sharing precisely one vertex. (see Section 2.7 for a simple reduction). For H-graphs when H is a single cycle (i.e., circular-arc graphs), the isomorphism problem can be solved in polynomial time as shown by Krawczyk in 2019, but it is a highly non-trivial task [47]. The isomorphism problem for so-called Helly H-graphs is open. Recalling the isomorphism of split graphs, H must be fixed for the isomorphism of H-graphs to be more efficient than in general. Moreover, the isomorphism of S_d-graphs (thus, the isomorphism of T-graphs as a superclass of S_d-graphs) is GI-complete when d is on the input. We have shown in 2020 and in 2022 that both these problems can be solved in FPT-time, parameterized by d [5], and parameterized by $|T|$ [11], respectively. In 2022, Arvind, Nedela, Ponomarenko, and Zeman also showed that T-graph isomorphism can be solved in FPT-time parameterized by $|T|$ [4] using a different approach

and without mutual influence with [11].

Based on our preprint [12], we have first shown that the isomorphism problem for both chordal and non-chordal proper H-graphs for unicyclic H can be solved in FPT-time. We also give a GI-completeness reduction for proper H-graph isomorphism when H contains two cycles sharing precisely one vertex.

1.4 Outline of the thesis

In this thesis, we mainly consider the recognition and the isomorphism problems for H-graphs when H contains at most one cycle as a generalization of both circular-arc graphs and T-graphs.

In Chapter 2, we outline two algebraic tools for the isomorphism problem, specifically Babai's tower-of-groups procedure and a procedure solving d-bounded color multiplicity graph isomorphism, define partially ordered sets, introduce the well-known combinatorial approach for interval graph isomorphism which utilizes PQ-trees used to recognize them, list some additional parameterized results on graph isomorphism, give our GI-completeness reduction to H-graphs when H-contains at least two cycles intersecting in precisely one vertex [1], and introduce S_d-graphs, T-graphs, L-graphs, U-graphs as well as Helly H-graphs, and proper H-graphs in detail since they are the main subject of this thesis.

Chapter 3 is based on our preprint [6] which is a collection of useful and non-trivial results already used in our papers [5, 10, 11]. We first consider the automorphism problem for colored set families, and give an FPT-algorithm parameterized by the maximum antichain size among those colored sets. We then consider the automorphism problem for colored clique families of an interval graph consistent with the automorphisms of the underlying interval graph, and give an FPT-algorithm parameterized by the maximum antichain size among those colored cliques.

Chapter 4 is based on our paper [5] where we considered the isomorphism problem for S_d-graphs. We first focus on the case where the input S_d-graphs have bounded clique size, and give a simple FPT-algorithm to solve their isomorphism problem parameterized by the maximum clique size. We then give a polynomial time reduction showing that the isomorphism problem for S_d-graphs is at least as hard as the isomorphism problem of posets of bounded width. We finally show that S_d-graph isomorphism can be solved

in FPT-time by combining the results from Chapter 3 with additional tools.

Chapter 5 is based on our paper [10]. We first consider proper S_d- and proper T-graphs which are the restricted cases of general S_d- and T-graphs, and show that their isomorphism can be solved in FPT-time without needing the group theoretical tools, i.e. *fully combinatorially*. We also give a simple XP-procedure to test the isomorphism of general T-graphs using group-theoretical tools.

Chapter 6 is based on our paper [11]. We first give a polynomial time procedure to canonically decompose T-graphs in an isomorphism-invariant way. We then show that T-graph isomorphism can be solved in FPT-time by combining the results from Chapter 3 with our canonical decomposition. We also show that our results extends to an FPT-procedure to test the isomorphism of bounded leafage graphs, and the isomorphism problem for vertex leafage 2 (chordal) graphs is GI-complete which means that we cannot expect an XP-time isomorphism testing for them unless GI can be solved in polynomial time.

Chapter 7 is based on our preprint [12]. We first show that the recognition of proper $\mathcal{U}$-graphs is NP-complete when $\mathcal{U}$ is the class of all unicyclic graphs U where U contains exactly one cycle and $U \neq K_3$. We then show that the recognition and isomorphism problems for both chordal and non-chordal proper H-graphs for unicyclic H can be solved in FPT-time. We also give a GI-completeness reduction for proper H-graph isomorphism when H contains two cycles sharing precisely one vertex.

Chapter 8 is based on our paper [8]. We first show that the recognition of H-graphs is NP-complete when H contains two cycles intersecting in precisely one vertex. We then move on to L-graphs where L contains exactly one cycle and a single path attached to that cycle, and show that L-graphs can be recognized in polynomial time. We finally consider general U-graphs, and show that, for a fixed unicyclic graph U, U-graph recognition is in XP parameterized by the size of U when the input graphs are strictly cyclic, i.e. non-chordal. We also show that Helly $\mathcal{U}$-graphs can be recognized in polynomial time where $\mathcal{U}$ is the class of all unicyclic graphs.

In Chapter 9, we give a fully combinatorial polynomial time reduction from the automorphisms of set families of bounded antichain size to the automorphisms of posets of bounded width which brings us one step closer to the possible computation of the latter in FPT-time without utilizing the group-theoretical tools that are introduced in Chapter 2.

Lastly, we remark that it has recently been brought to our attention, by an anonymous reviewer of EUROCOMB'23, that the first result of Chapter 3 - Theorem 3.3, can be readily derived from the paper of Arvind, Das, Köbler, and Toda [3] via the duality between elements and sets in set families.

1.4.1 Summary of author's contribution

Paper and Co-authors	My contribution
[6] Automorphism group of marked interval graphs in FPT Co-author: Petr Hliněný	50%. I co-wrote the manuscript which contains the results from the following three papers with my specified contributions.
[5] Isomorphism problem for S_d-graphs Co-author: Petr Hliněný	50%. I co-wrote the manuscript, and figured out the isomorphism problem for d-bounded colored multiplicity graphs can be solved in `FPT`-time and that we needed to utilize Venn diagrams.
[10] Efficient isomorphism for S_d-graphs and T-graphs Co-author: Petr Hliněný	50%. I co-wrote the manuscript, and figured out the isomorphism problem for proper S_d- and T-graphs can be solved in `FPT`-time using a fully combinatorial and simpler approach in `FPT`-time by using the notions given in [21] to recognize proper T-graphs in `FPT`-time.
[11] Isomorphism testing for T-graphs in FPT Co-author: Petr Hliněný	50%. I co-wrote the manuscript, and figured out the T-graphs can be uniquely decomposed into a set of levels each having a bounded antichain size by the size of T using the notions given in [53].
[12] Recognition and isomorphism of proper U-graphs in FPT-time Co-author: Peter Zeman	50%. I co-wrote the manuscript, and obtained every result by myself except the `NP`-hardness section with the help of my supervisor Petr Hliněný, and the `GI`-completeness section with my co-author Peter Zeman.
[8] Recognizing H-graphs - Beyond circular-arc graphs Co-authors: Onur Çağırıcı, Jan Derbisz, Tim A. Hartmann, Petr Hliněný, Jan Kratochvíl, Tomasz Krawczyk, and Peter Zeman	12.5%. I co-wrote the manuscript, initiated the collaboration of my co-authors on the topic, and suggested that for the clique C placed on the intersection point of the circle and the stick of L, every component placed on the circle has bounded width when we consider their downsets in the posets formed by their neighborhoods in C.

We also briefly note that the above outcomes have been or are planned to be published in the subsequent conferences. [5] and [8] have been published at the rank-A conference MFCS, [11] has been published at the rank-B conference WALCOM, the full version [10] of [5] has appeared in ALGORITHMICA, [6] and [12] are still arXiv preprints, [6] awaits further development, and [12] has been submitted to WALCOM.

Chapter 2

Algebraic Tools

In this chapter, we introduce the classical group-based algorithm framework for graph isomorphism pioneered by Babai, and Furst-Hopcroft-Luks in the 80's. We first introduce automorphism groups which are very useful in graph isomorphism. Then, we review the famous tower-of-groups of Babai, followed by a brief explanation of the bounded color multiplicity graph isomorphism procedure that we use and adapt in our results.

2.1 Automorphism groups of graphs

We introduce the automorphism group of a graph which is an algebraic view of the graph isomorphism problem.

In general, a *group* is formed by a set X with a binary operation $*$ on X, satisfying basic axioms. We shall deal exclusively with *permutation groups*, where X is a set of permutations over some ground set A, and $*$ is the composition of permutations. Obviously, a permutation group is closed under the composition of its permutations.

Considering a permutation group Π, the *order* of Π is the number of all permutations in Π, and denoted by $|\Pi|$. The *symmetric group* $Sym(A)$ is the set of all permutations on a ground set A, and its order is $|Sym(A)| = |A|!$.

Given a graph G, an *automorphism* is an isomorphism of G to itself. The *automorphism group* of G is the group of all automorphisms of G and it is denoted by $Aut(G)$. Since an isomorphism is a permutation of vertices, an automorphism group is actually a group of permutations.

An automorphism group $Aut(G)$ is usually presented by a generator set R (each element of R is an automorphism of G) where each member of $Aut(G)$ is the result of some composition of finitely many elements in R. Generator sets can be much smaller than the whole group, e.g. the generator set for the symmetric group contains only two permutations.

There is a tight relation between automorphism and the isomorphism problem, which is easily explained as follows: Considering two graphs G and H, one can construct their disjoint union $G \uplus H$, and check whether there exists an automorphism of $G \uplus H$ which swaps G and H meaning that G and H are isomorphic, and if such an automorphism of $G \uplus H$ exists, it is seen in the generators of the automorphism group. In fact, if G and H are connected, it is enough to have one automorphism of $G \uplus H$ sending a vertex of G to H. On the other hand, it was shown that one can compute the automorphism group of $G \uplus H$ by applying the isomorphism test between G and H $\mathcal{O}(n^4)$ times [52].

2.2 Babai's tower-of-groups

The famous classical paper of Babai [14] is not just a standalone algorithm, but more an outline of how to efficiently compute a subroup Γ of a given group Γ' in a setting in which the conditions defining the subgroup Γ can be stepwise refined in sufficiently small steps. Here, by "computing a group" we mean to output a set of its generators which is at most polynomially large by [34]. Note that this task, in general, cannot be done directly by processing all members (even though we have an efficient membership test at hand) of the group Γ' which can be exponentially large.

Babai's "tower-of-groups" approach informally works as follows: we iteratively compute a chain of subgroups $\Gamma' = \Gamma_0 \supseteq \Gamma_1 \supseteq \ldots \supseteq \Gamma_h = \Gamma$, where each Γ_{i+1} consists of those members of Γ_i which satisfy a suitably chosen additional condition, until $\Gamma_h = \Gamma$ satisfies all the defining conditions, and so it is the desired outcome. The important ingredient which makes this procedure work efficiently is that the ratio of orders (sizes) of consequent groups Γ_i and Γ_{i+1} in the chain is always bounded. The number h of steps should also not be too large. In this setting, each of the refinement steps can be done using another classical result:

Theorem 2.1 (Furst, Hopcroft and Luks [34, Corollary 1])**.** *Let Π be a permutation group given by its generators, and Π_1 be any subgroup of Π such that one can test in polynomial time whether $\pi \in \Pi_1$ for any $\pi \in$*

Π *(membership test). If the ratio* $|\Pi|/|\Pi_1|$ *is bounded by a function of a parameter* d*, then a set of generators of* Π_1 *can be computed in* `FPT`*-time (with respect to* d*).*

This result should be understood in the following way: When the ratio of orders of the groups Π and Π_1 is manageable (meaning "small"), then one can compute the generators of the subgroup efficiently, e.g. in `FPT`-time. The theorem has been used for testing the isomorphism of bounded valence graphs where the ratio of orders was originally bounded by a polynomial in the input size n. However, we present it in the parameterized setting to match our research case.

2.3 Bounded color multiplicity graph isomorphism

We now describe a method to solve the bounded color multiplicity graph isomorphism in `FPT`-time by applying Theorem 2.1 [34] together with Babai's tower-of-groups approach [14]. We first define the bounded color multiplicity graphs which we apply in our `FPT` isomorphism algorithm for colored set families in Chapter 3, S_d-graphs in Chapter 4 and T-graphs in Chapter 6.

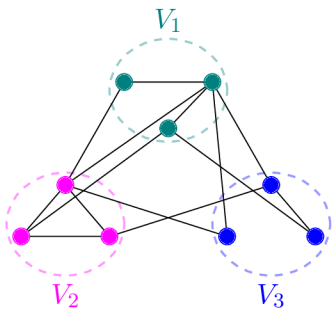


Figure 2.1

A *d-bounded color multiplicity graph* is a graph G whose vertex set is arbitrarily partitioned into k color classes $V(G) = V_1 \cup \ldots \cup V_k$ where $V(G) = \bigcup_{i=1}^{k} V_i$ and for each i, j, $V_i \cap V_j = \emptyset$. The number k of colors is arbitrary, but for all $1 \leq i \leq k$, we have $|V_i| \leq d$ (this d is called the multiplicity). Figure 2.1 shows a 3-bounded color multiplicity graph on three color classes V_1, V_2 and V_3. We have the following classical result:

Theorem 2.2 (Babai [14], with Furst, Hopcroft and Luks [35])**.** *Let* G *be a* d*-bounded color multiplicity graph. The color-preserving automorphism*

group of G (i.e., its generators) can be determined by an FPT*-time algorithm with the parameter d.*

We now outline the proof of the above result. Let $Sym^{col}(G)$ be the symmetric group which respects the colors of the vertices in G, i.e. $Sym^{col}(G)$ is the product of the symmetric groups of the color classes. Starting with $Sym^{col}(G)$, the goal is to iteratively compute the tower-of-groups $Sym^{col}(G) = \Pi_0 \supseteq \Pi_1 \supseteq \ldots \supseteq \Pi_h = Aut(G)$ where $Aut(G)$ consists of the color-preserving automorphisms of G.

For a graph G on n vertices, and its color classes $V(G) = V_1 \cup \ldots \cup V_k$ with $|V_i| \leq d$, the following is a short overview of this approach to compute the color-preserving automorphisms of G:

1. $Aut(G)$ can be emdedded on the direct product $Sym^{col}(G) = Sym(V_1) \times \cdots \times Sym(V_k)$. Let Π_0 denote this group of all permutations which preserve the colors of vertices.

2. Let the pairs of color indices be enumerated as (a_i, b_i) for $i = 1, 2, \ldots, \binom{k}{2}$, where $a_i < b_i$. For each pair $(a, b) := (a_i, b_i)$, let E_i be the set of edges of the induced subgraph $G[V_a \cup V_b]$, and let Π_i be the subgroup of Π_{i-1} consisting of all permutations $\pi \in \Pi_{i-1}$ such that π induces an automorphism of the graph with the vertex set $V(G)$ and the edge set $E_1 \cup E_2 \cup \ldots \cup E_i$. Since $|\Pi_{i-1}|/|\Pi_i| \leq |V_a|! \cdot |V_b|! \leq (d!)^2$, we can compute Π_i from Π_{i-1} using Theorem 2.1.

3. Finally, since the number of color pairs $\binom{k}{2} = \mathcal{O}(n^2)$, the color-preserving automorphism group $\Pi_{\binom{k}{2}} = Aut(G)$ of G can be computed in FPT.

2.4 Partially ordered sets (posets)

In this section, we review the folklore definitions of posets needed in this thesis.

A *partial order* is a binary relation $\preceq$ over a set P which is reflexive, antisymmetric and transitive. This is usually denoted by $(P, \preceq)$. A *partially ordered set* or shortly a *poset* is a set with partial order indicating that some elements of this set precede some other elements. $a \preceq_P b$ means that the element a of partially ordered set P is related to another element b of P, and related elements are said to be *comparable*. Otherwise, they are called *incomparable*. A *chain* is a set of elements in a partially ordered set P in which every two elements are comparable. A *chain cover* of a partially or-

dered set P is a collection of chains whose union is P. An *antichain* is a set of elements in a partially ordered set P in which every two elements are incomparable.

The *greatest (or maximum) element* of a poset is the element greater than all other elements of that poset. The *least (or minimum) element* of a poset is oppose to it, smaller than all other elements. A *maximal element* of a poset is an element such that there exists no element greater than this element. On the other hand, a *minimal element* of a poset is an element such that there exists no element smaller than this element. In posets, the greatest and least element are unique and not every poset has to have these elements. Moreover, when a poset has the greatest (or least) element, then this poset has only one maximal (or minimal) element which is identical to the former.

The *depth* of an element e in the poset is the maximum number of elements from that element to the smallest possible element of all chains which e belongs to, starting from 1. The *level* i of a poset contains all the elements which are at depth i. The *width* of a poset is the size of the greatest antichain.

The *Hasse diagram* D of a poset P is a directed graph whose vertex set is the element set of P and for each pair of vertices $u, v \in D$, there exists an edge directed from u to v if and only if $u \preceq_P v$ but $u \preceq_P w \preceq_P v$ holds for no $w \in P$.

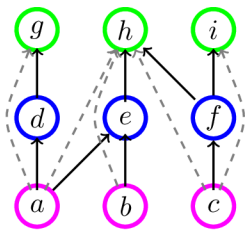


Figure 2.2

In Figure 2.2, we see an example poset P of which Hasse diagram can be obtained by discarding the gray dashed edges. The comparable pairs of elements are shown by directed edges (but reflexive loops are ignored for simplicity). Specifically, there is a directed edge from u to v if $u \preceq_P v$. The colors correspond to the levels of P where level 1 colored with magenta contains the minimum elements, and level 3 colored with green contains the maximum elements of P. Each level forms an antichain of P, and some

example antichains not corresponding to the levels of P are the triples (b, d, i) and (c, d, h).

We say that two posets P and Q are *isomorphic* if there exists a bijection f from P to Q such that for each pair of elements $u, v \in P$, $u \preceq_P v$ if and only if $f(u) \preceq_Q f(v)$. Poset isomorphism is `GI`-complete as can be seen from [62]. *Poset automorphism*, defined analogously to graph isomorphism, is an isomorphism from a poset P to itself. Figure 9.1 illustrates a poset P with its comparability relation shown by directed black lines, and the directed colored dashed lines demonstrate its automorphisms.

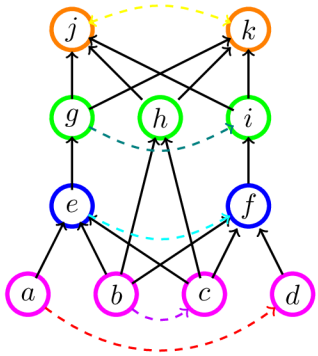


Figure 2.3

In the parameterized setting, the computation of poset automorphisms, thus the isomorphism of posets, can be solved using the classical group-based approach pioneered in Babai [14] and given in the previous section in a bounded color multiplicity setting. Consider a poset P of width $\leq d$ and the levels $L_1, \ldots, L_k$ of P, where $|L_i| \leq d$ for $1 \leq i \leq k$. Note that any automorphism of P preserves the levels. Having the levels L_i as color classes, we may forget about the edge directions between comparable elements of P as it is implicit from lower to higher levels, and view P as a d-bounded color multiplicity graph in which there are no edges between vertices from the same color class. Hence we immediately get from Theorem 2.2:

Corollary 2.3 ([14, 35])**.** *The automorphism group of a poset P of width d can be determined by an* `FPT`*-time algorithm with the parameter d.*

2.5 PQ-trees of interval graphs

In this section, we describe a well-known linear-time method to recognize and test the isomorphism problem for interval graphs which we apply in our FPT isomorphism algorithm for marked clique families of interval graphs in Chapter 3, S_d-graphs in Chapter 4, T-graphs in Chapter 6, and proper S_d- and T-graphs in Chapter 5, and FPT isomorphism and recognition algorithms for proper U-graphs in Chapter 7.

To recognize and test the isomorphism of interval graphs, Booth and Lueker [17] invented *PQ-trees* (see Figure 3.6), ordered rooted trees which have the maximal cliques of an interval graph in its leaves, and every internal node is either one of the following:

- A *P-node*: the order of its children can be permuted arbitrarily.
- A *Q-node* : the order of its children can be reversed (but not changed otherwise).

The above permissible reorderings at P- and Q-nodes are called *equivalence transformations* of a PQ-tree, and the following is well-known:

Theorem 2.4 (Booth and Lueker [17])**.** *For every interval graph G, one can in linear time construct a PQ-tree T, such that the following hold:*

- *This PQ-tree T of G is unique up to equivalence transformations.*
- *Every possible interval representation of G corresponds to a PQ-tree T' of G (i.e., one equivalent to T). The correspondence is that the linear order of the maximal cliques in the representation of G is the same as the one given by the linear order of the leaves of T'.*

In particular, the latter point means that every automorphism of G can be represented as an equivalence transformation of a PQ-tree of G (though, not the other way round without further information associated with the tree).

Figure 2.4a shows an intersection representation of an interval graph of which maximal cliques are labelled from A to L respecting their linear ordering in this representation. Figure 2.4b shows its unique (up to equivalence transformations) PQ-tree. P-nodes are triangle-shaped and Q-nodes are rectangle-shaped, and the leaves of this PQ-tree are in a one-to-one correspondence of its maximal cliques. In Figure 2.4b, colored dashed lines shows the automorphisms of those maximal cliques with respect to the equivalence transformations of this PQ-tree.

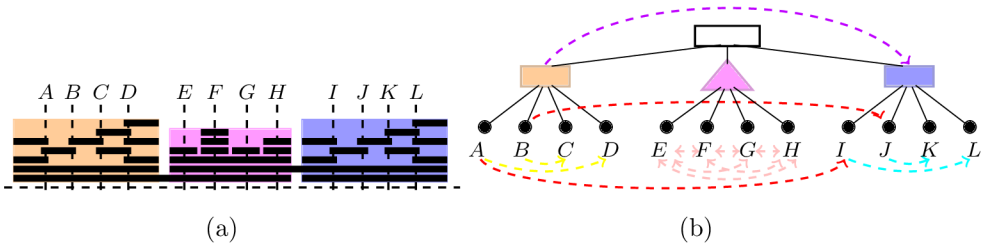


Figure 2.4

We say that an assignment of PQ-trees to interval graphs is *canonical* if, whenever we take isomorphic graphs $G \simeq G'$ and their canonical PQ-trees T and T', then T and T' are isomorphic respecting the order of the trees (one may say "the same"). The following fact is crucial for us:

Corollary 2.5 (Colbourn and Booth [25], noted already in [17])**.** *For every interval graph G, one can in linear time compute the automorphism group of G and a PQ-tree of G which is canonical.*

The definition of a PQ-tree (of an interval graph) explicitly refers only to the maximal cliques of G and not directly to its vertices, but it will be useful to clearly understand the relation of PQ-tree nodes to the particular vertices of G. Every node p of a PQ-tree T of G can be associated with a subgraph of G formed by the union of all cliques of the descendant leaves of p – this subgraph is said to *belong* to p. Then, for a node p, we define the *inner vertices assigned to* p as those vertices of G which belong to p and, if p is not a leaf, they belong to at least two child nodes of p, but they do not belong to any node which is not an ancestor or a descendant of p.

Note that every vertex of G is an inner vertex of precisely one node of its PQ-tree T. Moreover, by the 'consecutive-ones' property of a PQ-tree, the following holds; if a vertex v of G is an inner vertex of a node p of T, and v also belongs to a son p_1 of p, then v belongs to every descendant of p_1 in T. This illustrates the unique nature of the node p (to which v is an inner vertex) for the vertex v within the PQ-tree T.

In the case of a P-node, the inner vertices assigned to p belong to all child nodes of p, but this is generally not true for Q-nodes. We thus additionally define the *ranking of inner vertices*; this is trivial for P-nodes (all inner vertices of the same rank). For a Q-node q of T, we index the sons of q from left o right in a palindromic way (i.e., as $1, 2, 3, 2, 1$ or $1, 2, 3, 3, 2, 1$

depending on parity), which is invariant upon reversal. The *rank* of every inner vertex w assigned to q is then the multiset of indices of the sons of q that w belongs to (obviously, this must be a consecutive section of the index sequence). Observe that since two inner vertices of the same node and of the same rank are in the same collection of maximal cliques of G, they are mutually symmetric in the automorphism group of G.

2.6 Some parameterized results on graph isomorphism

The graph isomorphism problem has not been studied much in parameterized setting, and the parameterized complexity of this problem remains to be open for various graph parameters. On the other hand, there are some parameterizations of graph isomorphism yielding to `FPT`- and `XP`-algorithms listed below.

The isomorphism problem can be solved in `FPT` parameterized by the size of the minimum feedback vertex set[46], by the eigenvalue multiplicity [31], by tree-depth [18] as well as by tree-width [50].

There are some other parameters that result in `XP`-time isomorphism algorithms. The well-known examples are the bounded degree [15], the bounded genus [56] and the size of the smallest excluded minor [58].

2.7 `GI`-completeness reduction to H-graphs when H contains at least two cycles sharing a vertex

In this section, we show that the isomorphism problem for H-graphs, where H has at least two cycles intersecting in precisely one vertex, is `GI`-complete. This result was already included in the proposal of this thesis [1]. To obtain that result, we present a simple reduction which "models" an arbitrary graph in an H-graph in such a way that isomorphism relation is preserved.

Informally, Figure 2.5 illustrates that `GI`-completeness of the isomorphism problem for H-graphs when H has two cycles sharing a vertex. In Figure 2.5a, we have an intersection representation of an H-graph where H consists of two cycles sharing a vertex, and it has three partitions V_b, V_m and V_t indicated by blue, magenta and teal, respectively. Each of the induced subgraphs on V_b, V_m and V_t forms a clique, the induced subgraph on $V_b \cup V_t$ has no edges between blue and teal vertices, V_m consists of arbitrary

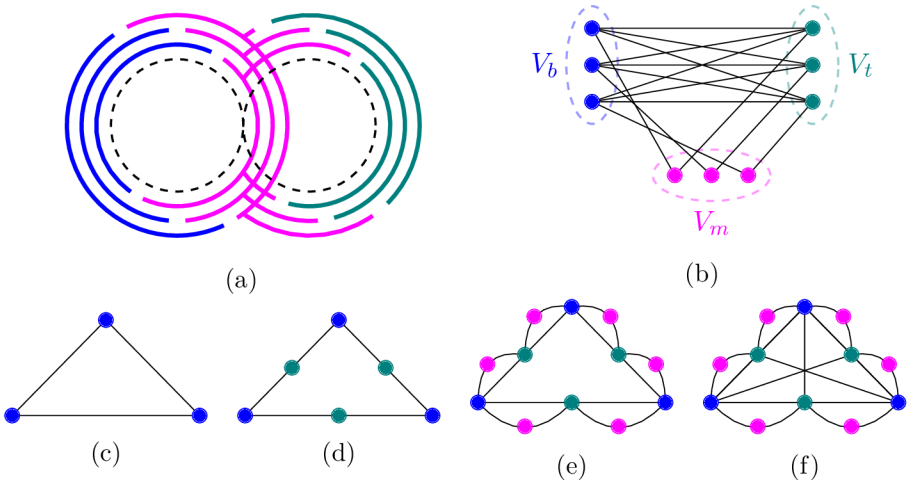


Figure 2.5

connectors between V_b and V_t, and each vertex of V_m is adjacent to all vertices in V_b and V_t except one $v_i \in V_b$ and one $v_j \in V_t$. In Figure 2.5b, we see the complement of the graph given in Figure 2.5a where the induced graph $V_b \cup V_t$ is a complete bipartite graph on blue and teal partitions, each magenta vertex is adjacent to only the corresponding v_i and v_j, and there is no pair of adjecent vertices of the same color.

We construct the complement G^* of an H-graph where H is a graph on two cycles sharing precisely one vertex from a general graph G. In Figure 2.5c, we have a general graph G on blue vertices ${V_b}^*$, in Figure 2.5d, we see the graph G' obtained from G by subdividing its edges with teal vertices ${V_t}^*$, and in Figure 2.5e, we see the graph G'' obtained by subdividing G' into triangles with magenta vertices ${V_m}^*$. We finally add the missing edges between pairs (v_i, v_j) such that $v_i \in {V_b}^*$ and $v_j \in {V_t}^*$ obtaining the graph G^* given in Figure 2.5f. Note that G^* models a general graph isomorphic to the complement graph given in Figure 2.5b when the size of the blue, magenta and teal partitions are the same.

Theorem 2.6. *The isomorphism problem for H-graphs is* `GI`*-complete when H contains at least two cycles intersecting in precisely one vertex.*

Proof. Let G_1 and G_2 be two arbitrary graphs on the same number n of vertices and on the same number m of edges. We construct ${G_i}^*$, $i = 1, 2$, as described above. Assume that G_1 and G_2 are isomorphic. Then, we easily

get that ${G_1}^* \simeq {G_2}^*$ since the above construction is deterministic. Now assume that ${G_1}^*$ and ${G_2}^*$ are isomorphic under a bijection $f : V({G_1}^*) \to V({G_2}^*)$. Then, a restriction of f mapping $V(G_1)$ to $V(G_2)$ is an isomorphism due to vertex degrees. Therefore, $G_1 \simeq G_2$ if and only if ${G_1}^* \simeq {G_2}^*$. Since each ${G_i}^*$ for $i = 1, 2$ is the complement of an H-graph where H consists of two cycles sharing a vertex, solving their isomorphism would solve also the isomorphism of G_1 and G_2. Therefore, the isomorphism problem for H-graphs is already GI-complete. □

2.8 S_d-graphs, T-graphs, L-graphs, U-graphs, Helly H-graphs and Proper H-graphs

We now focus on the graph classes which are a subject to this thesis, and introduce S_d-, T-, L-, U-, Helly H- and proper H-graphs in detail.

S_d-graphs

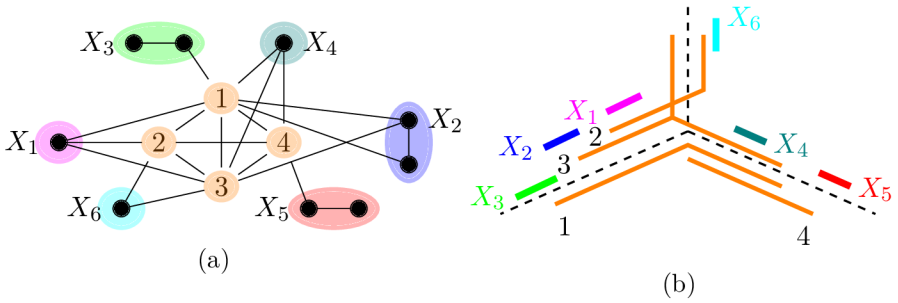


Figure 2.6

An S_d-*graph* G is the intersection graph of connected subgraphs of a suitable subdivision S' of a fixed star S_d. Note that the center of S' implicitly defines a maximal clique of G, called the *central clique* of the representation, and every ray of S' defines an interval subgraph of G. They are a subclass of chordal graphs.

Given a graph G and some maximal clique C of G, let $\mathcal{X}$ be the set of connected components of $G - C$, and for each connected component $X_i \in \mathcal{X}$, let $N_C(X_i)$ be the set of neighbors of X_i in C which is called as the *attachment* of X_i. In [22], Chaplick, Töpfer, Voborník and Zeman showed that G is an S_d-graph *if and only if* there exists a maximal clique C of G such that

- the induced subgraph $G[C \cup X_i]$ is an interval graph for all $X_i \in \mathcal{X}$,
- the partial order P on the connected components with respect to their attachments can be covered by at most d chains, and
- for each chain of P covering $X_1, \ldots, X_k$, the induced subgraph $G[C \cup X_1, \ldots, X_k]$ is an interval graph.

Since $C \cup X_i$ induces an interval subgraph of G, the neighborhoods of the vertices of X_i in C form a chain by inclusion. Then, the upper attachment of X_i, denoted by ${N_C}^U(X_i)$, is the maximum neighborhood in C among the vertices of X_i, and $N_C(X_i) = {N_C}^U(X_i)$. Analogously, the lower attachment of X_i, denoted by ${N_C}^L(X_i)$, is the minimum neighborhood in C among the vertices X_i. The minimal edge-cutset between C and X_i is called *attachment edges*, and the component X_i together with its attachment edges is called a *bridge X_i of C*.

After determining the attachments of all connected components of $G-C$, the poset P is constructed as follows. A pair (X_i, X_j) of components of $G-C$ is comparable in P, denoted by $X_i \preceq_P X_j$, if and only if ${N_C}^U(X_i) \subseteq {N_C}^L(X_j)$ holds. Otherwise, they are incomparable. When different connected components of $G-C$ have the same attachment, precisely ${N_C}^U(X_i) = {N_C}^L(X_i) = {N_C}^U(X_j) = {N_C}^L(X_j)$, we will always implicitly treat their union as one component / bridge of C in order to maintain antisymmetry of $\preceq_P$.

Observe that $X_i \preceq_P X_j$ if and only if there is an interval representation of the subgraph of G induced by $C \cup X_j \cup X_i$ in which C is "to the left" of X_j and X_i is "to the right" of X_j [22].

In Figure 2.6a, we see an S_d-graph G whose maximal clique C is colored orange and the connected components of $G - C$ are colored differently. In Figure 2.6b, we see one of its S_d-representations for $d = 3$ where the attachments of three tuples of components (X_1, X_2, X_3), (X_4, X_5) and (X_6) form a chain cover of P. Note also that the attachments of three tuples of components (X_1, X_6), (X_2, X_3) and (X_4, X_5) is another chain cover of P giving another S_d-representation. Therefore, for an S_d-graph G, there can be more than one S_d-representation.

While the recognition problem can be solved in polynomial time for S_d-graphs [22], S_d-graph isomorphism is `GI`-complete with d is on the input [63].

In Chapter 4, we show that S_d-graph isomorphism can be solved in `FPT`-time parameterized by d which is based on our paper [5].

T-graphs

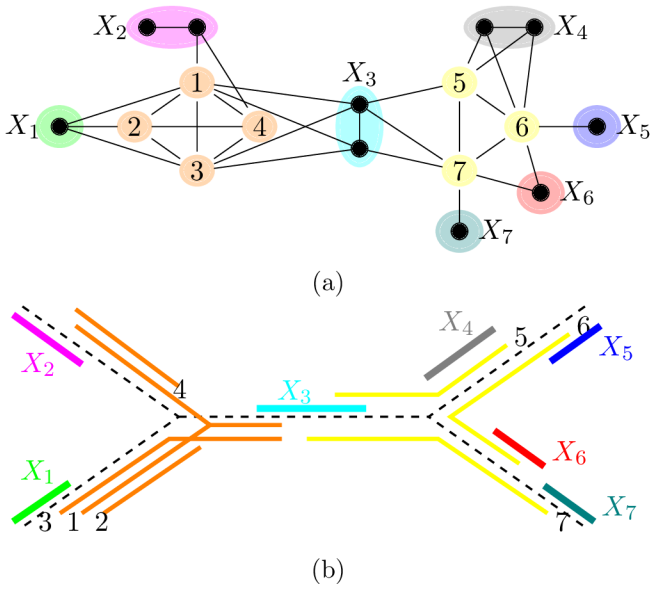


Figure 2.7

A *T-graph* G is the intersection graph of connected subtrees of a suitable subdivision T' of a fixed tree T. Note that each branching vertex of T' defines a maximal clique of G, called a *branching clique*, and the path between any pair of branching vertices of T' defines an interval subgraph of G. When T has one branching vertex, then $T = S_d$ and therefore S_d-graphs are a subclass of T-graphs. Moreover, T-graphs are a subclass of chordal graphs.

Given a graph G, a fixed tree T with branching vertices $b_1, \ldots, b_k$, and the placement of maximal cliques $C_1, \ldots, C_k$ of G to the branching vertices of T, let $\mathcal{X}$ denote the set of connected components of $G - (C_1 \cup \cdots \cup C_k)$. In [22], Chaplick, Töpfer, Voborník and Zeman also showed that G is a T-graph *if and only if* there exists an assignment f of maximal cliques to the branching vertices such that for each branching vertex b_i of T and the maximal clique C_i placed on b_i, the induced subgraph graph on C_i and the connected components placed on the incident vertices to b_i is an S_d-graph where $d = deg(b_i)$.

They also showed that such an assignment f determines the components placed between consecutive branching nodes of T. Specifically, for a component $X \in \mathcal{X}$ and two maximal cliques C_i and C_j placed on consecutive

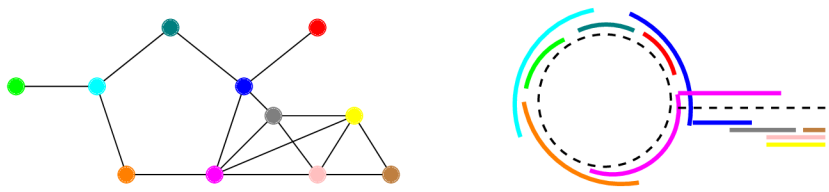
branching nodes of T, if both $(C_i \setminus C_j) \cap N_{C_i}(X) \neq \emptyset$ $(C_j \setminus C_i) \cap N_{C_j}(X) \neq \emptyset$ hold, then X has to be placed on the edge between nodes of T where C_i and C_j are placed [22]. This result clearly extends to any H-graph.

In Figure 2.7 (a), we see a T-graph G whose maximal cliques C_1 and C_2 placed on the branching vertices of T are colored orange and yellow, respectively, and the connected components of $G-(C_1 \cup C_2)$ are colored differently. In Figure 2.7 (b), we see one of its representations on T.

When T is on the input, then the recognition problem for T-graphs is `NP`-complete [42]. On the other hand, for a fixed T, deciding whether a given graph G is a T-graph is in `XP` [22] and their isomorphism is `GI`-complete as an implication of the `GI`-completeness of S_d-graphs [63]. In Chapter 5, we give a simple `XP`-time isomorphism testing based on our paper [10], and in Chapter 6, we give a non-trivial `FPT`-time isomorphism testing for T-graphs based on our paper [11], both parameterized by the size of T.

L-graphs

As we mentioned above, the recognition and isomorphism problems for H-graphs when H contains exactly one cycle are open except the case when $H = K_3$, and the isomorphism problem for H-graphs when H has at least two cycles is `GI`-complete. Since the recognition and isomorphism questions of circular-arc graphs have already been solved, it is natural to consider the second most trivial case (other that $H = T$ for some tree T) where H consists of a cycle and a single path is attached to that cycle. We call this kind of H-graphs a *lollipop* and denote them as *L-graphs*. Note that S_d-graphs for $d \leq 3$ and circular-arc graphs are a subclass of L-graphs. In Figure 2.8a, we see an L-graph, and in Figure 2.8b, one of its L-representations.



(a) (b)

Figure 2.8

In Chapter 8 which is based on our paper [8], we show that L-graphs can be recognized in polynomial time.

U-graphs

After tackling L-graphs, we naturally consider H-graphs which consist of exactly one cycle and several paths or trees attached to it (i.e., *unicyclic*), and we denote them as U-graphs. Figure 2.9 shows an intersection representation of a U-graph with one path, one star and one tree attached to the unique circle of U.

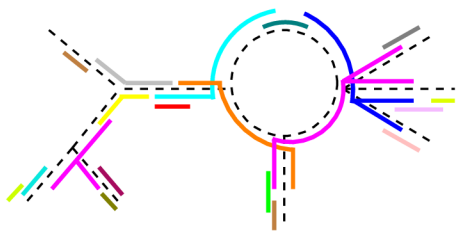


Figure 2.9

The recognition problem for U-graphs is NP-complete [42] since they are a superclass of T-graphs, and their isomorphism is also GI-complete as an implication of the GI-completeness of S_d-graphs [63]. In Chapter 8 which is based on our paper [8], we show that U-graphs which are non-chordal can be recognized in XP-time parameterized by the size of U.

Helly H-graphs and proper H-graphs

A family F of sets is said to satisfy the *Helly-property* if and only if every subfamily of F consisting of pairwise intersecting subsets has a common intersection. It is known that circular-arc graphs may not satisfy the Helly-property, and thus, they may have exponentially many maximal cliques which do not correspond to the points of the circle [49]. For instance, the intersection representation in Figure 2.10b of the circular-arc graph in Figure 2.10a does not satisfy the Helly-property since the maximal clique with blue, cyan and green vertices can not be realized as a point of the circle.

In Figure 2.11, we have a circular-arc graph constructed as the complement of an induced matching, and this kind of graphs has exponentially many maximal cliques.

Considering a particular H-representation of an H-graph G, a clique C of G is called *Helly* if it satisfies the Helly property, i.e. in that H-representation of G, all vertices of C mutually intersect at some point of H. Then, a *Helly H-graph* is an H-graph which has some H-representation where all its cliques

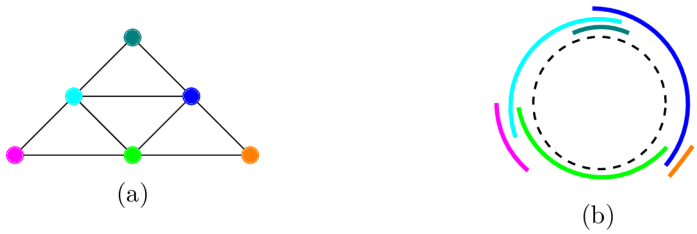


Figure 2.10

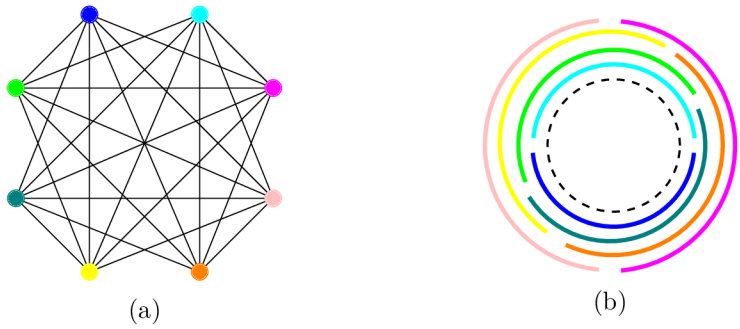


Figure 2.11

satisfy the Helly property. With respect to a specific H-representation, we refer the cliques not satisfying the Helly property as *non-Helly* cliques. In addition, the size of a non-Helly clique is at least 3 by the definition of the Helly property.

In Chapter 7, we show that, for a fixed unicyclic graph U, Helly U-graph recognition and isomorphism can be solved in FPT-time parameterized by the size of U which is based on our preprint [12]. In Chapter 8 which is based on our paper [8], we consider the class $\mathcal{U}$ of all unicyclic graphs U, and show that Helly $\mathcal{U}$-graphs can be recognized in polynomial time.

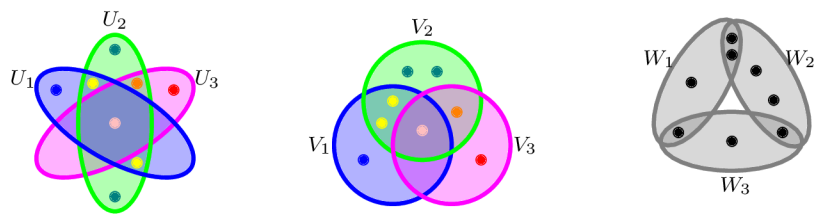
Proper H-graphs are defined as H-graphs which have at least one proper H-representation. A *proper* H-representation is an intersection representation in which we require that no connected subgraph of H is properly contained in another one. It is known that proper interval graphs can be recognized and tested for isomorphism in linear time [41, 26]. Similarly, both the recognition and isomorphism problems are linear time solvable on proper circular-arc graphs [26, 27]. Therefore, we assume that $U \neq K_3$ since K_3-graphs are circular-arc graphs.

Considering proper S_d-graphs, their isomorphism is GI-complete with d is on the input [63] which applies to all proper U-graphs for $U \neq K_3$. Considering proper T-graphs, their recognition is solvable in FPT-time [21]. In Chapter 5, we show that proper S_d-graph isomorphism and proper T-graph isomorphism can be solved in FPT-time by fully combinatorial approaches based on our paper [10], parameterized by the d and the size of T, respectively. Therefore, we also assume that $U \neq T$ for any tree T. In Chapter 7, we show that both the recognition and isomorphism problems can be solved in FPT-time on proper U-graphs.

Chapter 3

Automorphisms of Colored Set Families and Colored Cliques of Interval Graphs

In this chapter, we solve two problems related to computing the automorphism group. By 'computing a group', we mean to output a set of its generators which is at most polynomially large by [34] while the permutation group itself is often exponentially large compared to the ground set.



(a) $\mathcal{U}$ (b) $\mathcal{V}$ (c) $\mathcal{W}$

Figure 3.1

We first define the *isomorphism of set families* which is GI-complete and differs from the graph isomorphism problem where the vertices are mapped to each other (corresponding to the ground set elements here). Given two set families $\mathcal{U}$ and $\mathcal{V}$, an isomorphism from $\mathcal{U}$ to $\mathcal{V}$ maps the sets in $\mathcal{U}$ to the sets in $\mathcal{V}$ and preserves all relations among those sets. The latter condition assures the existence of a bijection from the ground sets in $\mathcal{U}$ to those in $\mathcal{V}$.

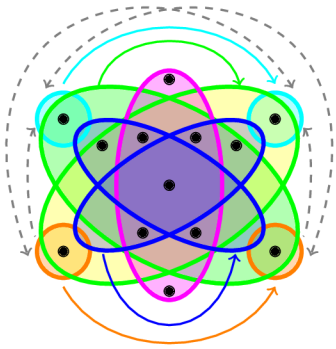


Figure 3.2

Figure 3.1a, Figure 3.1b and Figure 3.1c shows three set families $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{W}$ where the sets and the ground set elements are demonstrated by ellipses and dots, respectively, and $\mathcal{U} \simeq \mathcal{V}$ and the bijection between their sets (as well as the witnessing bijection between their ground set elements) is shown by the same coloring while $\mathcal{W}$ is not isomorphic to these set families.

We first solve the problem of computing the automorphism group of a colored set family $\mathcal{U}$ which is known to be `GI`-complete even when every set $B \in \mathcal{U}$ has cardinality equal to 2, and in the general setting, it coincides with the automorphism problem for hypergraphs which is also `GI`-complete.

Definition 3.1. The problem $\textsc{AutomSET}(X; \mathcal{U}^{[m]})$ is defined as follows:

Input A finite ground set X, and for $m \geq 1$ color classes, a partition $\mathcal{U}^{[m]} = (\mathcal{U}^1, \ldots, \mathcal{U}^m)$ of a finite set family $\mathcal{U} \subseteq 2^X$ where $\mathcal{U} = \mathcal{U}^1 \cup \ldots \cup \mathcal{U}^m$.

Task Compute the group Γ of the color-preserving permutations of $\mathcal{U}^{[m]}$ which come from a permutation of the ground set X.

Precisely, a permutation τ of $\mathcal{U}^{[m]}$ belongs to Γ, if and only if there exists a permutation σ of X such that, for every $i \in \{1, \ldots, m\}$ and all $B \in \mathcal{U}^i$, we have $\tau(B) \in \mathcal{U}^i$ and $\tau(B) = \sigma(B)$ (this trivially gives $|B| = |\tau(B)|$). Note that the latter condition also immediately implies that the (possible) multiplicities of B and of $\sigma(B)$ in each family $\mathcal{U}^i$ are equal.

In Figure 3.2, the black dots demonstrate the ground set elements in X, the ellipses demonstrate the set family $\mathcal{U}$, and the colors of the sets demonstrate the partition $\mathcal{U}^{[m]}$ of $\mathcal{U}$ into blue, cyan, green, magenta and orange

color classes where $m = 5$. The colored lines correspond to the non-trivial automorphisms of $\mathcal{U}^{[5]}$ which are color-preserving. The dashed black lines together with colored lines correspond to the non-trivial automorphisms of $\mathcal{U}^{[5]}$ which are not color-preserving.

Since $\textsc{AutomSET}(X; \mathcal{U}^{[m]})$ is GI-complete, we consider it in the parameterized setting and show that it can be solved in FPT-time parameterized by the maximum antichain size of $\mathcal{U}^{[m]}$.

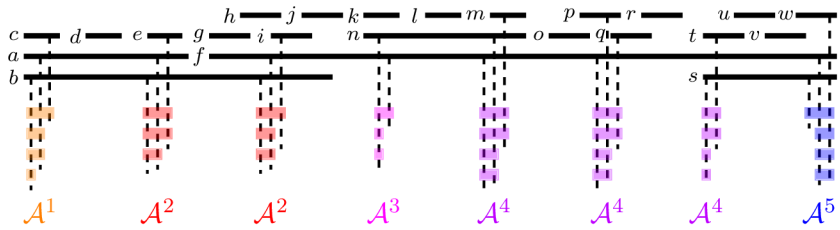


Figure 3.3

We then solve the problem of computing the automorphism group of a colored clique family $\mathcal{A}$ of an interval graph G which results from the automorphisms of the underlying interval graph G.

Definition 3.2. The problem $\textsc{AutomMarkedINT}(G; \mathcal{A}^{[m]})$ is defined as follows:

Input An interval graph G, and for $m \geq 1$ color classes, a partition $\mathcal{A}^{[m]} = (\mathcal{A}^1, \ldots, \mathcal{A}^m)$ of a finite set family $\mathcal{A} \subseteq 2^{V(G)}$ where $\mathcal{A} = \mathcal{A}^1 \cup \cdots \cup \mathcal{A}^m$, and for all $i \in \{1, \ldots, m\}$, every set $A \in \mathcal{A}^i$ is nonempty and induces a clique of G. The sets $A \in \mathcal{A}^{[m]}$ are called the *marked sets* of G, and specially, $A \in \mathcal{A}^i$ is called a *marked set of color* i.

Task Compute the group Γ of such permutations of $\mathcal{A}^{[m]}$ that are the *actions of the* $\mathcal{A}^{[m]}$*-preserving automorphisms* of G. In other words, a permutation τ of $\mathcal{A}^{[m]}$ belongs to Γ, if and only if there exists an automorphism ϱ of G such that, for every $i \in \{1, \ldots, m\}$ and all $A \in \mathcal{A}^i$, we have $\tau(A) \in \mathcal{A}^i$ and $\tau(A) = \varrho(A)$.

We also call the $\mathcal{A}^{[m]}$-preserving automorphisms of G *marking-preserving*.

In Figure 3.3, the black lines demonstrate the vertices of an interval graph G, the rectangles demonstrate the sets in $\mathcal{A}$, and the colors of the sets demonstrate the partition $\mathcal{A}^{[m]}$ of $\mathcal{A}$ into $\mathcal{A}^1, \ldots, \mathcal{A}^5$ which contain the cliques on the vertices shown by the dashed black lines.

Note that AutomSET$(X; \mathcal{U}^{[m]})$ is a special case of AutomMarkedINT$(G; \mathcal{A}^{[m]})$ where the input interval graph G is a clique. Since the former problem is GI-complete, the latter problem is also GI-complete. Therefore, we show that AutomMarkedINT$(G; \mathcal{A}^{[m]})$ reduces to AutomSET$(X; \mathcal{U}^{[m]})$ and show that it can be solved in FPT-time parameterized by the maximum antichain size of $\mathcal{A}^{[m]}$.

3.1 AutomSET$(X; \mathcal{U}^{[m]})$ in FPT

In this section, we give a procedure efficiently computing the automorphism group of a set family under the assumption of bounded antichain size, which builds on ideas used already in our paper [5]. Recall from the introduction that we have just very recently found out that the same result shortly follows from an older paper [3] via the duality between elements and sets in set families. We formulate this standalone result as follows:

Theorem 3.3. AutomSET$(X; \mathcal{U}^{[m]})$ *is solvable in FPT-time parameterized by the maximum antichain size of* $\mathcal{U}^{[m]}$.

To stay on the more general side, we consider set families in the *multiset* setting, i.e. the same set U may occur in a family $\mathcal{U}^i$ multiple times without posing any additional difficulties besides having to observe the multiplicity as a label on a set. In regard of the above stated condition, a permutation τ of $\mathcal{U}^{[m]}$ then obviously has to map $U \in \mathcal{U}^{[m]}$ to a set $\tau(U) = U' \in \mathcal{U}^{[m]}$ such that the multiplicities of U and of U' in each of $\mathcal{U}^1, \ldots, \mathcal{U}^m$ are the same.

We first give the following simple observation on the input size of AutomSET $(X; \mathcal{U}^{[m]})$.

Observation 3.4. *If the maximum antichain size in the family* $\mathcal{U}^{[m]} \subseteq 2^X$ *equals* a*, then the input size of an instance of* AutomSET$(X; \mathcal{U}^{[m]})$ *is at most* $\sum_{B \in \mathcal{U}^{[m]}} |B| \in \mathcal{O}(a \cdot |X|^2)$*. The possible multiplicities of sets in the family* $\mathcal{U}^{[m]}$ *are negligible since they can be encoded as integer labels of multiple sets.*

The problem AutomSimpleSET$(X; \mathcal{U}^{[m]})$ is the same as AutomSET$(X; \mathcal{U}^{[m]})$ with the following condition on the input: For every set $B \in \mathcal{U}^{[m]}$, there is exactly one index $i \in \{1, \ldots, m\}$ such that $B \in \mathcal{U}^i$, and the multiplicity of B in $\mathcal{U}^i$ equals one. We prove the following that will simplify the proof of Theorem 3.3:

Proposition 3.5. *For suitable* $\mathcal{U}^{[m]}$, AUTOMSET$(X;\, \mathcal{V}^{[p]})$ *reduces in linear time to* AUTOMSIMPLESET$(X;\, \mathcal{U}^{[m]})$.

Recall that in *AutomSET*$(X;\, \mathcal{V}^{[p]})$, $\mathcal{V}^{[p]} = (\mathcal{V}^1, \ldots, \mathcal{V}^p)$ denotes the input partition of a set family $\mathcal{V} = \mathcal{V}^1 \cup \ldots \cup \mathcal{V}^p$, and it is analogous for AUTOMSIMPLESET$(X;\, \mathcal{U}^{[m]})$.

Proof. For every set $B \in \mathcal{V}$, we record the integer vector $m_B := (b_1, \ldots, b_p)$ where b_i is the multiplicity of B in the family $\mathcal{V}^i$. As noted already in Definition 3.1, any permutation τ in the solution of AUTOMSET$(X;\, \mathcal{V}^{[p]})$ must preserve this multiplicity vector; $m_B = m_{\tau(B)}$. Let $\mathcal{U}$ be the simplification of the (multi)family $\mathcal{V}$, i.e., without repetition of multiple member sets. We hence define a partition $\mathcal{U}^{[m]} = (\mathcal{U}^1, \ldots, \mathcal{U}^m)$ as the partition of $\mathcal{U}$ given by the equality of the vectors m_B. Then τ projects to a permutation in AUTOMSIMPLESET$(X;\, \mathcal{U}^{[m]})$.

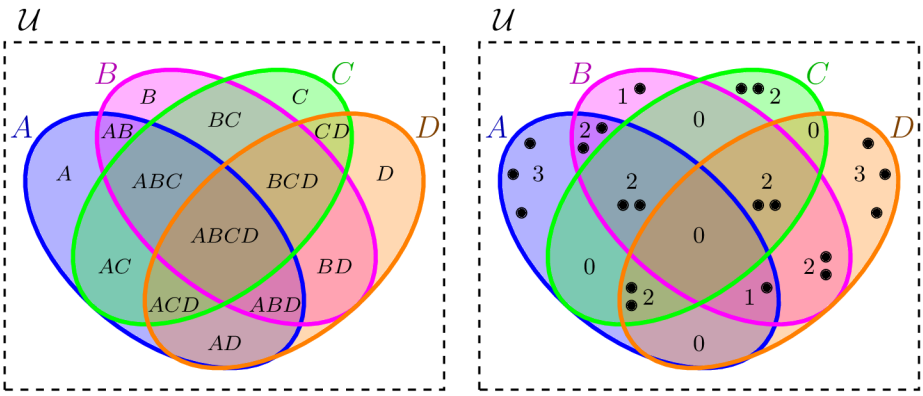
Conversely, for any permutation σ in the solution of AUTOMSIMPLESET$(X;\, \mathcal{U}^{[m]})$, we expand σ to permutations τ of $\mathcal{V}^{[p]}$ as follows. Every functional assignment $B \mapsto \sigma(B)$ is lifted, for each $i \in \{1, \ldots, m\}$, to all bijections of the set (possibly empty) of multiple copies of B in $\mathcal{V}^i$ to the set of multiple copies of $\sigma(B)$ in $\mathcal{V}^i$. Then every such τ belongs to the solution group of AUTOMSET$(X;\, \mathcal{U}^{[m]})$. □

On the other hand, we remark that in AUTOMSIMPLESET$(X;\, \mathcal{U}^{[m]})$, the number m of colors is not bounded, and so there is likely no easy way to reduce this problem to the uncolored case which has been implicitly solved in [5].

In the rest of this section, we prove Theorem 3.3 assuming the input is an instance of AUTOMSIMPLESET$(X;\, \mathcal{U}^{[m]})$ since it is sufficient by Proposition 3.5. There are two ingredients making our solution work. The first one is the *Cardinality Venn diagrams* that we introduce next, and the Babai's *tower-of-groups* approach introduced in Section 2.2.

Cardinality Venn diagrams. First of all, we review what Definition 3.1, specifically the words "there exists a permutation σ of X", mean for us. Since it is not much efficient to deal with the many permutations of X, we now show a simple fact that it will be enough to observe certain cardinalities to decide the existence of such permutation σ as required in the definition. For a set family $\mathcal{U}$, we call a *cardinality Venn diagram* of $\mathcal{U}$ the integer vector $\big(\ell_{\mathcal{U},\mathcal{U}_1} : \emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}\big)$ such that $\ell_{\mathcal{U},\mathcal{U}_1} := |L_{\mathcal{U},\mathcal{U}_1}|$ where $L_{\mathcal{U},\mathcal{U}_1} =$

$\bigcap_{A\in\mathcal{U}_1} A \setminus \bigcup_{B\in\mathcal{U}\setminus\mathcal{U}_1} B$. That is, informally, we record the cardinality of every internal cell of the Venn diagram of $\mathcal{U}$.



(a) (b)

Figure 3.4

Figure 3.4 illustrates the concept of a cardinality Venn diagram for the set family $\mathcal{U}$ shown by a dashed rectangle containing four sets A, B, C and D shown by colored ellipses. In Figure 3.4a, the labels of the cells show the sets in $\mathcal{U}$ containing that cell. In Figure 3.4a, the ground set elements of $\mathcal{U}$ are depicted with the dots, and the number in each cell is its *cardinality*, i.e. the number of ground set elements appearing in that cell. One can check, using the depicted cardinalities, that there exists a permutation of the ground set which permutes our sets (A, B, C, D) into (D, B, C, A), but there is no such permutation permuting (A, B, C, D) into (A, C, B, D).

For $\mathcal{U}_1 \subseteq \mathcal{U}$, let naturally $\varrho(\mathcal{U}_1) = \{\varrho(B) : B \in \mathcal{U}_1\}$. We have easily got:

Lemma 3.6 ([5])**.** *Let ϱ be a permutation of $\mathcal{U}$ over X. There exists a permutation σ of X such that, for every $B \in \mathcal{U}$, we have $\varrho(B) = \sigma(B)$, if and only if the cardinality Venn diagrams of $\mathcal{U}$ and of $\varrho(\mathcal{U})$ are the same (equal), meaning that $\ell_{\mathcal{U},\mathcal{U}_1} = \ell_{\mathcal{U},\varrho(\mathcal{U}_1)}$ for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$.*

Furthermore, for $\mathcal{U}' \subseteq \mathcal{U}$ such that $\varrho(\mathcal{U}') = \mathcal{U}'$, one can in $\mathcal{O}\big(|X| + \sum_{U\in\mathcal{U}'} |U|\big)$ time test the condition as above, i.e., whether the equalities $\ell_{\mathcal{U}',\mathcal{U}_1} = \ell_{\mathcal{U}',\varrho(\mathcal{U}_1)}$ hold for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}'$.

Proof. $\Rightarrow$ Suppose that there exists such a permutation σ of X. Then, for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$, every element of $L_{\mathcal{U},\mathcal{U}_1}$ is mapped by σ into $L_{\mathcal{U},\varrho(\mathcal{U}_1)}$, and so

the claim follows since σ is a permutation.

$\Leftarrow$ For any $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$, we have that $|L_{\mathcal{U},\mathcal{U}_1}| = |L_{\mathcal{U},\varrho(\mathcal{U}_1)}|$ which implies an existence of a bijection from $L_{\mathcal{U},\mathcal{U}_1}$ to $L_{\mathcal{U},\varrho(\mathcal{U}_1)}$. Since $L_{\mathcal{U},\mathcal{U}_1} \cap L_{\mathcal{U},\mathcal{U}_2} = \emptyset$ for $\mathcal{U}_1 \neq \mathcal{U}_2$, the composition of these bijections is sound and results in a permutation σ of X. Picking any $B \in \mathcal{U}$, we have $B = \bigcup \big\{L_{\mathcal{U},\mathcal{U}_1} : \{B\} \subseteq \mathcal{U}_1 \subseteq \mathcal{U}\big\}$, and hence $\sigma(B) = \bigcup \big\{\sigma(L_{\mathcal{U},\mathcal{U}_1}) : \{B\} \subseteq \mathcal{U}_1 \subseteq \mathcal{U}\big\} = \bigcup \big\{L_{\mathcal{U},\varrho(\mathcal{U}_1)} : \{B\} \subseteq \mathcal{U}_1 \subseteq \mathcal{U}\big\} = \varrho(B)$.

As for testing the condition '$\ell_{\mathcal{U}',\mathcal{U}_1} = \ell_{\mathcal{U}',\varrho(\mathcal{U}_1)}$ hold for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}'$', we loop through all elements $x \in X$, and for each x we record in $\mathcal{O}(n)$ time to which of the sets in $\mathcal{U}'$ this x belongs to. Summing the obtained records at the end precisely gives the $\mathcal{O}(|X|)$ nonzero values $\ell_{\mathcal{U}',\mathcal{U}_1}$ over $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}'$. We analogously compute the values $\ell_{\mathcal{U}',\varrho(\mathcal{U}_1)}$ over $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}'$, and then compare. $\square$

Our strategy for the proof of Theorem 3.3 is as follows.

- Let, for $n = |X|$ and every $j \in \{1, \ldots, m\}$, $\mathcal{U}^j = \bigcup_{c=1}^{n} \mathcal{U}^j_c$ be a partition of $\mathcal{U}^j$ into subfamilies of sets of cardinality $c \in \{1, \ldots, n\}$. We make the initial group Γ' (of permutations of $\mathcal{U} = \bigcup_{j=1}^{m} \mathcal{U}^j$) as the direct product of the symmetric groups (those of all permutations) on all nonempty families $\mathcal{U}^j_c$ over $j \in \{1, \ldots, m\}$ and $c \in \{1, \ldots, n\}$.
- We compute the subgroup $\Gamma \subseteq \Gamma'$ of those permutations which fulfill the condition '$\ell_{\mathcal{U},\mathcal{U}_1} = \ell_{\mathcal{U},\varrho(\mathcal{U}_1)}$ for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$' of Lemma 4.5. Then Γ will be, by Lemma 4.5, the solution to AutomSimpleSET$(X; \mathcal{U}^{[m]})$. At this point it will be quite important that each of the subfamilies $\mathcal{U}^j_c$ (as an antichain) has bounded cardinality.

The latter point, however, is not an easy task, and we will employ classical Babai's tower-of-groups procedure, introduced in Section 2.2, to "gradually refine" Γ' into Γ, ensuring that the conditions of Lemma 4.5 hold, in a sense, for more and more combinations of the subfamilies $\mathcal{U}^j_c$ until all are satisfied.

We introduce one more technical result which will be crucial in the gradual refinement of Γ' into Γ. In the setting of Lemma 4.5, we say that $\mathcal{U}' \subseteq \mathcal{U}$ is *Venn-good* with ϱ if $\ell_{\mathcal{U}',\mathcal{U}_1} = \ell_{\varrho(\mathcal{U}'),\varrho(\mathcal{U}_1)}$ holds true for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}'$, and we call such $\mathcal{U}_1$ a *witness* (of $\mathcal{U}'$ not being Venn-good) if $\ell_{\mathcal{U}',\mathcal{U}_1} \neq \ell_{\varrho(\mathcal{U}'),\varrho(\mathcal{U}_1)}$.

Lemma 3.7 ([5])**.** *Let ϱ be a permutation of a set family $\mathcal{U}$, and $\mathcal{U}' \subseteq \mathcal{U}$ be such that $\varrho(\mathcal{U}') = \mathcal{U}'$. If $\mathcal{U}'$ is not Venn-good with ϱ, then there exist*

$\mathcal{U}_2, \mathcal{U}_3 \subseteq \mathcal{U}'$ such that $|\mathcal{U}_2| \leq 2$ or $\mathcal{U}_2$ is an antichain in the inclusion, $\emptyset \neq \mathcal{U}_3 \subseteq \mathcal{U}_2$ and $\ell_{\mathcal{U}_2,\mathcal{U}_3} \neq \ell_{\varrho(\mathcal{U}_2),\varrho(\mathcal{U}_3)}$ (not Venn-good).

Proof. Choose $\mathcal{U}_2 \subseteq \mathcal{U}'$ such that $\mathcal{U}_2$ is not Venn-good with ϱ and it is minimal such by inclusion, and assume (for a contradiction) that there are $B_1, B_2 \in \mathcal{U}_2$ such that $B_1 \subseteq B_2$. If $\varrho(B_1) \not\subseteq \varrho(B_2)$, then already $\mathcal{U}_2 := \{B_1, B_2\}$ is not Venn-good (with a witness $\{B_1\}$), and so let $\varrho(B_1) \subseteq \varrho(B_2)$. Let $\mathcal{U}_3$ be a witness of $\mathcal{U}_2$ not being Venn-good, and for $j = 2, 3$ denote: $\mathcal{V}_j^0 := \mathcal{U}_j \setminus \{B_1, B_2\}$, $\mathcal{V}_j^1 := \big(\mathcal{U}_j \cup \{B_1\}\big) \setminus \{B_2\}$, $\mathcal{V}_j^2 := \big(\mathcal{U}_j \cup \{B_2\}\big) \setminus \{B_1\}$, $\mathcal{V}_j^3 := \mathcal{U}_j \cup \{B_1, B_2\}$.

Since $B_1 \subseteq B_2$ and $\varrho(B_1) \subseteq \varrho(B_2)$ (and so $L_{\mathcal{U}_2,\mathcal{V}_3^1} = \emptyset$), we easily derive

$$\ell_{\mathcal{U}_2,\mathcal{V}_3^1} = \quad 0 \quad = \ell_{\varrho(\mathcal{U}_2),\varrho(\mathcal{V}_3^1)}\,,$$

and since, by our minimality assumption, all three subfamilies $\mathcal{V}_2^0$, $\mathcal{V}_2^1$ and $\mathcal{V}_2^2$ are Venn-good,

$$\begin{aligned}
\ell_{\mathcal{U}_2,\mathcal{V}_3^0} = \ell_{\mathcal{U}_2\setminus\{B_1\},\mathcal{V}_3^0} = \ell_{\mathcal{V}_2^2,\mathcal{V}_3^0} &= \ell_{\varrho(\mathcal{V}_2^2),\varrho(\mathcal{V}_3^0)} = \ell_{\varrho(\mathcal{U}_2)\setminus\{\varrho(B_1)\},\varrho(\mathcal{V}_3^0)} \\
&= \ell_{\varrho(\mathcal{U}_2),\varrho(\mathcal{V}_3^0)}\,, \\
&\text{and} \\
\ell_{\mathcal{U}_2,\mathcal{V}_3^3} = \ell_{\mathcal{U}_2\setminus\{B_2\},\mathcal{V}_3^3\setminus\{B_2\}} = \ell_{\mathcal{V}_2^1,\mathcal{V}_3^1} &= \ell_{\varrho(\mathcal{V}_2^1),\varrho(\mathcal{V}_3^1)} = \ell_{\varrho(\mathcal{U}_2)\setminus\{\varrho(B_2)\},\varrho(\mathcal{V}_3^3)\setminus\{\varrho(A_2)\}} \\
&= \ell_{\varrho(\mathcal{U}_2),\varrho(\mathcal{V}_3^3)}\,.
\end{aligned}$$

Then, using the previous equalities, and the trivial observation $\ell_{\mathcal{V}_2^0,\mathcal{V}_3^0} = \ell_{\mathcal{U}_2,\mathcal{V}_3^0} + \ell_{\mathcal{U}_2,\mathcal{V}_3^1} + \ell_{\mathcal{U}_2,\mathcal{V}_3^2} + \ell_{\mathcal{U}_2,\mathcal{V}_3^3}$ with the analogous equality under ϱ, we conclude

$$\begin{aligned}
\ell_{\mathcal{U}_2,\mathcal{V}_3^2} &= \ell_{\mathcal{V}_2^0,\mathcal{V}_3^0} - \ell_{\mathcal{U}_2,\mathcal{V}_3^0} - 0 - \ell_{\mathcal{U}_2,\mathcal{V}_3^3} \\
&= \ell_{\varrho(\mathcal{V}_2^0),\varrho(\mathcal{V}_3^0)} - \ell_{\varrho(\mathcal{U}_2),\varrho(\mathcal{V}_3^0)} - 0 - \ell_{\varrho(\mathcal{U}_2),\varrho(\mathcal{V}_3^3)} = \ell_{\varrho(\mathcal{U}_2),\varrho(\mathcal{V}_3^2)}\,.
\end{aligned}$$

However, $\mathcal{U}_3 \in \{\mathcal{V}_3^0, \mathcal{V}_3^1, \mathcal{V}_3^2, \mathcal{V}_3^3\}$, and so one of the latter four derived equalities contradicts the assumption that $\mathcal{U}_3$ witnessed $\mathcal{U}_2$ not being Venn-good with ϱ. □

Figure 3.5a shows a Venn diagram on four sets B_1, B_2, B_3 and B_4, two set families $\mathcal{U}_3 \subseteq \mathcal{U}_2$ over those sets, and the cardinalities $a, \ldots, f \in \mathbb{Z}^+ \cup \{0\}$ of the corresponding Venn cells. Note that if $B_1 \in \mathcal{U}_3$, then $b = 0$ holds. Similarly, if $B_2 \in \mathcal{U}_3$, then both $a = 0$ and $b = 0$ hold. Moreover, all the following also hold: $\ell_{\mathcal{U}_2,\mathcal{V}_3^0} = e$, $\ell_{\mathcal{U}_2,\mathcal{V}_3^1} = 0$, $\ell_{\mathcal{U}_2,\mathcal{V}_3^2} = d$ and $\ell_{\mathcal{U}_2,\mathcal{V}_3^3} = c$.

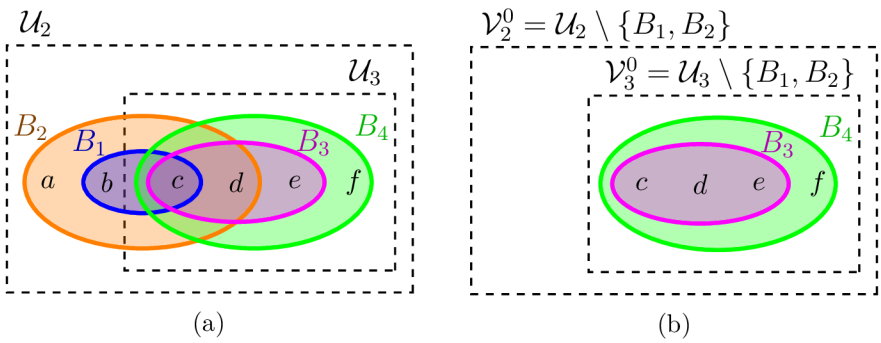


Figure 3.5

Figure 3.5b shows the Venn diagram on B_3 and B_4 and two set families $\mathcal{V}_3 \subseteq \mathcal{V}_2$ over those sets defined as in the proof of Lemma 3.7 where $\ell_{\mathcal{V}_0,\mathcal{V}_3^0} = c + d + e = \ell_{\mathcal{U}_2,\mathcal{V}_3^0} + \ell_{\mathcal{U}_2,\mathcal{V}_3^1} + \ell_{\mathcal{U}_2,\mathcal{V}_3^2} + \ell_{\mathcal{U}_2,\mathcal{V}_3^3}$.

Since we deal with set families of bounded-size antichains, Lemma 3.7 essentially tells us that either a family is Venn-good, or it contains a "small" subfamily not being Venn-good.

The collected ingredients show a clear road to computing the subgroup $\Gamma \subseteq \Gamma'$ from the above outlined proof strategy for Theorem 3.3. In every intermediate step of Babai's tower-of-groups $\Gamma' = \Gamma_0 \supseteq \Gamma_1 \supseteq \cdots \supseteq \Gamma_h = \Gamma$, we assume that the whole family $\mathcal{U}$ is not Venn-good (with some generator of Γ_i), and we use Lemma 3.7 to argue that there exists a small subfamily $\mathcal{U}_2$ of $\mathcal{U}$ which is not Venn-good with some generator of Γ_{i-1}. Such $\mathcal{U}_2$ can be found efficiently with a little trick (note that a brute-force approach would not give an FPT-time algorithm here), and then we define and compute the next group Γ_i as that of permutations for which $\mathcal{U}_2$ is Venn-good with Theorem 2.1. In this way, we eventually arrive at the group Γ such that $\mathcal{U}$ is Venn-good with every generator of Γ. The details follow.

Proof of Theorem 3.3. Recall that, for an instance of AutomSimpleSET(X; $\mathcal{U}^{[m]}$) over an n-element set X and for every $j \in \{1, \ldots, m\}$, we have defined a refined partition $\mathcal{U}^j = \bigcup_{c=1}^{n} \mathcal{U}_c^j$ where $\mathcal{U}_c^j$ consists of the sets from $\mathcal{U}^j$ of cardinality c. Since the maximum antichain size of $\mathcal{U}$, thus of $\mathcal{U}^{[m]}$, is a, we have that each $\mathcal{U}_c^j$ contains at most a distinct sets. For simplicity, we may assume $a \geq 2$ since the case of $a = 1$ is trivial. Let $\mathcal{W} := \{\mathcal{U}_c^j : 1 \leq j \leq m, 1 \leq c \leq n, \mathcal{U}_c^j \neq \emptyset\}$ be a system of all these nonempty families.

Algorithm 1 One (i-th) step of the computation of the subgroup $\Gamma \subseteq \Gamma'$

Require: a set family $\mathcal{U}$ of maximum antichain size $a \in \mathbb{N}$, a partition $\mathcal{W} \subseteq 2^{\mathcal{U}}$ of $\mathcal{U}$ into parts of size $\leq a$, and a group Γ_{i-1} (via a generator set) of permutations of $\mathcal{U}$ which set-wise stabilizes every part in $\mathcal{W}$.

Ensure: either a certificate that $\mathcal{U}$ is Venn-good with every generator of Γ_{i-1}; or
a subgroup $\Gamma_i \subsetneq \Gamma_{i-1}$ (via a generator set) such that, for $\mathcal{T} \subseteq \mathcal{U}$ which is the union of some at most a parts of $\mathcal{W}$, $\mathcal{T}$ is Venn-good precisely with every member of Γ_i (and not with members of $\Gamma_{i-1} \setminus \Gamma_i$).

Let $\mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_k\}$
$\mathcal{T} \leftarrow \emptyset$
repeat for $p := 1, 2, \ldots, a$:
  repeat for $q := 1, 2, \ldots, k+1$:
    if $q > k$ **then return** "$\mathcal{U}$ is Venn-good with all generators of Γ_{i-1}";
    $\mathcal{T}_1 \leftarrow (\mathcal{W}_1 \cup \mathcal{W}_2 \cup \cdots \cup \mathcal{W}_q) \cup \mathcal{T}$;
  until $\mathcal{T}_1$ is not Venn-good (cf. Lemma 4.5) with some generator of Γ_{i-1};
  $j_p \leftarrow q$;
  $\mathcal{T} \leftarrow \mathcal{W}_{j_1} \cup \mathcal{W}_{j_2} \cup \cdots \cup \mathcal{W}_{j_p}$;
until $j_p = 1$ or $p = a$;
Call the algorithm of Theorem 2.1 to compute the subgroup $\Gamma_i \subseteq \Gamma_{i-1}$, such that the membership test of $\varrho \in \Gamma_i$ checks whether $\mathcal{T}$ is Venn-good with ϱ (again Lemma 4.5);
return Γ_i

To solve the given instance, we start with the group $\Gamma' = \Gamma_0$ of permutations of $\mathcal{U}^{[m]}$ which is the direct product of the symmetric groups on all parts of $\mathcal{W}$. For $i := 1, 2, \ldots$, we iteratively call Algorithm 1 with $\mathcal{U}^{[m]}$, $\mathcal{W}$ and Γ_{i-1}, until the outcome (in h-th step) is that $\mathcal{U}^{[m]}$ is Venn-good with all generators of Γ_h. Note that the latter immediately gives that $\mathcal{U}^{[m]}$ is Venn-good with all permutations in Γ_h. By the assurance of Algorithm 1, $\Gamma = \Gamma_h$ contains all permutations for which $\mathcal{U}^{[m]}$ is Venn-good and indeed is the desired solution of AutomSimpleSET$(X; \mathcal{U}^{[m]})$.

So, it remains to finish two things; analyze and prove one call to Algorithm 1, and prove that the number h of steps is finite and not "too large". We start with the former. If $\mathcal{U}^{[m]}$ is Venn-good for every generator of Γ_{i-1}, then we

find this already in the first iteration of $p = 1$, on line 5. Hence we may further assume that $U^{[m]}$ is not Venn-good. By Lemma 3.7, there exists a subfamily $\mathcal{U}_2 \subseteq \mathcal{U}^{[m]}$ which is also not Venn-good, and $|\mathcal{U}_2| \leq \max(a, 2) \leq a$ since $\mathcal{U}_2$ is an antichain. Note that $\mathcal{U}_2$ thus intersects at most a parts of $\mathcal{W}$, which implies that there exist at most a parts of $\mathcal{W}$ whose union is not Venn-good.

Let $k \geq j'_1 > j'_2 > \cdots > j'_r \geq 1$ be an index sequence of length $r \leq a$ such that the subfamily $\mathcal{W}_{j'_1} \cup \mathcal{W}_{j'_2} \cup \cdots \cup \mathcal{W}_{j'_r}$ is not Venn-good with some generator of Γ_{i-1}, and the vector $(j'_1, j'_2, \ldots, j'_r)$ is lexicographically minimal of these properties. Then one can straightforwardly verify that $(j'_1, j'_2, \ldots, j'_r)$ is a prefix of (or equal to) the vector $(j_1, j_2, \ldots, j_p)$ computed by Algorithm 1. Consequently, the collection $\mathcal{T}$, when leaving the cycle on line 10, is not Venn-good with some generator of Γ_{i-1}. We look at the subset $\Gamma_i \subseteq \Gamma_{i-1}$ defined as on line 11. In particular, $\Gamma_i \subsetneq \Gamma_{i-1}$. The important point is that, as Γ_{i-1} set-wise stabilizes every part of $\mathcal{W}$, Γ_i is closed under composition of permutations, and so it forms a subgroup as expected by the algorithm.

Next, we verify the fulfillment of the assumptions of Theorem 2.1. Generators of $\Gamma_{i-1} = \Pi$ have been given to Algorithm 1. The ratio $|\Pi|/|\Pi_1|$, where $\Pi_1 = \Gamma_i$ in our case, can be bounded as follows (despite we do not know Γ_i yet): by standard algebraic arguments, $|\Gamma_{i-1}|/|\Gamma_i|$ equals the number of distinct cosets of the subgroup Γ_i in Γ_{i-1}. If we consider two automorphisms $\alpha, \beta \in \Gamma_{i-1}$ which are equal when restricted to $\mathcal{T}$ (recall that they set-wise stabilize $\mathcal{T}$), then the automorphism $\alpha^{-1}\beta$ determines a permutation of $\mathcal{U}$ which is identical on $\mathcal{T}$ (so it is Venn-good with $\alpha^{-1}\beta$), and hence $\alpha^{-1}\beta \in \Gamma_i$. The latter means that α and β belong to the same coset of Γ_i, and consequently, the number of distinct cosets is at most the number of distinct subpermutations on $\mathcal{T}$ possibly induced by Γ_{i-1}, that is at most $(a!)^a$ (at most the symmetric group on each of at most a parts of $\mathcal{W}$ which form $\mathcal{T}$). Therefore, we can finish one iteration on line Algorithm 1 in FPT-time with respect to $d = a$ by Theorem 2.1.

Lastly, we estimate the number of steps h in the refinement process $\Gamma' = \Gamma_0 \supsetneq \Gamma_1 \supsetneq \ldots \supsetneq \Gamma_h = \Gamma$. By Lagrange's group theorem, $|\Gamma_i|$ divides $|\Gamma_{i-1}|$, and since $\Gamma_{i-1} \neq \Gamma_i$, we have $|\Gamma_i| \leq \frac{1}{2}|\Gamma_{i-1}|$. Hence the number of strict refinement steps in our chain of subgroups is $h \leq \log_2 |\Gamma'|$. Since trivially $|\Gamma'| \leq (a!)^n$ (where $n = |X|$), we get $h = \mathcal{O}(na \log a)$. Therefore, the overall computation of the resulting group Γ of the problem instance of AutomSimpleSET$(X; \mathcal{U}^{[m]})$ is finished in FPT-time with respect to the parameter a, the maximum antichain size of the set family $\mathcal{U}^{[m]}$. We remark

that the only steps in our algorithm which require FPT-time (i.e., are possibly not of polynomial time) are the calls to the algorithm of Theorem 2.1. □

3.2 AutomMarkedINT(G; $\mathcal{A}^{[m]}$) in FPT

In this section, we give a procedure efficiently computing the automorphism group of marked cliques of an interval graph under the assumption of bounded antichain size, which is used in our paper [11] and builds on ideas used in Section 3.1. We formulate this as follows:

Theorem 3.8. AutomMarkedINT(G; $\mathcal{A}^{[m]}$) *is solvable in FPT-time with respect to the parameter a which is the maximum antichain size of $\mathcal{A}^{[m]}$.*

We first mention the input size of an AutomMarkedINT(G; $\mathcal{A}^{[m]}$) instance. If G is an n-vertex graph, the number of sets in $\mathcal{A}^{[m]}$ may be up to exponential in n. However, considering our parameter a equal to the maximum antichain size of $\mathcal{A}^{[m]}$, we easily get that there are at most an distinct sets in $\mathcal{A}^{[m]}$ (at most a of each cardinality between 1 and n). Hence we can always upper-bound the input size by $\big(|V(G)| + |E(G)| + \sum_{A\in\mathcal{A}^{[m]}} |A|\big) \in \mathcal{O}(a \cdot |V(G)|^2)$.

In the course of proving Theorem 3.8, we combine classical PQ-trees for capturing the internal structure of interval graphs [17] introduced in Section 2.5, and a colored extension of the algorithm for computing the automorphism group of a set family of bounded antichain size from Section 3.1.

We add that our algorithm solving Theorem 3.8 does not first compute the $\mathcal{A}^{[m]}$-preserving automorphism group of G, not even implicitly; with our approach it is easier this way. However, the algorithm can be straightforwardly extended to compute also the full $\mathcal{A}^{[m]}$-preserving automorphism group of G in FPT time with respect to a.

To prove Theorem 3.8, we revisit the PQ-trees of Section 2.5 from the point of view of marked interval graphs of Definition 3.2. While efficient handling of PQ-trees with an arbitrary marking seems basically infeasible, we deal with the special assumption of bounded antichains which appears very helpful within PQ-trees. Informally, we can say that the marked sets under this assumptions affect only a very small part of the whole PQ-tree of our marked graph G.

Recall that we have got an interval graph G and a marked clique family $\mathcal{A}^{[m]}$ of nonempty subsets of $V(G)$ such that every set $A \in \mathcal{A}^i$ where $i \in$

$\{1, \ldots, m\}$ induces a clique of G, $\mathcal{A}^{[m]}$ is a partition $(\mathcal{A}^1, \ldots, \mathcal{A}^m)$, and $\mathcal{A} := \mathcal{A}^1 \cup \ldots \cup \mathcal{A}^m$. Let the maximum antichain size in $\mathcal{A}$ be a. Let T be a PQ-tree of G, and recall what are inner vertices of G assigned to nodes of T and their rank from Section 2.5. We call a node p of T *clean* if the inner vertices assigned to p are disjoint from $\bigcup \mathcal{A}$, i.e. p does not belong to any marked set. The subtree rooted at p is then *clean* if p and all descendants of p in T are clean. It can be shown that the number of "incomparable" non-clean subtrees is bounded by the antichain size, but we skip the partial details since we actually prove much more in Lemma 3.9.

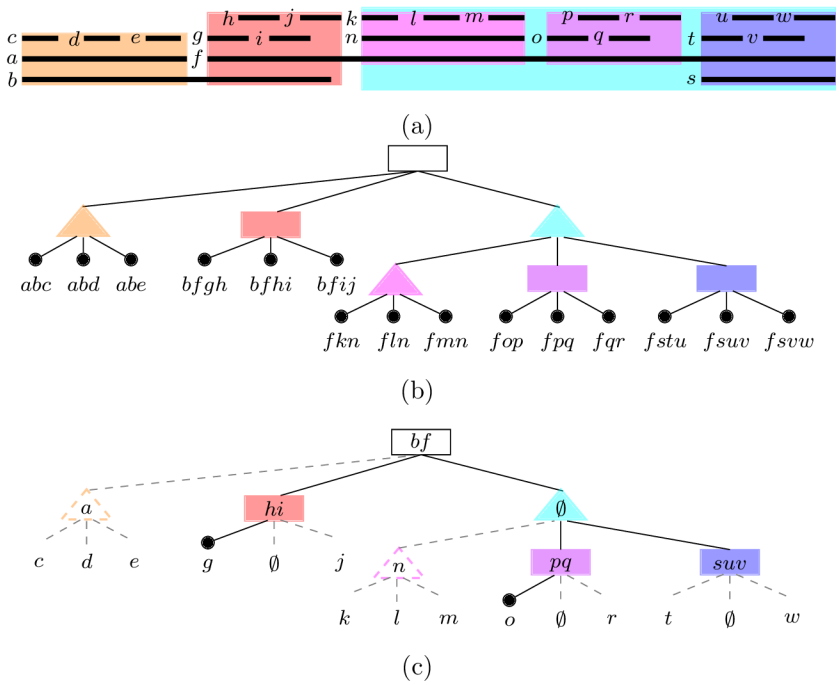


Figure 3.6

In Figure 3.6a, we see an interval graph G on the vertex set $V(G) = \{a, b, \ldots, w\}$, where the vertices are labelled on the left of the intervals in its representation. In Figure 3.6b, we see the PQ-tree T of G with the internal nodes (P-nodes are triangle-shaped and Q-nodes are rectangle-shaped) labelled with lists of the corresponding inner vertices, and with the leaves labelled by the maximal cliques of G. For an illustration, let $\mathcal{A}^{[m]} = \{\{b, f, g\}, \{f, o, p\}, \{s, u, v\}\}$ be a family of three marked sets in G. In Figure 3.6c, we see the subtree T' obtained from T by discarding all clean

subtrees (wrt. $\mathcal{A}^{[m]}$, now only outlined in dashed lines); this is used in the proof of Lemma 3.9.

Our answer to handling PQ-trees with marked sets is to precompute the complete isomorphism types of all clean subtrees in T, and then to introduce new marked sets (in addition to $\mathcal{A}^{[m]}$) which encode the (few) non-clean subtrees of T. Importantly, the latter can be done without increasing the maximum antichain size of the marked sets. Altogether, this is formulated in detail as follows:

Lemma 3.9. *Assume an instance of* AutomMarkedInt$(G; \mathcal{A}^{[m]})$*, where the maximum antichain size of* $\mathcal{A} := \mathcal{A}^1 \cup \ldots \cup \mathcal{A}^m$*, thus* $\mathcal{A}^{[m]}$*, equals* a*. There exist:*

- *a family* $\mathcal{B} = \mathcal{B}^1 \cup \ldots \cup \mathcal{B}^k = \mathcal{A}$ *of subsets of* $V(G)$ *where* $\mathcal{B}^{[k]} = (\mathcal{B}^1, \ldots, \mathcal{B}^k)$ *is a partition of* $\mathcal{A}^{[m]}$ *refining* $(\mathcal{A}^1, \ldots, \mathcal{A}^m)$*, and*
- *another family* $\mathcal{C} \subseteq 2^{V(G)}$ *of nonempty subsets of vertices where* $\mathcal{C}$ *is partitioned into* $\mathcal{C}^{[\ell]} = (\mathcal{C}^1, \ldots, \mathcal{C}^\ell)$

such that if a group Γ *is the solution to* AutomSet$(V(G); \mathcal{B}^{[k]}, \mathcal{C}^{[\ell]})$*, then* Γ *restricted to* $\mathcal{A}^{[m]}$ *is the sought solution of* AutomMarkedInt$(G; \mathcal{A}^{[m]})$*. Furthermore, the maximum antichain size in* $\mathcal{B} \cup \mathcal{C} = \mathcal{A} \cup \mathcal{C}$ *is also* a*, and the families* $\mathcal{B}$ *and* $\mathcal{C}$ *as well as the partitions* $\mathcal{B}^{[k]}$ *and* $\mathcal{C}^{[\ell]}$ *can be computed in linear time.*

Proof. Let T be a PQ-tree of our interval graph G. We first show how to refine the partition $(\mathcal{A}^1, \ldots, \mathcal{A}^m)$ of $\mathcal{A}^{[m]}$, using an auxiliary annotation assigned to the sets of $\mathcal{A}^{[m]}$. Observe that since every set $A \in \mathcal{A}^{[m]}$ induces a clique in G, the set A cannot intersect the inner vertices of two nodes of T which are incomparable in the tree order. This holds since A must be a subset of some maximal clique of G, and so if A intersects some inner vertex assigned to a node q, the max clique containing A must be among the descendant leaves of q in T (it may be useful to imagine this fact in an actual interval representation of G).

Consequently, the nodes with assigned inner vertices from $A \in \mathcal{A}^{[m]}$ lie on a root-to-leaf path of T, and so it is sound (and automorphism-invariant) to annotate A with the numbers of its elements which are the inner vertices assigned to nodes at each level of T from the root. For Q-nodes of T, we additionally annotate A with the numbers of elements which are inner vertices of each rank at every level. Clearly, by uniqueness of a PQ-tree up

to transformations, if an automorphism of G maps $A \in \mathcal{A}^{[m]}$ into $A' \in \mathcal{A}^{[m]}$, then the annotations of A and of A' are the same.

Then we "reduce" the PQ-tree T of G to a tree $T' \subseteq T$ by discarding all clean subtrees (with all their nodes) from it. Moreover, for every node q of T', we denote by T_q the subtree of T rooted at q and formed by all clean subtrees of q. We compute by Corollary 2.5 the canonical PQ-tree T_q^o of the subgraph G_q represented by T_q, and store T_q^o with the implied presentation of the (unlabelled) graph G_q as an annotation of the node q in T'. If q is a Q-node, we additionally store in the annotation of q the appropriate positions of the non-clean subtrees of q in T within the order of the sons of q in T_q^o.

For $q \in V(T')$, we now define the set C_q as the vertices of G which belong to q in T, i.e., C_q equals the union of all maximal cliques (of G) being in the descendant leaves of q. Since q belongs also to T', it is not clean, and so there is a set $A \in \mathcal{A}^{[m]}$ intersecting the inner vertices assigned to q. Observe that, whenever $A \in \mathcal{A}^{[m]}$ contains some of the inner vertices assigned to q, then $A \subseteq C_q$. This holds, as above, since A is a subset of some maximal clique of G, and this clique containing A must be among the descendant leaves of q in T.

Let $\mathcal{C} := \{C_q : q \in V(T')\}$. We claim that the maximum antichain size in $\mathcal{A} \cup \mathcal{C}$ is a as in $\mathcal{A}$ itself. To prove this claim, let $\mathcal{D} \subseteq \mathcal{A} \cup \mathcal{C}$ be an antichain such that $|\mathcal{D} \cap \mathcal{C}|$ is the least possible among antichains of the same size. If $\mathcal{D} \subseteq \mathcal{A}$, we are done, and so let $C_q \in \mathcal{D} \cap \mathcal{C}$ for some node $q \in V(T')$. By the previous, there exists $A_q \in \mathcal{A}$ such that A_q intersects some of the inner vertices of G assigned to q and $A_q \subseteq C_q$. Since $\mathcal{D} \ni C_q$ is an antichain, no set from $\mathcal{D}$ is contained in A_q. Assume that there is some $A \in \mathcal{D} \cap \mathcal{A}$ such that $A \supsetneq A_q$. Then A contains an inner vertex of q, and hence $A \subseteq C_q$ which contradicts $\mathcal{D}$ being an antichain. Finally, assume that there is $C_{q'} \in \mathcal{D} \cap \mathcal{C}$, where $q' \in V(T')$, such that $C_{q'} \supseteq A_q$. Then q' is not a descendant of q since $C_{q'} \not\subseteq C_q$ and, likewise, q' is not an ancestor of q. However, $C_{q'}$ contains an inner vertex of A assigned to q which contradicts the definition of inner vertices. Therefore, $(\mathcal{D} \setminus \{C_q\}) \cup \{A_q\}$ is an antichain again. Overall, we get that indeed $\mathcal{D} \subseteq \mathcal{A}$, and so $|\mathcal{D}| \leq a$.

We refine the partition $A^{[m]} = (\mathcal{A}^1, \ldots, \mathcal{A}^m)$ of $\mathcal{A}$ into $B^{[k]} = (\mathcal{B}^1, \ldots, \mathcal{B}^k)$ (for appropriate $k \geq m$) according as follows:

- For $i = 1, \ldots, m$, a pair of sets $A_1, A_2 \in \mathcal{A}^i$ fall into the same part of $(\mathcal{B}^1, \ldots, \mathcal{B}^k)$ if and only if the annotations of A_1 and A_2 are the same

in T.

Likewise, we partition $\mathcal{C}$ into subfamilies $C^{[\ell]} = (\mathcal{C}^1, \ldots, \mathcal{C}^\ell)$ (with an arbitrary number ℓ of parts) as follows:

- A pair of sets C_q and $C_{q'}$ fall into the same part if and only if the nodes q and q' have received in T' the same annotation. In particular, the canonical PQ-trees of the clean subtrees of q and q' are the same.

We now verify the claimed properties of the defined partitions of vertex set families. Recall that, by Lemma 3.9, the permutation group Γ is the solution to $\textsc{AutomSET}(V(G); \mathcal{B}^{[k]}, \mathcal{C}^{[\ell]})$. Let a permutation group Δ be the solution of $\textsc{AutomMarkedINT}(G; \mathcal{A}^{[m]})$.

For every permutation $\alpha \in \Delta$ of $\mathcal{A}^{[m]}$, there exists (by Definition 3.2) an underlying $\mathcal{A}^{[m]}$-preserving automorphism β of G. Then β induces a permutation on the maximal cliques of G which, in turn, gives an automorphism β' of the PQ-tree T. Note that in this context, naturally, an automorphism of a PQ-tree not only preserves the underlying rooted tree, but also the order of every Q-node up to reversal. As noted above, since β is an automorphism, α must preserve the annotations of the sets of $\mathcal{A}^{[m]}$, i.e., the partition $\mathcal{B}^{[k]}$. Since clean subtrees are also preserved by β', the restriction of β' gives an annotation-preserving automorphism of the tree T' by the definition of a canonical PQ-tree. Hence β' induces a permutation of $\mathcal{C}$ respecting the partition $\mathcal{C}^{[\ell]}$, which composed together with α on $\mathcal{A}^{[m]}$ gives a unique permutation $\gamma \in \Gamma$.

Conversely, let $\gamma \in \Gamma$ be a permutation on $\mathcal{A} \cup \mathcal{C}$ respecting both partitions $B^{[k]}$ and $C^{[\ell]}$. In particular, every set $C_q \in \mathcal{C}^i \subseteq \mathcal{C}^{[\ell]}$, where $q \in V(T')$ and $1 \leq i \leq \ell$, is mapped into $C_r = \gamma(C_q) \in \mathcal{C}^i$ where $r \in V(T')$. Henceforth, γ induces (as '$q \mapsto r$') an annotation-preserving permutation β_0 of $V(T')$. Since we have, for any two nodes $p, p' \in V(T')$, that p is an ancestor of p' if and only if $C_p \supseteq C_{p'}$, and γ preserves the inclusion relation, we conclude that β_0 is an annotation-preserving automorphism of T'. Since our annotation at every node of T' includes the canonical PQ-tree of the clean subtrees, and the ordering of the non-clean subtrees under Q-nodes, β_0 extends to a permutation β_1 of whole $V(T)$ which hence is an automorphism of the whole PQ-tree T. This gives an underlying automorphism β of the graph G. Since, moreover, γ preserves the annotations of the sets in $\mathcal{A}^{[m]}$, the automorphism β can be chosen such that it agrees with the permutation γ on $\mathcal{A}^{[m]}$. We have got the restriction of γ to $\mathcal{A}^{[m]}$ in the group Δ.

It remains to analyze the runtime of the described reduction, i.e., of the computation of the partitions $\mathcal{B}^{[k]}$ and $\mathcal{C}^{[\ell]}$. For that we first compute a PQ-tree T of our interval graph G in linear time by Theorem 2.4. From T, we straightforwardly compute the anntations of the sets in $\mathcal{A}^{[m]}$, as specified above. Then we easily in linear time identify all clean subtrees of T, and hence get the tree T'. The annotations of the nodes of T' are computed again in linear time using Corollary 2.5 applied to their clean subtrees, and since the considered clean subtrees in this computation are pairwise disjoint, the overall runtime is linear in the size of G. Knowing the annotations, we then easily output the partitions $\mathcal{B}^{[k]}$ of $\mathcal{A}^{[m]}$ and $\mathcal{C}^{[\ell]}$. □

Proof of Theorem 3.8. We first apply Lemma 3.9, and transform an instance of AutomMarkedINT(G; $\mathcal{A}^{[m]}$) into an instance of AutomSET($V(G)$; $\mathcal{B}^{[k]}$, $\mathcal{C}^{[\ell]}$) of the same parameter value a. Then, we use Theorem 3.3 to solve the latter in FPT-time with respect to a, and straightforwardly output the corresponding restricted solution of AutomMarkedINT(G; $\mathcal{A}^{[m]}$). □

We have introduced AutomMarkedINT(G; $\mathcal{A}^{[m]}$) problem and thus finished this chapter by proving that it can be solved in FPT-time with respect to a where a is the maximum antichain size of the marked sets.

Chapter 4

S_d-graph Isomorphism

In this chapter, we study the complexity of the isomorphism problem for S_d-graphs when parameterized by the maximum clique size and then by the number d, and prove that these cases can be solved in `FPT`-time. The presented results are based on our conference paper [5] and its journal version [10]. In the course of our exposition, we also show that S_d-graph isomorphism is computationally at least as hard as the isomorphism problem of posets of bounded width. We start with the following alternative `GI`-completeness reduction:

Proposition 4.1 (originally shown in [63])**.** *The isomorphism problem of proper S_d-graphs with d on the input (and hence, of all proper and general H-graphs) is `GI`-complete.*

Proof. Let G_1 and G_2 be two arbitrary graphs on the same number of at least 4 vertices. We construct G'_i, $i = 1, 2$, as follows: subdivide every edge with a new vertex, and then make a clique on the original vertex set $V(G_i)$. Then, $G_1 \simeq G_2$ if and only if $G'_1 \simeq G'_2$; this is easy since any isomorphism of G'_1 and G'_2 must map $V(G_1)$ to $V(G_2)$ due to vertex degrees. Since each G'_i is an S_d-graph for $d = |E(G_i)|$, with the central clique on $V(G_i)$, solving their isomorphism would solve also the isomorphism of G_1 and G_2. Furthermore, G'_i is a proper $S_{d'}$-graph for $d' = |E(G_i)| + |V(G_i)|$, where the additional $|V(G_i)|$ rays of $S_{d'}$ are used to make the subgraphs representing the central clique pairwise incomparable. □

Figure 4.1 illustrates the above reduction. Figure 4.1a and Figure 4.1b show two arbitrary graphs G_1 and G_2, Figure 4.1c and Figure 4.1d show the subdi-

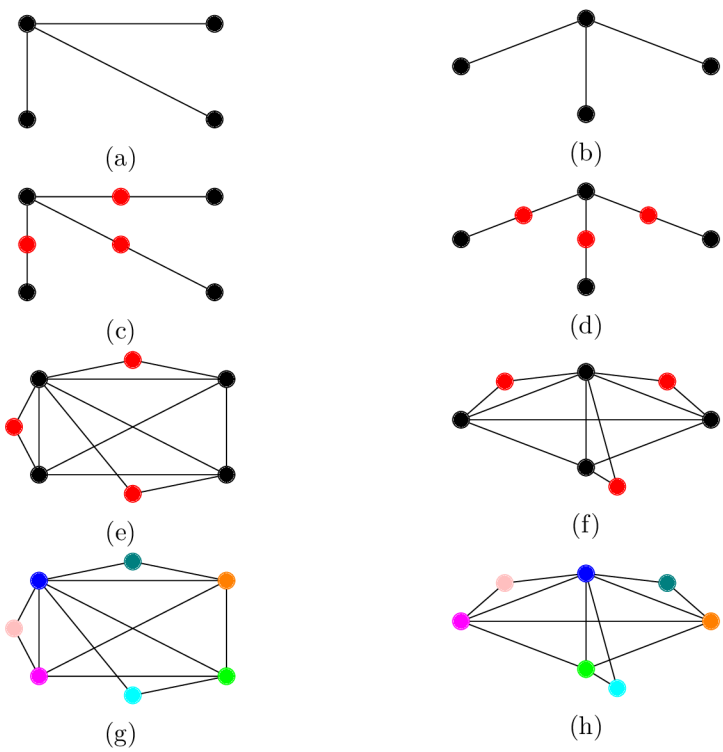


Figure 4.1

vision of the previous graphs, Figure 4.1e and Figure 4.1f show the obtained graphs $G_1{}'$ and $G_2{}'$, and Figure 4.1g and Figure 4.1h show an isomorphism between $G_1{}'$ and $G_2{}'$ by the coloring of their vertices, respectively.

4.1 Isomorphism of S_d-graphs of bounded clique size

In this section, we focus on an easy case where the maximum clique size of the input S_d-graphs is bounded by the parameter p, and shown that it can be solved in FPT-time only parameterized by p using a straightforward approach. In fact, this easy case does not require us to bound d, and so we will use p as the only parameter here.

Assume that we are given two isomorphic S_d-graphs G and H, and f is an isomorphism between G and H. Let C be a maximal clique of G, and let

$D = f(C)$ be the corresponding maximal clique of H. Then each connected component X_i of $G - C$ is mapped to corresponding $Y_i = f(X_i)$ of $H - D$. Moreover, if C happens to be the central clique of some S_d- representation of G, then the subgraph induced by $C \cup X_i$ is an interval graph and we may use the interval graph isomorphism algorithm [17] to compare between $G[C \cup X_i]$ and $H[D \cup Y_i]$. Finally, even if we do not know the right central clique C and corresponding D in advance, we may use the fact that S_d graphs are chordal. Any chordal graph with n vertices has at most n maximal cliques which can be listed in linear time [59]. So we may simply try all possible pairs C, D and not depend on particular S_d-representations of G and H. The fine details are given in Algorithm 2.

The complexity analysis of Algorithm 2. Let n be the number of vertices of both G and H. The maximal clique collections $\mathcal{S}$ and $\mathcal{T}$ of them (each of length $\leq n$) are found using simplicial vertex elimination in time $\mathcal{O}(pn)$. Comparing the cardinalities of the members of $\mathcal{S}$ and $\mathcal{T}$ takes $\mathcal{O}(n \log n)$ time. Finding the connected components of $G - C$ (also of $H - D$) takes linear time and testing for interval graphs is also in linear time [17], and so a suitable clique $C \in \mathcal{S}$ can be fixed in time $\mathcal{O}(n^3)$.

Then we loop through $D \in \mathcal{T}$ and all labelings of D, in the worst case, which means at most $p! \cdot n$ iterations. Comparing the interval graphs $G[C \cup X_i]$ and $H[D \cup Y_j]$ to isomorphism takes linear time for each pair X_i, Y_j, but we can do a better runtime analysis since the following holds:

$$\sum_{i=1}^{k} |C \cup X_i| \leq |C|n + n \leq pn + n$$

Therefore, comparing all the pairs of graphs $G[C \cup X_i]$ and $H[D \cup Y_j]$ takes altogether $\mathcal{O}(pn^2)$. Checking for a perfect matching of symmetric (i.e., isomorphic) pairs among them is then trivially in time $\mathcal{O}(n^2)$.

Thus, the overall complexity of this algorithm is

$$\mathcal{O}(n^3 + p! \cdot n \cdot (pn^2 + n^2)) = \mathcal{O}(p! \cdot pn^3)$$

which belongs to `FPT` with respect to p.

Algorithm 2 Isomorphism test for S_d-graphs with bounded clique size (G,H)

Require: Given two S_d-graphs G and H (their representations not required), the parameters d of S_d and p bounding the maximal clique size of G and H.

Ensure: Result of the isomorphism test between G and H.

Find the maximal clique collections $\mathcal{S}$ and $\mathcal{T}$ of the chordal graphs G and H;
if $|\mathcal{S}| \neq |\mathcal{T}|$, or the cardinalities of members of $\mathcal{S}$ and $\mathcal{T}$ do not match **then**
    return "G and H are not isomorphic";
repeat for each maximal clique $C \in \mathcal{S}$:
    Find the connected components $X_1, X_2, \ldots, X_k$ of $G - C$, assuming the equivalent connected components are joined into single bridge(s) of C;
until all $G[C \cup X_i]$, $1 \leq i \leq k$, are interval graphs;
Fix any bijective labeling $C \to \{1, 2, \ldots, |C|\}$ on the vertices of C;
for each $D \in \mathcal{T}$ with $|C| = |D|$ **do**
    Find the connected components $Y_1, Y_2, \ldots, Y_l$ of $H - D$, assuming the equivalent connected components are joined into single bridge(s) of D;
    if $k = l$, and all $H[D \cup Y_j]$, $1 \leq j \leq k$, are interval graphs **then**
        for each bijective $D \to \{1, 2, \ldots, |D|\}$ on the vertices of D **do**
            for each pair X_i and Y_j, $1 \leq i, j \leq k$ **do**
                Compare the interval graphs $G[C \cup X_i]$ and $H[D \cup Y_j]$ to isomorphism, respecting the labels of C and D;
                Mark isomorphic pairs of them as "symmetric";
            $\mathcal{Y} \leftarrow \{Y_1, Y_2, \ldots, Y_k\}$;
            for each X_i, $1 \leq i \leq k$ **do**
                Find greedily $Y_j \in \mathcal{Y}$ symmetric to X_i, and delete Y_j from $\mathcal{Y}$;
            if successful for all X_i, $1 \leq i \leq k$ **then**
                return "G and H are isomorphic";
return "G and H are not isomorphic";

Note that efficiency of this procedure strongly depends on the bound $q \leq p$ of maximum clique size of G and H. In the general case, we could be processing up to $n!$ distinct permutations of the vertices of D which would not give a polynomial time algorithm. On the other side, d can be arbitrary since this complexity is independent of the poset width d, and we do not need S_d-representations.

Theorem 4.2. *Isomorphism of S_d-graphs with n vertices, where the maximal clique sizes are bounded by a fixed parameter p, can be solved in $\mathcal{O}(p! \cdot pn^3)$ time, which belongs to* `FPT` *with respect to p (and regardless of d).*

Proof. For runtime analysis, see the previous paragraph. We show that Algorithm 2 correctly decides whether two S_d-graphs G and H with bounded clique size are isomorphic. First, assume that Algorithm 2 returns "G and H are isomorphic". Thanks to respecting the fixed labelings of C and D when comparing each pair $G[C \cup X_i]$ and $H[D \cup Y_j]$ to isomorphism, we indeed have that $G \simeq H$. Note that this conclusion holds true regardless of whether the fixed clique C is the central clique in some S_d-representation of G.

Let, on the other hand, G and H be two isomorphic S_d-graphs with cliques of size $\leq p$, and f be their isomorphism. Since G has an S_d-graph representation, an admissible clique C must be found on line 6 of the algorithm (as the central clique of the representation, if not before). Then, for the fixed labeling of C and the image of this labeling in D under f, all the isomorphism tests between $G[C \cup X_i]$ and $H[D \cup Y_j]$ where $Y_j = f(X_i)$ on line 13 must succeed. Hence the algorithm returns "G and H are isomorphic", as desired. □

4.2 S_d-graphs to posets of bounded width

Recall the characterization of S_d-graphs from Section 2.8, involving a poset on graph components. In the previous section, we were able to try all possible labelings of the central clique vertices. However, it means to process up to $n!$ labelings which we can not afford in the general case. It is a simple observation that to improve the algorithm, we need to solve poset isomorphism, and to justify our observation, we prove that solving the isomorphism of S_d-graphs involves solving the isomorphism of posets of bounded width d by giving a polynomial reduction.

Given a poset P of bounded width d with n elements, we form an S_d-graph G using the following procedure:

1. Model P by its set inclusions between the sets $M_1, \ldots, M_n$ where $M_i = \{j \in P : j \preceq_P i\}$ for all $i \in P$.
2. Take the union $M = \bigcup_{i \in P} M_i$ and form the central clique C of G with $|M| + 2$ vertices where these two dummy vertices are the markers of C being the unique maximum clique of G.
3. For each M_i, add a vertex v_i to G with the attachment set in C corresponding to exactly M_i.

Note that two dummy vertices are added to C to ensure that C is the unique maximum-size clique of G.

Figure 4.2 illustrates the above reduction. Figure 4.2a shows an example poset P (its Hasse diagram) of width $d = 3$, and its levels, Figure 4.2b shows the corresponding S_d-graph G where the dots and two circles inside cyan dashed ellipse correspond to the central (and unique) maximum clique C of G where the dummy vertices are depicted by circles and the attachment set of each vertex v_i is shown by a colored ellipse for $i \in \{1, \ldots, 9\}$, and Figure 4.2c shows the poset P' (its Hasse diagram) on the attachment sets of the connected components of $G - C$ where $P \simeq P'$ by mapping i to v_i for all $i \in \{1, \ldots, 9\}$.

This reduction can be carried out in polynomial time, and hence:

Theorem 4.3. *The isomorphism problem of posets of width d reduces in polynomial time to the isomorphism problem of S_d-graphs. This holds even if the poset elements are colored and the colors must be preserved by the isomorphism.*

Proof. Let P and Q with n elements be two posets of width d, and G and H be the S_d-graphs formed from P and Q, respectively, by the construction given in Section 4.2. Note that G and H have $2n+2$ vertices and a maximum clique of size $n + 2$ each. This maximum clique is the central clique of their S_d-representations. As mentioned above, a chain cover of P (and of Q) of size $\leq d$ determines a distribution of the vertices v_i of G to the d rays of the representation, such that their attachment vertices in C also form a chain by inclusion on each ray. Hence, this gives a valid S_d-representation of G.

We first assume that P and Q are isomorphic under a bijection $f : P \to Q$. Then f directly defines a bijection from the central clique C and a bijection

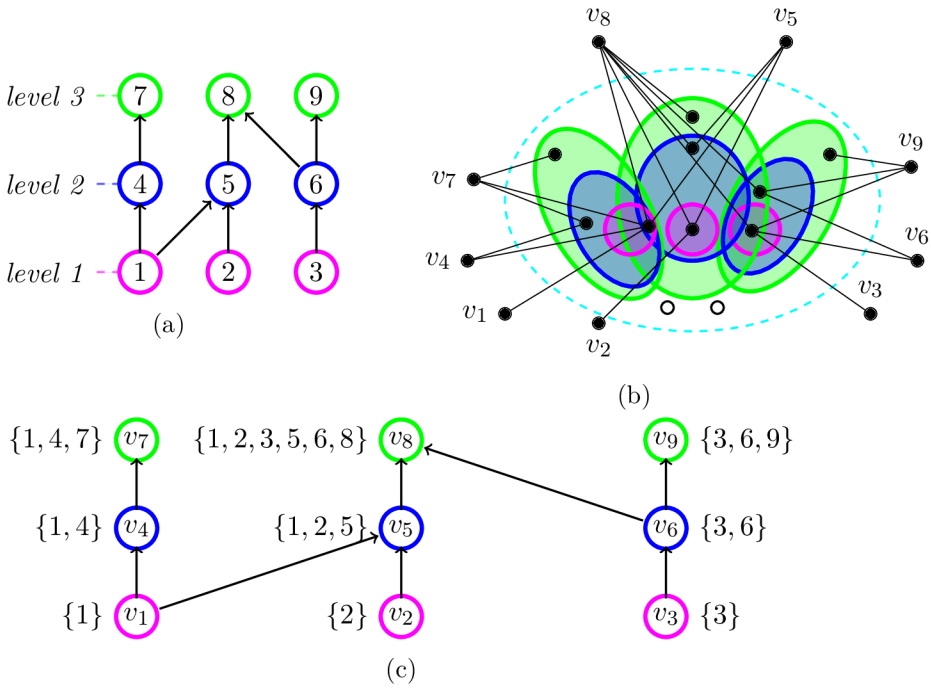


Figure 4.2

from $\{v_1, \ldots, v_n\}$ to those of H. The composition of these two mappings is clearly a graph isomorphism between G and H.

Assume that G and H are isomorphic under a bijection $g : V(G) \to V(H)$. Since $C \subseteq V(G)$ and $D \subseteq V(H)$ are the unique maximum-size cliques in G and H, we have that $G(C) = D$. Therefore, a restriction of g is a bijection from $V(G - C) = \{v_1, \ldots, v_n\}$ to $V(H - D) = \{w_1, \ldots, w_n\}$. With respect to the construction of G and H, this induces a bijection between the sets $\{M_1, \ldots, M_n\}$ of P (as represented by $v_1, \ldots, v_n$) and the sets $\{N_1, \ldots, N_n\}$ of Q (as represented by $w_1, \ldots, w_n$) and, in turn, a bijection $g' : P \to Q$. If $a \preceq_P b$, then $M_a \subseteq M_b$ and the neighborhood of v_a in G is included in the neighborhood of v_b. Then the neighborhood of $g(v_a)$ in H is included in the neighborhood of $g(v_b)$ and, in regard of the induced bijection g', $N_{g'(a)} \subseteq N_{g'(b)}$ and $g'(a) \preceq_Q g'(b)$. The converse implication holds the same way, and so the posets P and Q are isomorphic.

Lastly, we remark that if the input posets P and Q are given with colored elements, and we are looking for color-preserving isomorphism, we can also

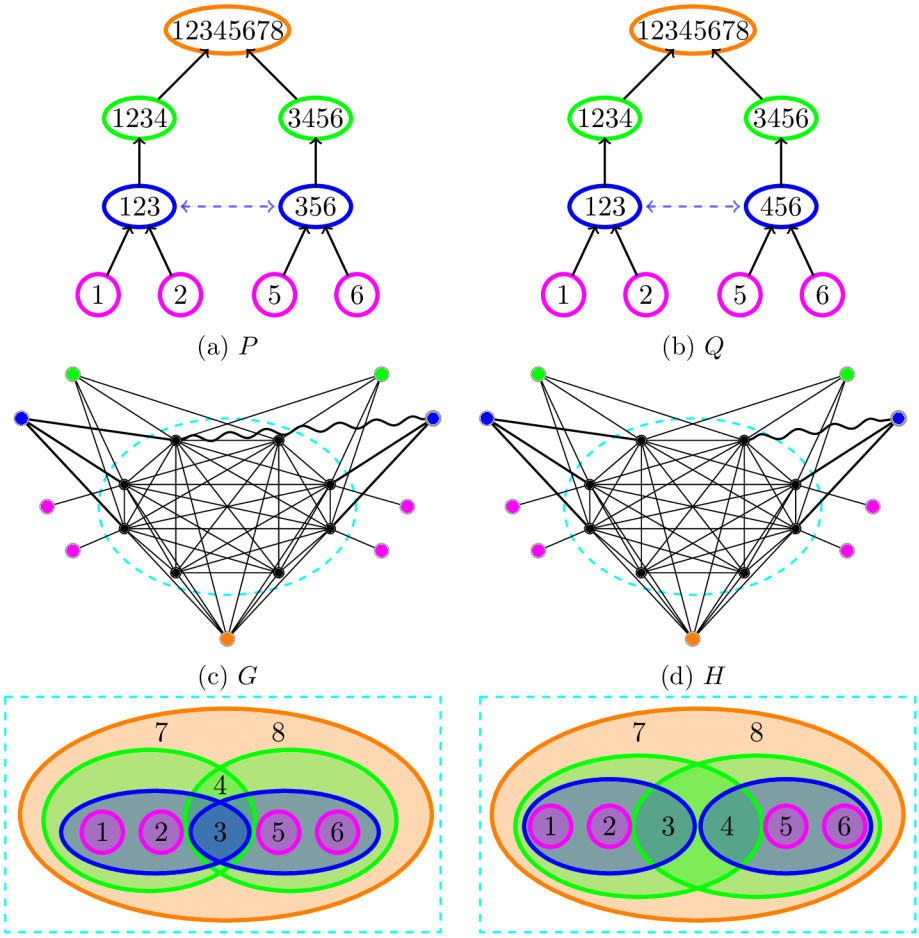
use the same reduction approach: If j is a poset element of color c, then we represent M_j in G not by a single vertex v_j, but by a copy of the clique K_c. We also enlarge the number of dummy vertices in C in such case. $\square$

We conclude that the isomorphism problem for posets of width d can be solved in FPT if S_d-graph isomorphism can be solved in FPT, with respect to d.

4.3 Isomorphism of S_d-graphs

In this section, we focus on general S_d-graphs. As our reduction from the previous section suggests, we first focus on the isomorphism problem for posets of bounded width. Posets of bounded width d are a subclass of d-bounded color multiplicity graphs of which automorphism group can be found in FPT-time using the group theoratical approach by Babai which we introduce in Section 2. Note that poset isomorphism is GI-complete when the width is not bounded [62]. We use the described approach to find the automorphism group of our underlying posets. However, poset isomorphism is a structural equivalence which does not guarantee a bijection between the vertex sets of the considered graphs.

Figure 4.3a and Figure 4.3b show two isomorphic posets P and Q, respectively. Even though $P \simeq Q$, there exists no bijection between their ground set elements since the blue poset elements in Figure 4.3a have the ground set element 3 in common while the blue poset elements in Figure 4.3b have no ground set element in common. Figure 4.3c and Figure 4.3d show the corresponding S_d-graphs G and H with the maximal cliques C and D on the vertices inside the cyan dashed ellipses, respectively, obtained using the reduction given in Section 4.2 without adding the dummy vertices since both C and D are the unique maximum cliques in both graphs. The thick edges show the blue attachments, and the curly thick edges show the reason why P and Q differ. Figure 4.3e and Figure 4.3f show the corresponding Venn diagrams for P and Q, respectively, in which the difference between P and Q is immediate.



(a) P (b) Q

(c) G (d) H

(e) Venn diagram for P (f) Venn diagram for Q

Figure 4.3

The above example shows that we need to combine Babai's procedure to compute the automorphism group of bounded color multiplicity graphs with additional checks to ensure a bijection between the maximal cliques of our S_d-graphs. Recall that when the maximal clique sizes are not bounded, we can not explicitly try all possible labelings as in the simple case given in Section 4.1. Therefore, we will utilize an isomorphism check between relevant Venn diagrams to test the existence of a bijection between the clique vertices which is possible using the results we give in Chapter 3. The outline of our FPT-approach is given next as Procedure 4.4, and in detail

given as Algorithm 3.

Procedure 4.4. Given two S_d-graphs G and H, we apply the following procedure for each pair of suitable maximal cliques C of G and D of H:

1. Find the connected components $\mathcal{X}$ of $G - C$ and $\mathcal{Y}$ of $H - D$, and construct the posets P and Q on the components of $\mathcal{X}$ and $\mathcal{Y}$.
2. Make the disjoint union $R = P \uplus Q$, and compute the levels of R.
3. For each pair of components $Z, Z' \in \mathcal{X} \cup \mathcal{Y}$ on the same level of R, compare the interval subgraphs $K(Z)$ and $K(Z')$ to isomorphism and color Z and Z' by the computed isomorphism type.
4. Compute the color-preserving automorphism group Π of R.
5. Compute the subgroup Π' of Π which contains the automorphisms of R respecting possible bijections between C and D as well as the attachment chains of isomorphic components. This is done by checking consistency in Venn diagrams defined by the involved attachments in C and D.
6. If P and Q are swapped in some automorphism of Π', return that G and H are isomorphic.

If the procedure does not return that G and H are isomorphic for some pair (C, D), then output that G and H are not isomorphic.

The key observations which make this procedure work are given below.

We first give the following Lemma 4.5 which shows that an automorphism of R is consistent on the maximal cliques and the attachment chains of components if and only if this automorphism is also an automorphism of the corresponding Venn diagrams corresponding to poset elements.

Lemma 4.5 ([5]). *Given $K = G \uplus H$ and C, D, the sets $\mathcal{X}$ and $\mathcal{Y}$, and the posets P, Q and R (on the ground set $C \cup D$) as above, let $\mathcal{U}$ be the attachment collection of $\mathcal{X} \cup \mathcal{Y}$ in K, and assume an automorphism ϱ of R and the corresponding permutation $\tilde{\varrho}$ of $\mathcal{U}$. There is an automorphism f of K such that $f(C \cup D) = C \cup D$ and, for every component $Z \in \mathcal{X} \cup \mathcal{Y}$, f maps $V(Z)$ to $V(\varrho(Z))$, if and only if the cardinality Venn diagrams of $\mathcal{U}$ and of $\tilde{\varrho}(\mathcal{U})$ are the same, meaning that the cardinalities of all cells are unchanged under the action of $\tilde{\varrho}$, i.e. $\ell_{\mathcal{U},\mathcal{U}_1} = \ell_{\mathcal{U},\tilde{\varrho}(\mathcal{U}_1)}$ for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$.*

Proof. $\Rightarrow$ Suppose that there exists an automorphism f of K such that

$f(C \cup D) = C \cup D$ and, for every component $Z \in \mathcal{X} \cup \mathcal{Y}$, f maps the vertices of Z to the vertices of $\varrho(Z)$. Then, each pair $K(Z)$ and $K(\varrho(Z))$ are isomorphic interval graphs, and if there are a distinct neighborhoods $N_1(Z), \ldots, N_a(Z)$ of vertices of Z in $C \cup D$, then there are a neighborhoods $N_1(\varrho(Z)), \ldots, N_a(\varrho(Z))$ of vertices of $\varrho(Z)$ in $C \cup D$ determined by the restriction of f to $C \cup D$. Since f is an automorphism of K, indeed, $\tilde{\varrho}(N_i(Z)) = N_i(\varrho(Z))$ for $1 \leq i \leq a$. Moreover, by our assumption, if $v \in N_i(Z)$ then $f(v) \in N_i(\varrho(Z))$, and vice versa.

Now consider arbitrary $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$. By the previous; if $v \in \bigcap_{A \in \mathcal{U}_1} A \setminus \bigcup_{B \in \mathcal{U} \setminus \mathcal{U}_1} B$, then $f(v) \in \bigcap_{A \in \mathcal{U}_1} \tilde{\varrho}(A) \setminus \bigcup_{B \in \mathcal{U} \setminus \mathcal{U}_1} \tilde{\varrho}(B) = \bigcap_{A \in \tilde{\varrho}(\mathcal{U}_1)} A \setminus \bigcup_{B \in \mathcal{U} \setminus \tilde{\varrho}(\mathcal{U}_1)} B$ (since $\tilde{\varrho}$ is a permutation of $\mathcal{U}$), and vice versa. Consequently, $\ell_{\mathcal{U},\mathcal{U}_1} = \ell_{\mathcal{U},\tilde{\varrho}(\mathcal{U}_1)}$ for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$.

$\Leftarrow$ Suppose that $\ell_{\mathcal{U}_1} = \ell_{\tilde{\varrho}(\mathcal{U}_1)}$ holds for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$. Then, in particular, for every such $\mathcal{U}_1$ there exists a bijection from $\bigcap_{A \in \mathcal{U}_1} A \setminus \bigcup_{B \in \mathcal{U} \setminus \mathcal{U}_1} B$ to $\bigcap_{A \in \tilde{\varrho}(\mathcal{U}_1)} A \setminus \bigcup_{B \in \mathcal{U} \setminus \tilde{\varrho}(\mathcal{U}_1)} B$. The composition of these bijections results in a permutation f_0 of $C \cup D$. Informally stating, f_0 respects all attachment sets of all components $Z \in \mathcal{X} \cup \mathcal{Y}$ under the permutation ϱ. Hence, for every $Z \in \mathcal{X} \cup \mathcal{Y}$, there is an isomorphism f_Z from $K(Z)$ to $K(\varrho(Z))$ which extends f_0. And, since the members of $\mathcal{X} \cup \mathcal{Y}$ are pairwise disjoint components of $K - (C \cup D)$, the composition of all f_Z over $Z \in \mathcal{X} \cup \mathcal{Y}$ is well-defined and it is an automorphism of the graph K satisfying the desired properties. $\square$

In the Venn diagram of the disjoint union R on at most $2n$ sets, there can be up to 2^{2n} cells which is exponential. However, in our case, only up to $2n$ cells are non-empty since they correspond to the maximal clique vertices in $C \cup D$, and thus we can check these cells efficiently by the following.

Lemma 4.6 ([5]). *One can check in $\mathcal{O}(n^2)$ time whether the cardinality Venn diagrams of $\mathcal{U}$ and $\tilde{\varrho}(\mathcal{U})$ are the same.*

Proof. We loop through all vertices w of $C \cup D$, and for each w we record in $\mathcal{O}(n)$ time to which of the sets in $\mathcal{U}'$ this w belongs to. Summing the obtained records at the end precisely gives the $\mathcal{O}(n)$ nonzero values $\ell_{\mathcal{U}',\mathcal{U}_1}$ over $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}'$. We analogously compute the $\mathcal{O}(n)$ nonzero values $\ell_{\mathcal{U}',\tilde{\varrho}(\mathcal{U}_1)}$ over $\mathcal{U}_1$, and then compare the two sets of values with respect to each $\mathcal{U}_1$ and matching $\tilde{\varrho}(\mathcal{U}_1)$. $\square$

Since our posets are of bounded width d, the maximum antichain size of the underlying posets, and the maximum antichain size of the corresponding

Venn diagrams is bounded by d. Recall the definition of a Venn-good set family from Chapter 3. Consider $\mathcal{U}$ being the attachment set collection of the connected components in $\mathcal{X}$, we get the following as a direct corollary which shows that two cardinality Venn diagrams are the same if and only if the same holds for every antichain of sets in them by Lemma 3.7.

Corollary 4.7. *Let a poset R, its automorphism ϱ, attachment collection $\mathcal{U}$ and permutation $\tilde{\varrho}$ of $\mathcal{U}$ be as in Lemma 4.5. We have that $\mathcal{U}$ is Venn-good (wrt. $\tilde{\varrho}$), if and only if every $\mathcal{U}'$ is Venn-good, where $\mathcal{U}' \subseteq \mathcal{U}$ is the subcollection of attachment sets of the union of some (any) d levels of the poset R.*

Consequently, we can carry the computation only on d-tuples of levels of poset R. However, we still can not afford to check all d-tuples of levels since it would mean to process $\mathcal{O}(n^d)$ many d-tuples which is not in FPT. On the positive side, instead of checking all d-tuples of levels, we can efficiently find a meaningful d-tuple of levels at each step of subgroup computation in Babai's tower-of-groups and finish in a reasonable number of steps by Algorithm 1 given in Chapter 3. Finally, we give Algorithm 3 to solve S_d-graph isomorphism.

Altogether, we obtain the following main result:

Theorem 4.8 ([5])**.** *The S_d-graph isomorphism problem can be solved in FPT-time with respect to the fixed parameter d.*

Proof. We apply Procedure 4.4. This actually runs $\mathcal{O}(n)$ iterations of choices of C and D; we first greedily find any valid central clique C of an S_d-representation of G, and then iterate all maximal cliques $D \subseteq H$. In each iteration, we routinely compute in polynomial time the sets of components $\mathcal{X}$ and $\mathcal{Y}$, and the posets P and Q on them and $R = P \uplus Q$. If any of P, Q has width greater than d, then we reject this iteration. Similarly, we reject the iteration if any of the graphs $K(Z)$ for $Z \in \mathcal{X} \cup \mathcal{Y}$ is not interval [17]. Otherwise, we compute colors on R such that $Z, Z' \in \mathcal{X} \cup \mathcal{Y}$ on the same level of R receive the same color iff $K(Z) \simeq K(Z')$, using the isomorphism algorithm of [17]. This finishes step 1.

Step 2 – computing the automorphism group Γ of R, is done by Theorem 2.2.

Step 3 – finding the subgroup $\Gamma' \subseteq \Gamma$, is accomplished by an iterated application of Theorem 2.1; the refinement steps are defined by d-tuples of levels of R according to Corollary 4.7 and the above outlined procedure for

finding them, and there are $\mathcal{O}(nd \log d)$ such steps. The membership test in Theorem 2.1 is provided by Lemma 4.6.

Finally, we straightforwardly check in step 4 on the generators of Γ' whether some of them maps an element of P to an element of Q.

If any iteration of Procedure 4.4 succeeds in step 4, then, by Lemma 4.5, there exists an automorphism of the graph $K = G \uplus H$ which moreover swaps G and H. Then $G \simeq H$. Conversely, assume $G \simeq H$. Since G is an S_d-graph, we find a maximal central clique $C \subseteq G$, and since all maximal cliques $D \subseteq H$ are tried, we get into an iteration with D being the isomorphic image of C. Then the posets P and Q (wrt. C, D) must be isomorphic respecting their colors, which follows from $G \simeq H$. Therefore, there exists an automorphism $\varrho \in \Gamma'$, and hence also a generator of Γ', swapping P and Q (and preserving the computed cardinality Venn diagram by Lemma 4.5). Some iteration hence succeeds and returns that $G \simeq H$.

We now briefly state the runtime since [34] which we use does not state an explicit runtime. We remark that Procedure 4.4 loops $\mathcal{O}(n)$ times with suitable C and different choices of D, analogously to the procedure of Theorem 4.2, and this initial setup of the procedure altogether takes time $\mathcal{O}(d^2 n^3)$. Then we have to account for $\mathcal{O}(n)$ calls to steps 2 and 3 of Procedure 4.4, that is, $\mathcal{O}(n)$ computations of the subgroups Γ and Γ'. Each time this part is dominated by step 3 which performs $\mathcal{O}(nd \log d)$ calls to the algorithm of Theorem 2.1 [34] in order to compute Γ' from Γ. Reading the fine details of [34], and adjusting it (the "sift table") to our setting, leads to an estimate of $\mathcal{O}(n^3) \cdot d!^{\mathcal{O}(d)}$ for each of these calls. After summarizing, we get the total estimate of $\mathcal{O}(n^5) \cdot d!^{\mathcal{O}(d)}$. $\square$

Algorithm 3 Isomorphism Test for general S_d-graphs (G,H)

Require: Given two S_d-graphs G and H (their representations not required), and the parameter d of S_d.

Ensure: Result of the isomorphism test between G and H.

Find the maximal clique collections $\mathcal{S}$ and $\mathcal{T}$ of S_d-graphs G and H;
if $|\mathcal{S}| \neq |\mathcal{T}|$, or the cardinalities of members of $\mathcal{S}$ and $\mathcal{T}$ do not match **then**
    return "G and H are not isomorphic";
$K \leftarrow G \uplus H$ (disjoint union);
repeat for each maximal clique $C \in \mathcal{S}$:
    Find the connected components $X_1, X_2, \ldots, X_k$ of $G - C$, assuming the equivalent connected components are joined into single bridge(s) of C;
    $P \leftarrow$ the partial order on $X_1, X_2, \ldots, X_k$ of $G - C$ determined by their attachments;
until all $G[C \cup X_i]$, $1 \leq i \leq k$, are interval graphs, and P is of width $\leq d$;
for each $D \in \mathcal{T}$ with $|C| = |D|$ **do**
    Find the connected components $Y_1, Y_2, \ldots, Y_l$ of $H - D$, assuming the equivalent connected components are joined into single bridge(s) of D;
    $Q \leftarrow$ the partial order on $Y_1, Y_2, \ldots, Y_l$ of $H - D$;
    if $k = l$, all $H[D \cup Y_i]$, $1 \leq i \leq k$, are interval graphs, and Q is of width $\leq d$ **then**
        $R \leftarrow P \uplus Q$ (disjoint union);
        Determine the levels $L_1, L_2, \ldots, L_k$ of the poset R;
        for $i := 1, 2, \ldots, k$ **do**
            for each pair of components $Z, Z' \in L_i$ **do**
                Compare the interval graphs $K(Z)$ and $K(Z')$ to isomorphism;
            Having computed in the previous step the isomorphism equivalence classes $L_i^1, \ldots, L_i^{c_i}$ of L_i, give the element $Z \in L_i$ of poset R color (i, j) iff $Z \in L_i^j$;
        $\Gamma \leftarrow$ the color-preserving automorphism group of R, computed by Theorem 2.2;
        $\Gamma' \leftarrow$ the subgroup of Γ consisting of those automorphisms ϱ of R which pass the test of Lemma 4.5 using **Algorithm 1**;
        if P and Q are swapped in some of the generators of Γ' **then**
            return "G and H are isomorphic";
return "G and H are not isomorphic";

Chapter 5

Proper S_d-graph Isomorphism and Proper T-graph Isomorphism

In this chapter, we consider the isomorphism problem for proper S_d- and proper T-graphs from the parameterized complexity point of view. As noted in the introduction, proper T-graphs present a significantly more restrictive graph class than general T-graphs. Recently, Chaplick et al. [21] have shown that proper T-graphs can be recognized in `FPT`-time parameterized by the size of T (which is not yet known for general T-graphs). Based on our paper [10], we show that also the isomorphism problem of proper T-graphs is notably easier than that of S_d-graphs given in Chapter 4 and T-graphs given in Chapter 6. Though, the isomorphism problem stays `GI`-complete for proper S_d-graphs with d on the input – recall Proposition 4.1, and so we cannot realistically hope for polynomial time algorithms here. We also give a simple `XP`-time isomorphism testing for T-graphs here as an extension of our `FPT`-time S_d-graph isomorphism testing, again based on our paper [10].

5.1 Isomorphism of proper S_d-graphs

In the case of $T = S_d$, the isomorphism algorithm is really simple and purely combinatorial:

Theorem 5.1. *The isomorphism of proper S_d-graphs can be tested in `FPT`-time parameterized by d.*

Proof. This is based on the following observation:

If $C \subseteq G$ is the central clique of a proper S_d-representation of G and $X_1, \ldots, X_k$ are the connected components of $G - C$ which have non-empty attachment in C, then $k \leq d$. Indeed, if $k > d$, then some two components, say X_i, X_j, would be represented on the same ray of a subdivision S'_d of S_d such that X_i is closer to the center than X_j. If X_j is adjacent to $v \in C$, then the representation of v in S'_d actually contains the whole representation of X_i, which is a contradiction to G being a proper S_d-graph.

Assume we get two proper S_d-graphs G and H on n vertices. Analogously to the previous algorithms, we hence fix a central maximal clique $C \subseteq G$, determine $X_1, \ldots, X_k$ which are the connected components of $G - C$ which have nonempty attachment in C, and denote by X_0 the possible connected components of G disjoint from C (or set $X_0 = \emptyset$). We loop through all maximal cliques $D \subseteq H$ such that $|D| = |C|$, and similarly determine components $Y_1, \ldots, Y_l$ and Y_0 of $H - D$. If $k = l$, we compare X_0 and Y_0 to (proper) interval graph isomorphism [17]. Then we loop through all bijections $\varrho : \{1, \ldots, k\} \to \{1, \ldots, k\}$, compare the graphs $G[C \cup X_i]$ and $H[D \cup Y_{\varrho(i)}]$, $i = 1, \ldots, k$, to interval graph isomorphism, and compare their attachment collections in C and in D using the test provided by Lemmas 4.5 and 4.6. (Recall that the attachment collections are automorphism-invariant.) If we ever succeed, then we mark G and H as isomorphic.

The overall approach takes $\mathcal{O}(k!n^3) \leq \mathcal{O}(d!\, n^3)$ time which is in FPT with respect to d. The correctness follows by the same arguments given in Chapter 4. □

5.2 Isomorphism of proper T-graphs

We now move onto proper T-graphs. As with S_d-graphs, we may always restrict to T-representations of graphs G (in a subdivision T' of T) such that the cliques represented at the branching nodes of T' are maximal cliques of G, and we call such cliques the *branching cliques* of this representation. Informally, our idea is to use the finding of Chaplick et al. [21] that those maximal cliques of a proper T-graph G which occur as the branching cliques of any T-representation of G are somehow special, and that their number, modulo an easy canonical adjustment, is bounded. We will call such cliques (after the adjustment) the *rich cliques* of G here. Then it will not be difficult to try all mappings between the rich cliques in FPT-time, and then answer the isomorphism problem similarly as in the proof of Theorem 5.1.

The full definition of the terms, following [21], is quite technical and we skip the details here since they are not important for our proof. For our purpose, it is enough to accept the notion of a *chain* in a proper T-graph G, which is (without further unnecessary details that can be found in [21]) a collection $\mathcal{Y}$ of maximal cliques of G such that $\mathcal{Y}$ is partitioned into a sequence $(C_1, \ldots, C_s)$ of the *inner* cliques and into two *terminals* which are nonempty subcollections $\mathcal{T}_1, \mathcal{T}_2 \subset \mathcal{Y}$ of (remaining) cliques. Then we use the following:

Proposition 5.2 (Chaplick et al. [21])**.** *Assume G is a connected proper T-graph. Then the following sets are unique in G (meaning that they appear the same in every proper T-representation of G); the set $\mathcal{H}(G)$ of the chains in G, and the set $\mathcal{S}(G)$ of the remaining maximal cliques of G not contained in the chains of $\mathcal{H}(G)$. Furthermore:*

a) The inner cliques of every chain of $\mathcal{H}(G)$ form a path in any proper T-representation of G,

b) if a terminal $\mathcal{T}_1$ of a chain in $\mathcal{H}(G)$ is not a singleton clique, then $\mathcal{T}_1$ is the set of the inner cliques of some (other) chain in $\mathcal{H}(G)$, and

c) if an inner clique C_i of a chain is a branching clique in some proper T-representation of G, then C_i is contained in a terminal, say $\mathcal{T}_1$, of some (other) chain in $\mathcal{H}(G)$, and any (other) clique $C_j \in \mathcal{T}_1$ may replace C_i in a proper T-representation of G.

The sets $\mathcal{H}(G)$ and $\mathcal{S}(G)$ can be computed in FPT-time parameterized by $|T|$, and their cardinalities are in $\mathcal{O}(|T|^2)$.

We immediately get:

Corollary 5.3. *If G is a connected proper T-graph, then every branching clique appearing in some proper T-representation of G is contained in the set $\mathcal{X} := \mathcal{S}(G) \cup \bigcup_{\mathcal{Y} \in \mathcal{H}(G)} \mathcal{T}(\mathcal{Y})$, where $\mathcal{T}(\mathcal{Y}) = \mathcal{T}_1 \cup \mathcal{T}_2$ for the two terminals $\mathcal{T}_1$ and $\mathcal{T}_2$ of the chain $\mathcal{Y}$.*

Note that, although the cardinality of the set $\mathcal{S}(G)$ and the number of chains in $\mathcal{H}(G)$ (and hence also the number of terminals) is bounded with respect to T by Proposition 5.2, the number of potential branching cliques in $\mathcal{X}$ is not bounded since the terminals may have arbitrary cardinality. Fortunately, Proposition 5.2(c) allows us to select just two representative cliques out of every non-singleton terminal as follows:

Lemma 5.4. *If G is a connected proper T-graph, then there is a special*

isomorphism-invariant set $\mathcal{C} \subseteq \mathcal{X}$ of maximal cliques of G such that there exists a T-representation of G in which every branching clique belongs to $\mathcal{C}$. This set $\mathcal{C}$ can be found in FPT*-time parameterized by $|T|$ and its cardinality is in $\mathcal{O}(|T|^2)$.*

Proof. We compute the set $\mathcal{C}$ of G algorithmically as follows:

1. Using the Algorithm of [21], identify the set $\mathcal{H}(G)$ of the chains in G, and $\mathcal{S}(G)$ of remaining maximal cliques of G in FPT-time parameterized by $|T|$.
2. Initially set $\mathcal{C} := \mathcal{S}(G)$.
3. For each terminal of every chain in $\mathcal{H}(G)$ which is a single clique, add it to $\mathcal{C}$.
4. For every pair of chains in $\mathcal{H}(G)$; if a non-singleton terminal of one is a set of the inner cliques of the other chain, ordered as the sequence $(C_1, \ldots, C_s)$, then add C_1 and C_s to $\mathcal{C}$.

The runtime is in FPT only because of the Algorithm of [21], while the rest of the computation of $\mathcal{C}$ is clearly in polynomial time. The cardinalities $|\mathcal{H}(G)| = \mathcal{O}(|T|^2)$ and $|\mathcal{S}(G)| = \mathcal{O}(|T|^2)$ are by Proposition 5.2, and for every chain in $\mathcal{H}(G)$, at most 4 cliques are added to $\mathcal{C}$. Finally, Proposition 5.2 also guarantees that the set $\mathcal{C}$ is invariant on the choice of a particular proper T-representation of G, hence isomorphism-invariant. □

We call the maximal cliques appearing in the isomorphism-invariant set $\mathcal{C}$ obtained by the algorithm given in the proof of Lemma 5.4 the *rich cliques* of a connected proper T-graph G, and give our fully combinatorial algorithm:

Theorem 5.5. *The isomorphism of proper T-graphs can be tested in* FPT*-time with respect to the size of T.*

Proof. We are given two proper T-graphs G and H on n vertices. Let $V_b(T)$ denote the set of branching nodes of T (which will be the set of branching nodes of any subdivision of T, too). For simplicity, we assume

- that both G and H are connected, as otherwise we can test the isomorphism between each pair of components separately, and
- that G and H are not proper T_1-graphs for any $T_1 \subsetneq T$, since we can exhaustively try the coming algorithm for all $T_1 \subsetneq T$ before trying with T.

Imagine now that $G \simeq H$. Then there exist proper T-representations of G and H in the same subdivision T' of T, such that the branching cliques of G are mapped to the branching cliques of H in a chosen isomorphism f of G and H. Let these branching cliques be $C_1, \ldots, C_m \subseteq G$ and $D_1, \ldots, D_m \subseteq H$, where $m = |V_b(T)|$ and $f(C_i) = D_i$. By Lemma 5.4, we may also assume that these cliques are among the rich cliques of G and of H. We define the graphs $G_0 := G - (C_1 \cup \ldots \cup C_m)$ and $H_0 := G - (D_1 \cup \ldots \cup D_m)$. Then, since $C_1, \ldots, C_m$ are the branching cliques in an intersection representation of G in T', we have that every connected component $G_1 \subseteq G_0$ is adjacent to at most two cliques (C_i, C_j) among $C_1, \ldots, C_m$, and we denote by $G_1^+ := G_1 \cup C_i \cup C_j$ which is always an interval graph. We do the same for H_0.

On the other hand, every branching clique C_i is adjacent to at most ℓ components of G_0, where ℓ is the number of leaves of T. If this was not true, then there would be two intervals in T' (indeed intervals since all branching cliques have been subtracted from G_0) representing two vertices of distinct components adjacent to C_i, and so one lying on a path in T' from the branching node of C_i to the other. That would violate that our representation is proper. In particular, the number of connected components of G_0 and of H_0 is in $\mathcal{O}(|T|^2)$.

Since f is an isomorphism from G to H, there is a pairing of the components of G_0 and H_0 such that for $G_1 \subseteq G_0$ and $H_1 \subseteq H_0$, we have $f(G_1) = H_1$, and this can be verified as an interval graph isomorphism. The attachment collections of the components in the branching cliques can then be compared (between G and H) in polynomial time the same way as in the proof of Theorem 5.1. Altogether, we can verify an isomorphism from G to H in polynomial time once we "guess" the right matching selections of branching cliques in G and H.

Our algorithm hence first identifies the sets of all rich cliques $\mathcal{C}$ and $\mathcal{D}$ of G and H, respectively, using Lemma 5.4. Then we loop through all assignments $f : V_b(T) \to \mathcal{C}$ and $g : V_b(T) \to \mathcal{D}$ of rich cliques to the branching nodes, and for each assignment we check whether the graph G_1^+ (resp., H_1^+) over all components of G_0 (resp., of H_0) as above is an interval graph. If this test succeeds, then we check whether there is an isomorphism from G to H respecting the assignment f and g, as described above. If no isomorphism is found in any of the rounds, then we conclude that $G \not\simeq H$.

As for the correctness, if the isomorphism test ever succeeds, then obviously $G \simeq H$. If $G \simeq H$, then, as argued above, there exist "matching" proper T-representations of G and H with their rich cliques as the branching cliques,

this assignment of the rich cliques is among the tested ones, and hence the outcome will be that $G \simeq H$. $\square$

5.3 T-graph isomorphism in XP-time

In this section, we give an easy XP-time approach to solve the isomorphism of T-graphs parameterized by the size of T before we show a more involved FPT-approach in Chapter 6.

Note that the brute-force assignments of maximal cliques to the branching vertices of T results in $n^{|T|}$ many guesses which is in XP, and for each guess, we can compare our T-graphs up to isomorphism using S_d-graph instances around branching vertices, together with some technical checks. Here, we "extract" S_d-instances recursively, resulting in a more efficient XP-time procedure instead of checking all assignments by brute-force.

Procedure 5.6. *Given two T-graphs G and H, we apply the following approach for each pair of suitable maximal cliques C of G and D of H:*

1. *Identify all interval components $\mathcal{X}$ of $G - C$ and $\mathcal{Y}$ of $H - D$. If $|\mathcal{X}| = |\mathcal{Y}|$, then let $A_1, \ldots, A_a$ and and $B_1, \ldots, B_b$ be the non-interval components of $G - C$ and $H - D$, respectively. If $a = b$:*
   (a) *Construct the posets P and Q formed by the interval components $\mathcal{X}$ and $\mathcal{Y}$, respectively.*
   (b) *Compute the automorphism group Π of $R = P \uplus Q$.*
   (c) *Loop through all matchings between $\{A_1, \ldots, A_a\}$ and $\{B_1, \ldots, B_a\}$, and for each pair (A_i, B_j), compute* "recursively" *the automorphism group of $(A_i \uplus B_j)$ getting the automorphism groups $\Pi_1, \ldots, \Pi_a$ at the end of recursion.*
2. *Take the disjoint product group $\Pi_+ := \Pi \times \Pi_1 \times \cdots \times \Pi_a$, and refine in to a subgroup Π' respecting possible bijections between C and D with respect to whole $G \uplus H$. This can be done analogously to Section 4.3.*
3. *If P and Q are swapped in some automorphism in Π' from step 2, then return that G and H are isomorphic.*

If the procedure does not return that G and H are isomorphic for some pair (C, D), then output that G and H are not isomorphic.

Theorem 5.7. *Procedure 5.6 correctly tests the isomorphism of T-graphs in* XP*-time parameterized by $|T|$.*

Proof. We first prove the correctness of Procedure 5.6.

By a suitable pair of maximal cliques, we mean a pair of cliques of the same size. Our aim is to remove the rays of S_d-graph instances recursively while remembering the automorphisms of each pair of removed interval graphs in each recursion for some pair of initially matched maximal cliques. We first explain the non-recursive part of this procedure.

The core of the recursive part of our procedure is regarding to the connected components which are not interval graphs, i.e., the connected components obtained by the removal of interval subgraphs of G which are T-graphs.

While computing the automorphisms of a disconnected graph, direct product of connected components can be used only when its connected components are pairwise non-isomorphic, and otherwise, semidirect product is needed. However, even if we test the isomorphism on the disjoint union of our posets, we work on disjoint ground sets here, thus we can still consider the direct product of our computed subgroups since they commute and the generators of $\Pi_+ := \Pi \times \Pi_1 \times \cdots \times \Pi_a$ is the union of generators of single groups.

Therefore, each automorphism ϱ of Π_+ induces a permutation on the Venn diagram of $C \uplus D$ (even though we do not use S_d-graph isomorphism and Venn diagrams consistency checks here), which is composed of the neighborhoods of all interval components and of each pair (A_i, B_j), i.e., independent contributions from each subgroup of Π_+. On the other hand, we still have $\leq |T|$ (= leafage of T) antichains.

We can still apply Babai's tower-of-groups approach which we used in [5] to compute the subgroup $\Pi' \subseteq \Pi_+$ which is consistent on Venn diagrams of $C \uplus D$ which we do not consider up to this point, and after that we obtain the full automorphism group of $G \uplus H$.

Regarding to the recursion in this procedure, we first explain how the recursion is performed. We use the breadth-first search manner, thus our procedure exhausts all non-interval components which means that all central cliques (maximal cliques which can be placed on branching vertices of T) are identified. Therefore, it can finish all calls till the same level of recursion.

We now prove the complexity of Procedure 5.6.

We try all suitable pairs of maximal clique C and D, thus at most $\mathcal{O}(n^2)$ pairs. For each such pair, we have at most $|T|$ many non-interval components, thus the complexity of each recursion is bounded by a function of the parameter size $|T|$ and a polynomial in n, and the corresponding tower-of-groups can be computed in `FPT`-time. Moreover, the depth of recursions can be bounded by $r \leq diam(T) + log_2|T|$ where $diam(T)$ is the diameter of T. Therefore, the overall complexity is $f(|T|) \cdot \mathcal{O}(n^r)$ which is in `XP`. $\square$

Chapter 6

T-graph Isomorphism

In this chapter, we solve the isomorphism problem for T-graphs in FPT-time based on our conference paper [11], and its full preprint [2]. We achieve that by introducing a canonical decomposition method for chordal graphs into levels containing interval subgraphs of the given T-graphs, and applying our FPT-procedure to compute the automorphism group of the marked cliques of an interval graph in Section 3.2. We also note that our FPT-approach for T-graph isomorphism extends to an FPT-approach for the isomorphism problem of chordal graphs of bounded leafage. We also show that isomorphism problem of graphs of bounded vertex leafage ℓ is GI-complete even for the case $\ell = 2$ which is a joint work with Pascal Schweitzer.

6.1 Structure of chordal graphs

We now give several useful terms and facts related to chordal graphs. A vertex v of a graph G is called *simplicial* if its neighborhood corresponds to a clique of G. It is known that every chordal graph contains a simplicial vertex and, by removing the simplicial vertices of a chordal graph repeatedly, one obtains an empty graph.

A *weighted clique graph* $\mathcal{C}_G$ of a graph G is the graph whose vertices are the maximal cliques of G and there is an edge between two vertices in $\mathcal{C}_G$ whenever the corresponding maximal cliques have a non-empty intersection. The edges in $\mathcal{C}_G$ are weighted by the cardinality of the intersection of the corresponding cliques.

A *clique tree* of G is any maximum-weight spanning tree of $\mathcal{C}_G$ which may

not be unique. An edge of $\mathcal{C}_G$ is called *indispensable* (resp. *unnecessary*) if it appears in every (resp. none) maximum-weight spanning tree of $\mathcal{C}_G$. All other edges are called *dispensable*. If G is chordal, every maximum-weight spanning tree T of $\mathcal{C}_G$ is also a T-representation of G, e.g. [53]. Therefore, the clique tree of a chordal graph may not be unique. We also note that the clique graph of a chordal graph G may not be chordal.

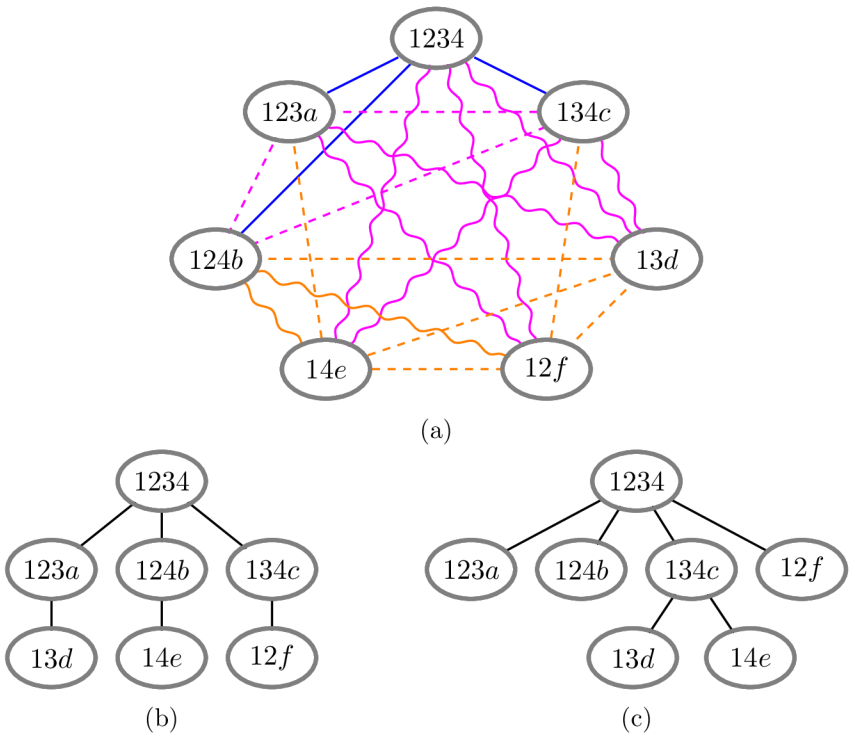


Figure 6.1

In Figure 6.1a, we see the weighted clique graph $\mathcal{C}_G$ of a chordal graph G where the nodes correspond to the maximal cliques of G, the node labels correspond to the vertices forming each maximal clique, the colors of the edges illustrate the intersection size of the corresponding maximal cliques, precisely the colors blue, magenta and orange correspond to intersection size three, two and one, respectively, and the shapes of the edges illustrate the edge types, precisely the shapes straight, curly and dashed correspond to indispensable, dispensable and unnecessary edges, respectively. In Figure 6.1b and Figure 6.1c, we see two distinct clique trees of G which are the maximum-weight spanning trees of $\mathcal{C}_G$.

For a graph G and two vertices $u \neq v \in V(G)$, a subset $S \subseteq V(G)$ is called a *u-v separator* (or *u-v cut*) of G if u and v belong to different components of $G - S$. When $|S| = 1$, then S is called a *cutvertex*. S is called *minimal* if no proper subset of S is a u-v separator. Minimal separators of a graph are the separators which are minimal for some pair of vertices. Chordal graphs, thus T-graphs, have linearly many minimal vertex separators [38].

A *leaf clique* of a T-graph G is a maximal clique of G which can be a leaf of some clique tree of G (informally, it can be placed on a leaf of T in some T-representation of G). We use the following lemma in our algorithm:

Lemma 6.1 (Matsui et al. [53])**.** *A maximal clique C of a chordal graph G can be a leaf of a clique tree if and only if C satisfies (1) C is incident to at most one indispensable edge of $\mathcal{C}_G$, and (2) C is not a cutvertex in $\mathcal{C}'_G$ which is the subgraph of $\mathcal{C}_G$ which includes all edges except the unnecessary ones. The conditions can be checked in polynomial time.*

6.2 Canonical decomposition of T-graphs

The overall goal now is to recursively find a unique decomposition of a given T-graph G into levels such that each level consists of a bounded number of interval fragments. In mathematics, a *canonical form* of an object means a simple mathematical expression specifying that object which is identified in a unique way. Here, we introduce a *canonical decomposition method* for T-graphs which means that such decomposition is invariant under automorphisms of a T-graph, and thus, for isomorphic T-graphs.

For an illustration, a similar decomposition can be obtained directly from a T-representation of G: pick the interval subgraphs of G which are represented exclusively on the leaf edges of T, forming the outermost level, and recursively in the same way obtain the next levels. Unfortunately, this is not a suitable solution for us, not only that we do not have a T-decomposition at hand, but mainly because we need our decomposition to be *canonical* while this depends on a particular representation.

The contribution of this section is to compute such a decomposition the right canonical way. As sketched above, the core task is to canonically determine in the given graph G one bounded-size collection of fragments which will form the outermost level of the decomposition, and then the rest of the decomposition is obtained in the same way from recursively computed

collections of fragments in the rest of the graph, which is also a T-graph[1].

For a chordal graph G and a (fixed) collection $Z_1, Z_2, \ldots, Z_s \subseteq G$ of distinct cliques, we write $Z_i \preceq Z_j$ if there exists $k \in \{1, \ldots, s\} \setminus \{i, j\}$ such that Z_j *separates* Z_i from Z_k in G (meaning that there is no path from $Z_i \setminus Z_j$ to $Z_k \setminus Z_j$ in $G - Z_j$), and say that $Z_i \preceq Z_j$ is witnessed by Z_k. Note that $\preceq$ is transitive, and hence a preorder. Let $Z_i \precneqq Z_j$ mean that $Z_i \preceq Z_j$ but $Z_j \not\preceq Z_i$. We also write $Z_i \approx Z_j$ if there exists $k \in \{1, \ldots, s\} \setminus \{i, j\}$ such that both $Z_i \preceq Z_j$ and $Z_j \preceq Z_i$ hold and are witnessed by Z_k. Note that $Z_j \approx Z_i$ is stronger than just saying '$Z_i \preceq Z_j$ and $Z_j \preceq Z_i$,' and that $Z_i \cap Z_j$ then separates $Z_i \Delta Z_j$ from Z_k.

Lemma 6.2. *Let T be a tree with d leaves, and G be a T-graph. Assume that $Z_1, \ldots, Z_s \subseteq G$ are distinct cliques of G such that one of the following holds:*
a) for any $1 \leq i \neq j \leq s$, neither $Z_i \precneqq Z_j$ nor $Z_i \approx Z_j$ is true, or
b) for each $1 \leq i \leq s$, the set Z_i is a minimal separator in G cutting off a component F of $G - Z_i$ such that F contains a simplicial vertex of a leaf clique of G, and that F is disjoint from all Z_j, $j \neq i$.
Then $s \leq d$.

Proof. Consider a T-representation of the graph G – as the intersection graph of subtrees of a subdivision T' of T. Then every clique X of G must be represented such that the representatives of all vertices of X intersect in (at least) one common node $v[X] \in V(T')$.

a) If $s > d$, then by (*) there exist $1 \leq i < j < k \leq s$ such that the three (distinct) nodes $v[Z_i]$, $v[Z_j]$ and $v[Z_k]$ of T' lie on one path in this order. Hence Z_j separates Z_i from Z_k in G by basic properties of a T-representation. If Z_i separates Z_J from Z_k, too, then $Z_i \approx Z_j$. Otherwise, we have $Z_i \precneqq Z_j$, and both cases lead to a contradiction.

b) For a component F of $G - Z_i$, let X_i be a leaf clique of G having a simplicial vertex in F. As in a), assuming $s > d$, we find nodes $v[X_i]$, $v[X_j]$ and $v[X_k]$ of T' (of three distinct cliques X_i, X_j, X_k of G) that lie on one path in this order. However, from the definition of a T-representation

[1]Since the requirement of canonicity of our collection does not allow us to relate this collection to a particular T-representation of G, we cannot say whether the rest of G (after removing our collection of fragments) would be a T_1-graph for some strict subtree $T_1 \subsetneq T$, or only a T-graph again. That is why we speak about T-graphs for the same T (or, we could say graphs of bounded leafage here) throughout the whole recursion. In particular, we cannot directly use this procedure to recognize T-graphs.

(note that the node $v[X_j]$ disconnects $v[X_i]$ from $v[X_k]$ in T'), we get that $Z_i \cap Z_k \subseteq Z_j$. From the assumption of minimality of our separators we conclude that $Z_i = Z_j = Z_k$. $\square$

If G is a chordal graph, $Z \subseteq G$ a minimal separator in G and $F \subseteq G$ a connected component of $G - Z$. Note that Z is a clique since G is chordal, and that whole Z is in the neighborhood of F by minimality (hence, F determines Z in G). We call a *completion of* F (in implicit G) the graph F^+ obtained by contracting all vertices of G not in $V(F) \cup Z$ into one vertex l (the neighborhood of l is thus Z) and joining l with a new leaf vertex l', called the *tail of* F^+.

We call a collection of disjoint nonempty induced subgraphs (not necessarily connected) $X_1, X_2, \ldots, X_s \subseteq G$, such that there are no edges between distinct X_i and X_j, a *fragment collection* of G (of size s). Note that the completion X_i^+ in chordal G is well-defined. We first describe our procedure for computing a suitable fragment collection, and subsequently establish the crucial properties of the computed collection and the whole decomposition.

Procedure 6.3. Let T be a tree with d leaves and no degree-2 vertex. Assume a T-graph G on the input. We compute an induced (and canonical) fragment collection $X_1, X_2, \ldots, X_s \subseteq G$ of G of size $0 < s \leq 2d$ as follows:

1. List all maximal cliques in G (using a simplicial decomposition) and compute the weighted clique graph $\mathcal{C}_G$ of G. Compute the list $\mathcal{L}$ of all possible leaf cliques of G by Lemma 6.1; in more detail, using [53, Algorithm 2] for computation of the indispensable edges in $\mathcal{C}_G$.

2. For every pair $L_1, L_2 \in \mathcal{L}$ such that $L_1 \precneqq L_2$, remove L_2 from the list. Let $\mathcal{L}_0 \subseteq \mathcal{L}$ be the resulting list of cliques, which is nonempty since $\precneqq$ is acyclic.

3. Let $\mathcal{L}_1 := \{L \in \mathcal{L}_0 : \forall L' \in \mathcal{L}_0 \setminus \{L\}.\ L \not\approx L'\}$ be the subcollection of cliques incomparable with others in $\approx$. By Lemma 6.2(a) we have $|\mathcal{L}_1| \leq d$. If $\mathcal{L}_1 \neq \emptyset$, then **output** the following fragment collection of G: for each $L \in \mathcal{L}_1$, include in it the set $F \subseteq L$ of all simplicial vertices of L in the graph G.

4. Now, for each $L \in \mathcal{L}_0$ we have $L' \in \mathcal{L}_0 \setminus \{L\}$ such that $L \approx L'$ (and so $L \cap L'$ is a separator in G). For distinct $L_1, L_2 \in \mathcal{L}_0$ such that $L_1 \approx L_2$, we call a set $Z \subseteq L_1 \cap L_2$ a *joint separator for* L_1, L_2 if Z separates $L_1 \Delta L_2$ from $L \setminus Z$ for some (any) $L \in \mathcal{L}_0 \setminus \{L_1, L_2\}$. We compute the

family $\mathcal{Z}$ of all inclusion-minimal sets Z which are joint separators for some pair $L_1 \approx L_2 \in \mathcal{L}_0$ as above, over all such pairs L_1, L_2. This is efficient since all minimal separators in chordal graphs can be listed in linear time. Note that no set $Z \in \mathcal{Z}$ contains any simplicial vertex of G, and so $V(G) \not\subseteq \bigcup \mathcal{Z}$.

5. Let $\mathcal{C}$ be the family of the connected components of $G - \bigcup \mathcal{Z}$, and $\mathcal{C}_0 \subseteq \mathcal{C}$ consist of such $F \in \mathcal{C}$ that F is incident to just one set $Z_F \in \mathcal{Z}$. Note that $\mathcal{C}_0 \neq \emptyset$, since otherwise the incidence graph between $\mathcal{C}$ and $\mathcal{Z}$ would have a cycle and this would in turn contradict chordality of G. Let $\mathcal{Z}_0 := \{Z_F \in \mathcal{Z} : F \in \mathcal{C}_0\}$. Moreover, by Lemma 6.2(b), $|\mathcal{Z}_0| \leq d$.

6. We make a collection $\mathcal{C}_0'$ from $\mathcal{C}_0$ by the following operation: for each $Z \in \mathcal{Z}_0$, take all $F \in \mathcal{C}_0$ such that $Z_F = Z$ and every vertex of F is adjacent to whole Z, and join them into one graph in $\mathcal{C}_0'$ (note that there can be arbitrarily many such F for one Z). Remaining graphs of $\mathcal{C}_0$ stay in $\mathcal{C}_0'$ without change. Then, we denote by $\mathcal{C}_1 \subseteq \mathcal{C}_0'$ the subcollection of those $F \in \mathcal{C}_0'$ such that the completion F^+ of F (in G) is an interval graph.[2]

7. If $\mathcal{C}_1 \neq \emptyset$, then **output** $\mathcal{C}_1$ as the fragment collection. (As we can show from Lemma 6.2(b), $|\mathcal{C}_1| \leq d + |\mathcal{Z}_0| \leq 2d$.)

8. Otherwise, for each graph $F \in \mathcal{C}_0'$, we call this procedure recursively on the completion F^+ of F (these calls are independent since the graphs in $\mathcal{C}_0'$ are pairwise disjoint). Among the fragments returned by this call, we keep only those which are subgraphs of F.[3] We **output** the fragment collection formed by the union of kept fragments from all recursive calls.

One call to Procedure 6.3 clearly takes only polynomial time (in some steps this depends on G being chordal – e.g., listing all cliques or separators). Since the possible recursive calls in the procedure are applied to pairwise disjoint parts of the graph (except the negligible completion of F to F^+), the overall computation of Procedure 6.3 takes polynomial time regardless of d. Regarding correctness, we must prove that $s \leq 2d$, which follows from stated Lemma 6.2 except in the last (recursive) step where it can be derived

[2]Informally, $F^+ \in \mathcal{C}_1$ iff F has an interval representation (on a horizontal line) to which its separator Z_F can be "attached from the left" on the same line.

[3]Note that, e.g., the separator and tail of F^+ may also be involved in a recursively computed fragment.

in a way similar (albeit more complicated) to Lemma 6.2.

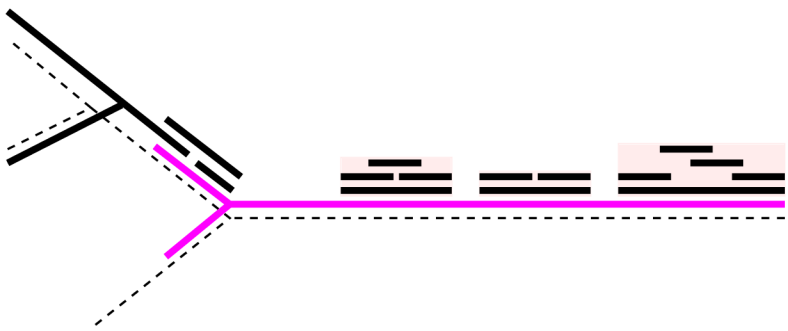


Figure 6.2

The last part is to prove a crucial fact that the collection $X_1, X_2, \ldots, X_s \subseteq G$ is indeed canonical, which is precisely stated as follows:

Lemma 6.4. *Let G and G' be isomorphic T-graphs. If Procedure 6.3 computes the canonical collection $X_1, \ldots, X_s$ for G and the canonical collection $X'_1, \ldots, X'_{s'}$ for G', then $s = s'$ and there is an isomorphism between G and G' matching in some order $X_1, \ldots, X_s$ to $X'_1, \ldots, X'_s$.*

Proof. One may easily verify that every step of Procedure 6.3 takes into account only isomorphism-invariant properties of the graph G, does not consider the input representation of G in any way and makes no arbitrary decisions. Consequently, every step performed by the procedure for the input G has an "isomorphic" step preformed for the input G'. This extends to possible recursive calls as well, and the conclusion follows. □

The following observation will be useful in the coming proofs.

(*) If T is a tree with d leaves, and we mark $d+1$ vertices of T, then some path in T contains 3 marked vertices.

This can be proved by successively removing unmarked leaves from T, until all $\leq d$ leaves of the remaining subtree of T are marked. However, then also some internal vertex is marked, and it can be prolonged into a path ending in two leaves which are marked, too.

Levels, attachments and terminal sets. Following Procedure 6.3, we now show how the full decomposition of a T-graph G is completed.

For every fragment X of the canonical collection computed by Procedure 6.3, we define the list of *attachment sets* of X in $G - X$ as follows. If $X = F$ is obtained in step 3, then it has one attachment set $L \setminus F$. Otherwise (steps 6 and 7), the attachment sets of $X = F$ are all subsets A of the corresponding separator Z (of F) such that some vertex of X has the neighborhood in Z equal to A. Observe that the attachment sets of X are always cliques contained in the completion X^+, as defined above. Moreover, it is important that the attachment sets of X form a chain by the set inclusion, since G is chordal, and hence they are *uniquely determined* independently of automorphisms of X^+.

Procedure 6.5. Given a T-graph G, we determine a **canonical decomposition** of G recursively as follows. Start with $i = 1$ and $G_0 := G$.

1. Run Procedure 6.3 for G_{i-1}, obtaining the collection $X_1, \ldots, X_s$.
2. We call the special interval subgraphs $X_1, \ldots, X_s$ *fragments* and their family $\mathcal{X}_i := \{X_1, \ldots, X_s\}$ a *level* (of number i) of the constructed decomposition.
3. Let $G_i := G - \big(V(X_1) \cup \ldots \cup V(X_s)\big)$. Mark every attachment set of each X_j in G_i as a *terminal set*. These terminal sets will be further refined when recursively decomposing G_i; namely, further constructed fragments of G_i will inherit induced subsets of marked terminal sets as their terminal sets.
4. As long as G_i is not an interval graph, repeat this from step 1 with $i \leftarrow i + 1$.

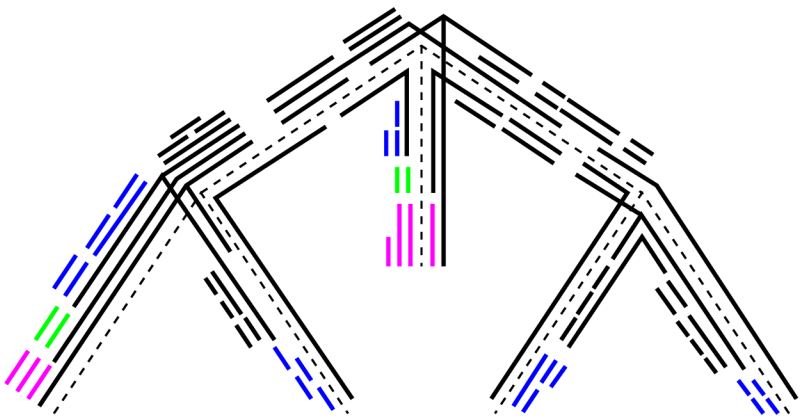


Figure 6.3

Regarding this procedure, we stress that the obtained levels are numbered "from outside", meaning that the first (outermost) level is of the least index. The rule is that fragments from lower levels have their attachment sets as terminal sets in higher levels. As it will be made precise in the next section, an isomorphism between two T-graphs can be captured by a mapping between their canonical decompositions, which relates pairwise isomorphic fragments and preserves the incidence (i.e., identity) between the attachment sets of mapped fragments and the terminal sets of fragments in higher levels. See Figure 6.3 where the magenta, green and blue fragments are obtained as the first, second and third level, respectively, and the black fragment is the last fragment which forms an induced interval graph.

6.3 Isomorphism of T-graph decompositions

There are two related classical algebraic tools which are introduced in Section 2.2 and used in this section. The first one is the algorithm, called the "sift table", performing the computation of a subgroup of an arbitrary group, provided that we can efficiently test the membership in the subgroup and the subgroup is not "much smaller" than the original group given as Theorem 2.1. The second tool is Babai's "tower-of-groups" procedure (cf. [14]) to compute an inclusion-ordered chain of subgroups of a graph where the final subgroup is the automorphism group of our graph.

Here we are going to apply Babai's "tower-of-groups" procedure and Theorem 2.1 given in Section 2.2, and Theorem 3.8 given in Section 3.2 in order to compute the automorphism group of a special object which combines the decompositions (Procedure 6.5) of given T-graphs G_1 and G_2, but abstracts from precise structure of the fragments as graphs.

Consider canonical decompositions of the graphs G_1 and G_2, as produced by Procedure 6.5 in the form of level families $\mathcal{X}_1^1, \ldots, \mathcal{X}_\ell^1$ and $\mathcal{X}_1^2, \ldots, \mathcal{X}_{\ell'}^2$, respectively. We may assume that $\ell = \ell'$ since otherwise we immediately answer 'not isomorphic'. A combined decomposition of $H = G_1 \uplus G_2$ hence consists of the levels $\mathcal{X}_i := \mathcal{X}_i^1 \cup \mathcal{X}_i^2$ for $i = 1, \ldots, \ell$ and their respective terminal sets. More precisely, let $\mathcal{X} := \mathcal{X}_1 \cup \ldots \cup \mathcal{X}_\ell$. Let $\mathcal{A}[X]$ for $X \in \mathcal{X}_k$ be the family of all terminal sets in X (as marked by Procedure 6.5 and then restricted to $V(X)$), and specially $\mathcal{A}^i[X] \subseteq \mathcal{A}[X]$ be those terminal sets in X which come from attachment sets of fragments on level $i < k$. Let $\mathcal{A}_k := \bigcup_{X \in \mathcal{X}_k} \mathcal{A}[X]$ and $\mathcal{A}_k^i := \bigcup_{X \in \mathcal{X}_k} \mathcal{A}^i[X]$ for $k = 1, \ldots, \ell$, and let $\mathcal{A} := \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_\ell$.

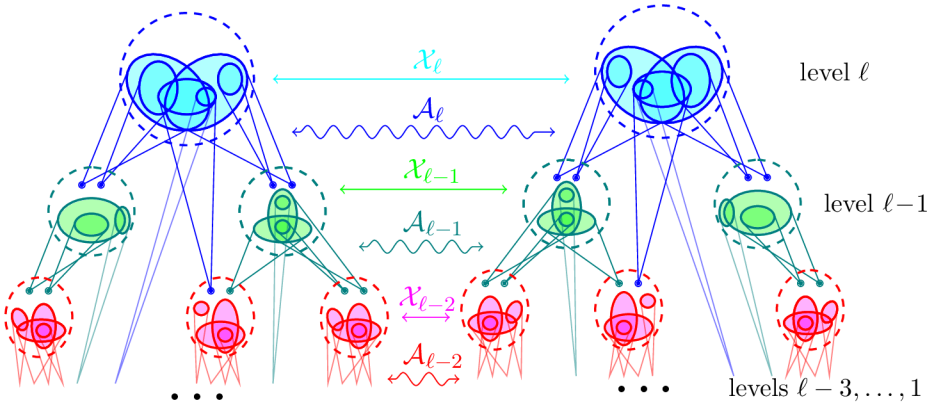


Figure 6.4

Figure 6.4 is an illustration of a (combined) canonical decomposition of the graph $H = G_1 \uplus G_2$ into ℓ levels, with the collections of fragments $\mathcal{X}$ (thick black circles) and of terminal sets $\mathcal{A}$ (colored ellipses inside them). The arrows illustrate an automorphism of this decomposition: straight arrows show the possible mapping between isomorphic fragments on the same level, as in 1, and wavy arrows indicate preservation of the incidence between attachment sets and the corresponding terminal sets, as stated by condition 2.

Recall the definition of the completion X^+ of any $X \in \mathcal{X}_i$ which, in the current context, is defined with respect to the subgraph of H induced on the union U of vertex sets of $\mathcal{X}_{i+1} \cup \ldots \cup \mathcal{X}_\ell$ (of the higher levels from X). This is, exactly, the completion of X defined by the call to Procedure 6.3 on the level i which defined X as a fragment. Recall also the attachment sets of X which are subsets of U (in X^+) and invariant on automorphisms of X^+.

The *automorphism group of such a decomposition of* H (Figure 6.4) acts on the ground set $\mathcal{X} \cup \mathcal{A}$, and consists of permutations ϱ of $\mathcal{X} \cup \mathcal{A}$ which, in particular, map $\mathcal{X}_i$ onto $\mathcal{X}_i$ and $\mathcal{A}_i$ onto $\mathcal{A}_i$ for all $i = 1, \ldots, \ell$. Overall, we would like the permutation ϱ correspond to an actual automorphism of the graph H, for which purpose we introduce the following definition. A permutation ϱ of $\mathcal{X} \cup \mathcal{A}$ is an *automorphism of the decomposition of* H if the following hold true;

1. for each $X \in \mathcal{X}_i$ ($i \in \{1, \ldots, \ell\}$), we have $\varrho(X) \in \mathcal{X}_i$, and there is a graph isomorphism from the completion X^+ to the completion $\varrho(X)^+$

mapping the tail of X^+ to the tail of $\varrho(X)^+$ and the terminal sets in $\mathcal{A}^j[X]$ to those in $\mathcal{A}^j[\varrho(X)]$ for each $1 \leq j < i$, and

2. for every $X \in \mathcal{X}_i$ and $A \in \mathcal{A}^i_k$ where $i \in \{1, \ldots, \ell\}$ and $k \in \{i + 1, \ldots, \ell\}$, we have that if A is an attachment set of the fragment X (to level k), then $\varrho(A)$ is the uniquely corresponding attachment set of the fragment $\varrho(X)$.

Notice the role of the last two conditions. While 1 speaks about consistency of ϱ with the actual graph H on the same level, 2 on the other hand ensures consistency "between the levels". Right from this definition we get:

Proposition 6.6. *Let $H = G_1 \uplus G_2$ and its canonical decomposition (Procedure 6.5) formed by families $\mathcal{X}$ and $\mathcal{A}$ be as above. A permutation ϱ of $\mathcal{X} \cup \mathcal{A}$ is an automorphism of this decomposition, if and only if there exists a graph automorphism of H which acts on $\mathcal{X}$ and on $\mathcal{A}$ identically to ϱ.*

Proof. In the 'only if' direction, we consider a permutation ϱ of $\mathcal{X} \cup \mathcal{A}$ satisfying conditions 1 and 2, i.e., an automorphism ϱ of the considered decomposition of H. We take the mapping π on $V(H)$ which is composed of all isomorphism bijections from $X \in \mathcal{X}$ to $\varrho(X)$ claimed by 1. Then π indeed is a permutation of $V(H)$ since ϱ is a permutation on $\mathcal{X}$, and π respects all edges of $E(H)$ which belong to some $X \in \mathcal{X}$. All remaining edges of H are between some fragment $X \in \mathcal{X}_i$ and one of its attachments ("higher up" in the decomposition) which coincides with some terminal set $A \in \mathcal{A}_j$ where $j > i$, by the way we decomposed H. Condition 2 ensures that those edges of H are preserved as well by π, and hence π is an automorphism of H. See Figure 6.4.

In the 'if' direction, by recursive application of Lemma 6.4, we get that

- any automorphism of G_1 (or of G_2, up to symmetry) preserves the fragments and the levels of the decomposition of G_1 and, consequently, it preserves also the terminal sets by their incidence with attachments of the fragments;

- if $G_1 \simeq G_2$, we have an isomorphism $\iota : G_1 \to G_2$ preserving the fragments and the levels between G_1 and G_2, and then ι can be composed with any automorphism of G_2 from the previous point.

Consequently, for every graph automorphism of H we get an induced permutation on $\mathcal{X} \cup \mathcal{A}$, which is an automorphism of the decomposition thanks to satisfying the conditions 1 and 2. □

We start with addressing the important point which has been skipped at the start of the proof of Theorem 6.10; how to efficiently handle condition 1 of an automorphism of a decomposition. That is, how to get from a well-understood automorphism group of an interval graph (cf. PQ-trees) to automorphisms acting on (and preserving) the marked terminal sets within involved interval fragments. The crucial property we use (and need) is that the maximum antichain of the terminal sets has bounded size.

Lemma 6.7. *Let F be an interval graph and $l \in V(F)$ a special (marked) degree-1 vertex l of F, called the tail of F, whose neighbor is simplicial in $F - \{l\}$. Let $\mathcal{A}^1, \ldots, \mathcal{A}^m$ be families of subsets of $V(F)$, inducing cliques in F and called the terminal sets of F, such that the maximum antichain size in $\mathcal{A}^i$ over $i = 1, \ldots, m$ equals t. Then one can in FPT-time with respect to t compute the following: for the subgroup Γ of automorphisms of F, defined below, the action of Γ on $\mathcal{A} := \mathcal{A}^1 \cup \ldots \cup \mathcal{A}^m$. The group Γ consists of those automorphisms τ of F such that τ preserves the tail of F and τ maps the set family $\mathcal{A}^i$ to $\mathcal{A}^i$ for all $i = 1, \ldots, m$.*

Proof. The proof directly follows from Theorem 3.8 given in Section 3.2. □

Using the same arguments as in Lemma 6.7, we also get:

Corollary 6.8. *Let, for $j = 1, 2$, F_j be interval graphs and $\mathcal{A}_j^1, \ldots, \mathcal{A}_j^m$ families of subsets of $V(F_j)$ and parameter t, as in Lemma 6.7 for each j. Then one can in FPT-time with respect to t decide whether there exists an isomorphism from F_1 to F_2 preserving the tails and bijectively mapping $\mathcal{A}_1^i$ to $\mathcal{A}_2^i$ for all $1 \leq i \leq m$.*

We are now ready to present our main result which gives an FPT-time algorithm for isomorphism of T-graphs (without need for a given decomposition). The algorithm is based on Proposition 6.6, and so on efficient checking of the conditions 1 and 2 in the combined decomposition of two graphs. Stated precisely:

Theorem 6.9. *For a fixed tree T, there is an FPT-time algorithm that, given graphs G_1 and G_2, correctly decides whether $G_1 \simeq G_2$, or correctly answers that one or both of G_1 and G_2 are not T-graphs.*[4]

Theorem 6.9 now follows using Procedure 6.5, basic knowledge of automorphism groups and Proposition 6.6, and the following refined statement:

[4]The latter outcome ('not a T-graph') happens when some of the assertions assuming a T-graph in Procedure 6.3 fails.

Theorem 6.10. *Assume two T-graphs G_1 and G_2, and their combined decomposition (Procedure 6.5) formed by families $\mathcal{X}$ and $\mathcal{A}$ in ℓ levels. Let $s = \max_{1 \le i \le \ell} |\mathcal{X}_i|$ be the maximum size of a level, and t be an upper bound on the maximum antichain size among the terminal set families $\mathcal{A}^i[X]$ over each $X \in \mathcal{X}$. Then the automorphism group of the decomposition, defined by 1 and 2 above, can be computed in* FPT*-time with the parameter $s + t$.*

Notice that, in our situation, the parameter $s + t$ indeed is bounded in terms of $|T|$; we have $s \le 2d$ and $t \le d$ directly from the arguments in Procedure 6.3.

Proof. First, we state that condition 1 can be dealt with (as below) efficiently w.r.t. the parameter t by Lemma 6.7. We prove the rest as a commented algorithm outline:

1. For every level $k \in \{1, \ldots, \ell\}$ of the decomposition of $H = G_1 \uplus G_2$ we compute the following permutation group Λ_k acting on $\mathcal{X}_k \cup \mathcal{A}_k$.
   (a) We partition $\mathcal{X}_k$ into classes according to the isomorphism condition 1; i.e., $X_1, X_2 \in \mathcal{X}_k$ fall into the same class iff there is a graph isomorphism from X_1^+ to X_2^+ preserving the tail and bijectively mapping $\mathcal{A}^i[X_1]$ to $\mathcal{A}^i[X_2]$ for all $1 \le i < k$. We add the bounded-order symmetric subgroup on each such class of $\mathcal{X}_k$ to Λ_k.
   (b) Now, for every permutation $\varrho \in \Lambda_k$ of $\mathcal{X}_k$ and all $X \in \mathcal{X}_k$, and for any chosen isomorphism $\iota_X : X^+ \to \varrho(X)^+$ conforming to 1, we add to Λ_k the permutation of $\mathcal{A}_k$ naturally composed of partial mappings of the terminal sets induced by the isomorphisms ι_X over $X \in \mathcal{X}_k$.
   (c) For every $X \in \mathcal{X}_k$, we compute generators of the automorphism subgroup of X^+ mapping $\mathcal{A}^i[X]$ to $\mathcal{A}^i[X]$ for all $1 \le i < k$, and we add to Λ_k the action of each such generator on $\mathcal{A}[X] \subseteq \mathcal{A}_k$ (as a new generator of Λ_k).
2. We let $\Gamma_0 = \Lambda_1 \times \ldots \times \Lambda_\ell$ be the direct product of the previous subgroups. Notice that Γ_0 is formed by the permutations conforming to condition 1.
3. Finally, we apply Babai's tower-of-groups procedure [14] to Γ_0 in order to compute the desired automorphism group of the decomposition. We loop over all pairs $1 \le i < j \le \ell$ of levels and over all cardinalities

r of terminal sets in $\mathcal{A}_j$, which is $\mathcal{O}(n^3)$ iterations, and in iteration $k = 1, 2 \ldots$ compute:

* $\Gamma_k \subseteq \Gamma_{k-1}$ consisting of exactly those automorphisms which conform to the condition 2 for every component $X \in \mathcal{X}_i$ and every terminal set $A \in \mathcal{A}_j^i$ such that $|A| = r$. Then Γ_k forms a subgroup of Γ_{k-1} (i.e., closed on a composition) thanks to the condition 1 being true in Γ_{k-1}, and so we can compute Γ_k using Theorem 2.1. This computation is efficient since $|\mathcal{X}_i| \leq s$ and the number of $A \in \mathcal{A}_j^i$ such that $|A| = r$ is clearly $\leq st$.

4. We output the last group Γ_m of step 3 as the result.

Correctness of the outcome of this algorithm is self-explanatory from the outline; Γ_m satisfies 1 and 2 for all possible choice of X and A. We finish with a brief argument of why the computation in step 3 via Theorem 2.1 is indeed efficient. By standard algebraic means (counting cosets of Γ_k in Γ_{k-1}), we get that $|\Gamma_{k-1}|/|\Gamma_k|$ is bounded from above by the order of the subgroup "induced" on $\mathcal{X}_i$ times the order of the subgroup on considered sets A of cardinality r. The latter number is at most $s! \cdot (st)!$ regardless of Γ_{k-1}. □

6.4 Isomorphism of chordal graphs parameterized by leafage and vertex leafage

In this section, we consider two parameters related to chordal graphs, namely the leafage and the vertex leafage of a chordal graph.

The *leafage* of a chordal graph G is defined as the least number of leaves of a tree T such that G is a T-graph. To conclude this chapter, we first give the following.

Lemma 6.11. *The graph isomorphism problem of chordal graphs G_1 and G_2 is in* FPT *parameterized by the leafage of G_1 and G_2.*

Proof. Let the promised leafage of G_1 and G_2 be at most d. Since there is only a bounded number of trees T of a given number of leaves, modulo the subdivisions, which depends exponentially only on d, we exhaustively try all trees $T_1, T_2, \ldots$ without degree-2 vertices and with at most d leaves. For each such T_i sequentially, we call the algorithm of Theorem 6.9 with $T = T_i$, and if we ever get an answer about $G_1 \simeq G_2$, we output it and quit.

If all answers are that G_1 or/and G_2 are not T_i-graphs, then the promise of leafage $\leq d$ is violated. $\square$

The *vertex leafage* of a chordal graph G is defined as the least number d such that there exists a tree representation of G in which every subtree has at most d leaves. We now give a polynomial reduction which is an unpublished joint work with Pascal Schweitzer proving that the isomorphism problem for graphs of vertex leafage 2 is `GI`-complete which means that we cannot realistically expect for an `XP` algorithm for bounded vertex leafage graphs in case `GI` $\neq$ P.

Theorem 6.12. *The isomorphism problem for graphs of vertex leafage* 2 *(and hence, of all bounded vertex leafage graphs except interval graphs) is* `GI`*-complete.*

Proof. Consider the class of split graphs with a split decomposition in which every vertex in the clique has at most two neighbors in the independent set. This class is graph isomorphism complete [62]. Let G_1 and G_2 be two graphs from that class on the same number n of vertices. We construct G'_i for $i = 1, 2$ as follows: subdivide every edge with a new vertex, and then make a clique on all new subdivision vertices. Then, $G_1 \simeq G_2$ if and only if $G'_1 \simeq G'_2$; this is easy since the graphs G'_1 and G'_2 constructed from G_1 and G_2, respectively, are obtained deterministically, and any isomorphism of G'_1 and G'_2 must map $V(G_1)$ to $V(G_2)$ due to vertex degrees. Since each G'_i is an S_d-graph of vertex leafage 2, with the central clique on new subdivision vertices and $d = n$, solving their isomorphism would solve also the isomorphism of G_1 and G_2. Therefore, the isomorphism problem for chordal graphs of vertex leafage 2 is already `GI`-complete. $\square$

Chapter 7

Proper U-graph Recognition and Isomorphism

In this chapter, we study the parameterized complexity of the recognition and isomorphism problems on U-graphs based on our preprint [12]. We first show that the recognition problem of $\mathcal{U}$-graphs and proper $\mathcal{U}$-graphs is NP-hard, where $\mathcal{U}$ is the class of all unicyclic graphs. We then deal with the recognition problem of proper U-graphs for a fixed unicyclic graph U and show that it can be solved in FPT-time parameterized by $|U|$. We then apply our recognition procedure to test the isomorphism of proper U-graphs in FPT-time, and we complement this result by proving that if H is not a unicyclic graph or a tree, then testing the isomorphism of H-graphs and proper H-graphs is GI-complete which is an alternative to our reduction given in Section 2.7.

Throughout this chapter, we call the unique cycle of U the *"circle"* and the vertices of U the *"nodes"* not to be confused with the cycles and the vertices of a U-graph G, respectively. We make the following two assumptions. Firstly, we assume that $U \neq K_3$ since both the recognition and isomorphism problems are linear time solvable on proper K_3-graphs [26, 27]. Secondly, we assume that $U \neq T$ for any tree T since both the recognition and the isomorphism problems are solvable in FPT-time [10, 21].

Recall that considering a particular U-representation of a U-graph G, we call a clique C of G *Helly* if it satisfies the Helly property in that representation, and otherwise, *non-Helly*. We also note here that U-graphs may have *dominating edges*, i.e. two adjacent vertices covering the circle and

having two disconnected intersections. We sometimes refer to such cliques as non-Helly for the sake of simplicity.

7.1 NP-hardness of $\mathcal{U}$-graph and proper $\mathcal{U}$-graph recognition

In this section, we show that the recognition problem is NP-hard for $\mathcal{U}$-graphs and proper $\mathcal{U}$-graphs where $\mathcal{U}$ is the class of all unicyclic graphs. A *hypergraph* is a graph where an edge called, an *hyperedge*, can join more than two vertices. If every edge of a hypergraph X joins k vertices, then X is called *k-uniform*. *Hypergraph c-coloring* is the problem to assign colors to vertices so that every hyperedge joins vertices of at least two different colors.

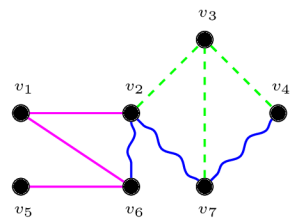


Figure 7.1: A hypergraph X on seven vertices $\{v_1, \ldots, v_7\}$ vertices and on three edges e_1, e_2 and e_3, demonstrated by straight magenta, curly blue and dashed green lines, respectively.

We define the problem HellyCliquesCARC$(G;\mathcal{C})$ as follows: Given a circular-arc graph G and a set $\mathcal{C}$ of cliques of G, decide whether there exists a circular-arc representation of G such that all cliques in $\mathcal{C}$ are Helly cliques. We first give the following.

Lemma 7.1. *The problem* HellyCliquesCARC$(G;\mathcal{C})$ *is NP-hard. Furthermore, this hardness result holds even if we restrict to circular-arc graphs that are the complements of perfect matchings for which every circular-arc representation is proper.*

Proof. The 2-coloring problem for 3-uniform hypergraphs was shown to be NP-hard [30], and we reduce it to the problem HellyCliquesCARC$(G;\mathcal{C})$. We will construct a (circular-arc) representation of a special kind of circular-arc graph G, from a 3-uniform hypergraph X. The obtained representation will be unique and computed by fixing the (circular-)ordering of the vertices

of G around the circle. The representation will provide a 2-coloring of X using the colors L and R, which will also correspond to the left and right configurations in the representation. The coloring will then be determined by the "forbidden configurations" of the vertices of G around the circle.

Let X be a 3-uniform hypergraph on n vertices, and the circular-arc graph G on $4n+4$ vertices be the complement of a perfect matching, i.e. the complete graph on $4n+4$ vertices from which a set of $2n+2$ edges forming a perfect matching is removed. For $i \in \{-1, 0, 1, \ldots, 2n\}$, let $e_i = \{v_i^1, v_i^2\}$ be an edge of this removed perfect matching, and the vertices v_i^1 and v_i^2 be represented with two non-intersecting half-arcs around a circle in G. We construct a circular-arc representation of G from X as follows for some $0^\circ < \alpha = \frac{180^\circ}{2n+2}$:

1. (v_{-1}^1, v_{-1}^2) is *the fixing pair* where v_{-1}^1 is represented with the arc starting at α and ending at $180^\circ - \alpha$, and v_{-1}^2 is represented with the arc starting at $180^\circ + \alpha$ and ending at $360^\circ - \alpha$ of the circle.
2. (v_i^1, v_i^2) for $i \in \{0, \ldots, n\}$ are *the ordering pairs* used to determine the ordered positions of the next pairs where v_i^j is represented with the arc obtained by shifting the arc representing v_{i-1}^j by α for $j \in \{1, 2\}$.
3. (v_{n+i}^1, v_{n+i}^2) for $i \in \{1, \ldots, n\}$ are *the hypergraph pairs* used to model the vertices of X.

The arc representing v_{-1}^1 is used to fix the left and right sides for the other arcs. Considering the fixed representing arcs of the ordering pairs given above, v_i^1 is said to be *to the left of* v_i^2 on v_{-1}^1 for $0 \leq i \leq n$, and v_i^1 (and v_i^2) is said to be *to the right of* v_j^1 (and v_j^2) on v_{-1}^1 for $0 \leq j < i \leq n$. We shortly denote by L and R the left and right sides on v_{-1}^1. For instance, LLR configuration for the arcs $\{v_i^1, v_j^1, v_k^2\}$ means that v_i^1 is to the left of v_j^1, v_j^1 is to the left of v_k^2 and v_k^2 is to the right of v_i^1 on v_{-1}^1, in this order.

In the constructed circular-arc representation of G, every vertex u_i of X has two representatives v_{n+i}^1 and v_{n+i}^2, thus there is a Helly clique corresponding to every triple of vertices of X since the arcs corresponding to v_{n+i}^1 and v_{n+i}^2 can freely change their positions. We now fix the arcs corresponding to following triples to be the Helly cliques $\mathcal{C}$ in this constructed representation of G:

1. $\{v_{-1}^1, v_i^1, v_{i+1}^1\}$, $\{v_{-1}^1, v_i^1, v_{i+1}^2\}$, $\{v_{-1}^1, v_i^2, v_{i+1}^2\}$ for $i \in \{0, \ldots, n\text{-}1\}$.
2. $\{v_{i\text{-}1}^2, v_i^1, v_{n+i}^1\}$, $\{v_{i\text{-}1}^2, v_i^2, v_{n+i}^1\}$ for $i \in \{1, \ldots, n\}$.
3. $\{v_{n+i}^1, v_{n+j}^1, v_{n+k}^2\}$ for every hyperedge $\{u_i, u_j, u_k\}$ for $i < j < k$.

After we fix the Helly cliques, the arcs corresponding to v_i^1 and v_i^2 can change the positions for no i since the representation is fixed. Now, on $v_{\text{-}1}^1$, we can not have LLR and RRL configurations for the arcs $\{v_{n+i}^1, v_{n+j}^1, v_{n+k}^2\}$ where $i < j < k$. Therefore, for the arcs $\{v_{n+i}^1, v_{n+j}^1, v_{n+k}^1\}$, the forbidden configurations are LLL and RRR which means that there is no monochromatic edge joining the vertices $\{u_i, u_j, u_k\}$ of X. As a result, the problem whether the fixed triples are Helly cliques in a circular-arc representation of G is NP-hard since the 2-coloring problem for 3-uniform hypergraphs is NP-hard [30].

Since the constructed circular-arc graph G is the complement of a perfect matching on $4n + 4$ vertices, and every vertex in a perfect matching has a distinct open neighborhood, every vertex in G has a distinct closed neighborhood, and in every circular-arc representation of G, no half-arc contains another. This means that G is a proper circular-arc graph, and therefore, the problem HellyCliquesCARC$(G; \mathcal{C})$ is NP-hard also for proper circular-arc graphs. □

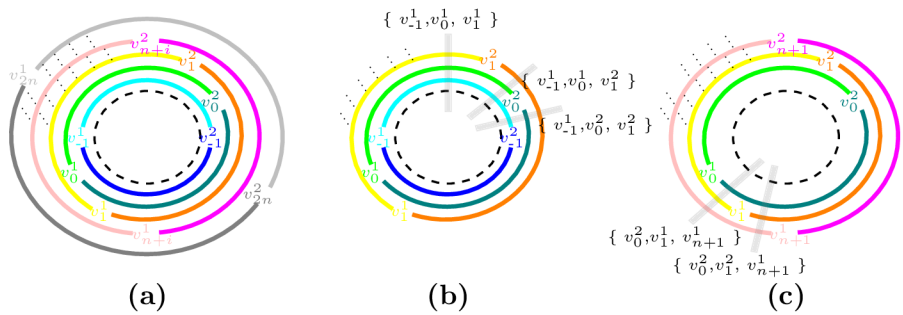


Figure 7.2: An illustration of the construction given to prove Lemma 7.1 where the colors are used solely to distinguish the arcs. (a) The circular-arc graph is the complement of the perfect matching where the dots correspond to the intermediate half-arcs, and the fixed Helly triples are (b) $\{v_{\text{-}1}^1, v_0^1, v_1^1\}$, $\{v_{\text{-}1}^1, v_0^1, v_1^2\}$ and $\{v_{\text{-}1}^1, v_0^2, v_1^2\}$, and (c) $\{v_0^2, v_1^1, v_{n+1}^1\}$ and $\{v_0^2, v_1^2, v_{n+1}^1\}$.

By Lemma 7.1, we get the following.

Corollary 7.2. *The recognition problem is NP-complete for $\mathcal{U}$-graphs and proper $\mathcal{U}$-graphs where $\mathcal{U}$ is the class of all unicyclic graphs.*

Proof. Let G be a circular-arc graph and $\mathcal{S}$ be a set of its cliques. We construct a graph G^* in polynomial time as follows: for each clique $C \in \mathcal{S}$,

add a new vertex v adjacent to all vertices in C. Then, for each $C \in \mathcal{S}$, $C \cup v$ is a maximal clique in G^*. Let $\mathcal{S}'$ denote the set of those new maximal cliques of G^*. Now, G^* is a U-graph for a unicyclic graph $U \in \mathcal{U}$ which has $|\mathcal{S}|$ branching nodes with $\mathcal{S}'$ placed on them if all cliques in $\mathcal{S}$ can be represented as Helly cliques in a circular-arc representation of G. Since the latter is NP-hard by Lemma 7.1, the recognition of $\mathcal{U}$-graphs is NP-hard. Moreover, if G is a proper circular-arc graph, then the constructed graph G^* is a proper U-graph if all cliques in $\mathcal{S}$ can be represented as Helly cliques in a circular-arc representation of G when the new vertex v added for each $C \in \mathcal{S}$ is represented on the edge not from the circle of U incident to the branching node which $C \cup v$ is placed on. Therefore, the recognition of proper $\mathcal{U}$-graphs is also NP-hard. □

7.2 Recognizing proper chordal and proper Helly U-graphs

In this section, we give two FPT-procedures to decide whether *i*) a given chordal graph G is a proper U-graph, and *ii*) a given graph G is a proper Helly U-graph. We refer to the maximal cliques of G placed on the branching nodes (i.e., the nodes of degree at least 3) of U as the *branching cliques* similar to Chapter 6. In a proper U-representation of G (if G is a proper U-graph), it is known that every branching clique can be extended to a maximal clique [22].

We first consider the case when the input graph G is chordal. Recall that chordal graphs can be recognized in linear time and they have linearly many maximal cliques which can be listed in linear time [59]. Therefore, given a graph which is chordal, we can list its set of maximal cliques efficiently, and among these maximal cliques, our aim is to identify a bounded number of maximal cliques which can exclusively be placed on the branching nodes of the circle in FPT-time parameterized by $|U|$.

Recall Lemma 5.4 from Chapter 5 which states that, given a proper T-graph, one can identify an isomorphism-invariant set of maximal cliques that can be used as branching cliques in some proper T-representations of G in FPT-time and the size of that set is bounded by $\mathcal{O}(|T|^2)$. The following lemma shows that also for proper U-graphs for a fixed unicyclic graph U, we can identify a bounded number of maximal cliques which can be used as branching cliques among polynomially many maximal cliques.

Lemma 7.3. *If a proper U-graph G has polynomially many maximal cliques, then one can, in* FPT*-time, identify an isomorphism-invariant set of maximal cliques which can be used as branching cliques placed on the circle of U in some proper U-representations of G, and the size of this isomorphism-invariant set is bounded by $\mathcal{O}(|U|^3)$.*

Proof. We can perceive a proper U-representation of G as the assignment of its maximal cliques to the nodes and edges of U where each node is a single maximal clique and each edge can have many maximal cliques on it. For every proper U-representation of G and any edge $e = \{v, w\}$ of the circle of U, let G' be the subgraph of G obtained by discarding $\mathcal{C}_e \setminus (C_v \cup C_w)$ where $\mathcal{C}_e$ denotes the set of maximal cliques placed on e, and C_v and C_w denote the maximal cliques placed on the branching nodes v and w of the circle of U, respectively. Note that every such G' is chordal and a proper T-graph where $T = U - e$ is a tree since U is unicyclic. Moreover, every G' is actually an induced subgraph of G except the case $C_v \cup C_w \supseteq C_x$ for all $C_e \in \mathcal{C}_x \setminus (C_v \cup C_w)$. Then, for each such G', one can identify the set of maximal cliques of G' which can be used as branching cliques in some proper T-representation of G' in FPT-time parameterized by $|T|$ and the size of such a set is bounded by $\mathcal{O}(|T|^2)$ by Lemma 5.4.

Therefore, one can identify a bounded number of maximal cliques of G which can be used as branching cliques in some proper U-representation of G in FPT-time parameterized by $|U|$ using the algorithm of [21], and the size of such a set is bounded by $\mathcal{O}(|U|^2 \cdot |U|) = \mathcal{O}(|U|^3)$ since the number of edges of the circle of U is at most $|U|$. The arguments on our selection being isomorphism-invariant and capturing some proper T-representations from Lemma 5.4 also hold here. □

We call the maximal cliques identified using Lemma 7.3 the *rich cliques*, and since their number in proper U-graphs is bounded by $\mathcal{O}(|U|^3)$ by Lemma 7.3, we can afford in FPT-time to try all possible assignments of rich cliques to the branching nodes on the circle of U in our recognition approach. We also emphasize that the obtained rich cliques are actually the maximal cliques which can be placed on the branching nodes of whole U. However, in our algorithm, it is enough for us to only try their assignments on the branching nodes of the circle part.

Given a graph G, we lastly make the following assumptions for the sake of simplicity before giving our recognition procedure:

A1. G is not a proper U^*-graph for some subgraph or minor U^* of U. This is due to the fact that the number of all subgraphs and minors of U is bounded by a function of $|U|$. Therefore, an FPT-algorithm that recognizes U-graphs (in this case, Procedure 7.4) can be extended to an FPT-algorithm that recognizes all subgraphs and minors of U.

A2. G is connected due to two facts. Firstly, the components of a disconnected proper U-graph without a rich clique forms an interval graph, their number may be up to $|V(G)|$ but all of them can be placed on any edge incident to a leaf of U. Secondly, the number of rich cliques, thus the components carrying a rich clique is bounded, and we can try all possible assignments of such components to the subgraphs of U.

Procedure 7.4. Given a connected chordal graph G on n vertices and a fixed unicyclic graph U, we decide whether G is a proper U-graph as follows:

1. Let $\mathcal{B}$ be the branching nodes on the circle of U and $deg(\mathcal{B})$ be the sum of degrees of nodes of U in $\mathcal{B}$.
2. Find the set $\mathcal{C}$ of rich cliques of G. If $|\mathcal{C}| < |\mathcal{B}|$, or $|\mathcal{C}|$ is not bounded by $\mathcal{O}(|U|^3)$, return that G is not a proper U-graph.
3. Looping through each assignment $f : \mathcal{C} \to \mathcal{B}$:
   (a) Let $\mathcal{C}_f$ denote the branching cliques placed on the circle of U, and $\mathcal{X}$ denote the connected components of $G - \mathcal{C}_f$.
   (b) If there are more than $deg(\mathcal{B}) - |\mathcal{B}|$ connected components in $\mathcal{X}$, move on to another assignment.
   (c) For each branching node $b_i \in \mathcal{B}$:
      i. For every $b_t \in \mathcal{B}$, let $C_t \subseteq \mathcal{C}_f$ denote the branching clique placed on b_t with respect to f.
      ii. Let P denote the maximal induced subgraph of the circle of U containing b_i and every $b_j \in \mathcal{B}$ such that $C_i \cap C_j \neq \emptyset$.
      iii. **If P does not contain at least one branching node b_j:**
         - Let $Y \subsetneq U$ denote the maximal connected subtree of U which contain only P from the circle.
         - Let $\mathcal{Y}^*$ denote the union of the branching cliques and the connected components placed on Y, and additionally the

components X_k and X_l placed on the edges next to the ends of P on the circle if they exist.

- If $G[\mathcal{Y}^*]$ is not a proper Y-graph, move on to another assignment.

iv. **Else if $\boldsymbol{P}$ is a circle and there is no connected components in $\boldsymbol{\mathcal{X}}$ placed on $\boldsymbol{P}$:**

- Let $\mathcal{D} \subseteq \mathcal{C}_f$ denote the set of branching cliques which have minimal intersections with C_i by vertex inclusion.
- If more than two branching cliques in $\mathcal{D}$ have distinct intersections with C_i, move on to another assignment.
- If no pair of branching cliques in $\mathcal{D}$ is placed on consecutive branching nodes on P, move on to another assignment.
- Let $\{b_j, b_{j+1}\} \in \mathcal{B}$ be any pair of consecutive branching nodes such that $C_l \cap C_i \subsetneq C_k \cap C_i$ for all b_k appearing on the path between b_i and b_l not passing through $b_{l'}$ where $l \in \{j, j+1\}$ and $l' = \{j, j+1\} \setminus l$.
- Let e denote the edge between b_j and b_{j+1}, $Y = U - e$ denote the maximally connected subtree of U which contains $P - e$ from the circle, and $\mathcal{Y}^* = G - (C_j \cap C_{j+1})$.
- If $G[\mathcal{Y}^*]$ is a proper Y-graph, return that G is a proper U-graph. Otherwise, move on to another assignment.

v. **Else, $\boldsymbol{P}$ is a circle and there is at least one connected component in $\boldsymbol{\mathcal{X}}$ placed on $\boldsymbol{P}$:**

- If there exist more than one such component, move on to another assignment.
- Let $X_k \in \mathcal{X}$ be that unique component placed on the edge e of P between the branching nodes b_j and b_{j+1}.
- Let $Y = U - e$ denote the maximal connected subtree of U which contain only $P - e$ from the circle.
- Let $\mathcal{Y}^*$ denote the union of the branching cliques and the connected components placed on Y, and additionally two

copies of X_k one with its attachment in C_j and the other with its attachment in C_{j+1}.

- If $G[\mathcal{Y}^*]$ is a proper Y-graph and $G[C_j \cup X_k \cup C_{j+1}]$ is a proper interval graph, return that G is a proper U-graph. Otherwise, move on to another assignment.

(d) Return that G is a proper U-graph.

4. Return that G is not a proper U-graph.

We first give the following sequence of statements which help us prove that Procedure 7.4 correctly decides whether a given chordal graph is a proper U-graph in FPT-time.

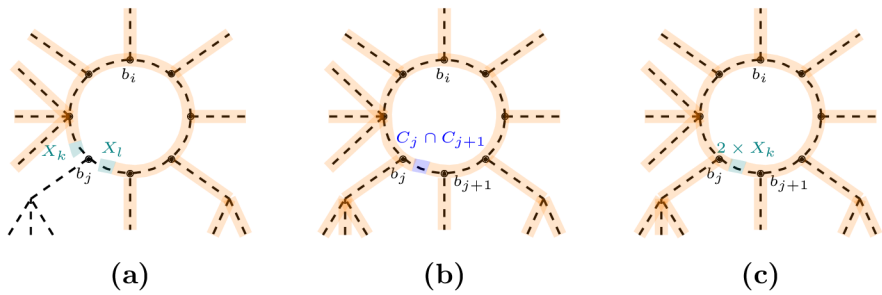


Figure 7.3: (a), (b) and (c) illustrate the Step *iii*, *iv* and *v* of Procedure 7.4, respectively, where $\mathcal{Y}^*$ contains all components and the branching cliques placed on the orange subgraph of U. In (a), $\mathcal{Y}^*$ also contains the teal components X_k and X_l. In (b), $\mathcal{Y}^*$ does not contain the blue subgraph $C_j \cap C_{j+1}$. In (c), $\mathcal{Y}^*$ also contains two copies of the teal component X_k.

Lemma 7.5. *Let G be a connected proper U-graph and $\mathcal{C}_f$ be the set of maximal cliques placed on the branching nodes on the circle of U in some proper U-representation of G. Then, $G - \mathcal{C}_f$ has at most $deg(\mathcal{B}) - |\mathcal{B}|$ connected components.*

Proof. Firstly, every edge of a proper U-representation carries at most one connected component due to the properness and the connectedness of G, and an assignment of maximal cliques to the branching nodes of U determines the components between the branching nodes if G is a proper U-graph [10], mentioned also in Chapter 5 and Section 2.8. Then, the number of connected components placed on the circle of U in $G - \mathcal{C}_f$ is at most $|\mathcal{B}|$ since there are $|\mathcal{B}|$ edges on the circle of U as there are $|\mathcal{B}|$ nodes in $\mathcal{B}$.

Similarly, the number of connected components placed not on the circle of U in $G - \mathcal{C}_f$ is at most $deg(\mathcal{B}) - 2|\mathcal{B}|$ since every node $b_i \in \mathcal{B}$ is incident to exactly two edges of the circle, i.e., for each $b_i \in \mathcal{B}$, there are $deg(b_i) - 2$ edges incident to b_i and not incident to any other $b_j \in \mathcal{B}$ where $deg(b_i)$ denotes the degree of b_i in U. Note that if a tree T is attached to a branching node $b_i \in \mathcal{B}$ via an edge e, more than one component may arise by removing the branching clique C_i on b_i. This contradicts the assumption A1 since G now forms a proper U^*-graph for the graph U^* obtained by contracting e.

Overall, there are at most $deg(\mathcal{B}) - 2|\mathcal{B}| + |\mathcal{B}| = deg(\mathcal{B}) - |\mathcal{B}|$ connected components in $G - \mathcal{C}_f$. □

Lemma 7.6. *Let the connected chordal graph G be a proper U-graph. In the setting of Procedure 7.4:*

1. *If the assignment f results in a proper U-representation of G, then every $G[\mathcal{Y}^*]$ in Step iii is a proper Y-graph for the obtained tree Y.*
2. *If there exists no $G[\mathcal{Y}^*]$ obtained from Step iv or v, then the application of Step iii for distinct branching nodes in $\mathcal{B}$ correctly tests whether G has a proper U-representation with respect to f.*

Proof. Firstly, if there is no $G[\mathcal{Y}^*]$ obtained in Step *iii*, then the lemma holds. Otherwise, if two branching cliques in $\mathcal{C}_f$ share a vertex, then there is no connected component between them in a proper T-representation since the shared vertex would contain the connected component [10], mentioned also in Chapter 5. Therefore, if P obtained from b_i excludes some branching node b_j on the circle as in Step *iii*, then $C_i \cap C_j = \emptyset$. Now, it is sufficient to check whether the components and the branching cliques contained in Y^* together with the addition of specified at most two components is a proper Y-graph. Moreover, since G is connected, P excludes a branching node on the circle and due to the assumption A1, at most two components X_k and X_l exist. The branching cliques that X_k is placed in between do not share a vertex, and this also holds for X_l. Therefore, if the graph $G[\mathcal{Y}^*]$ is a proper Y-graph for the obtained tree Y for every $b_i \in \mathcal{B}$ in Step *iii* in the existence of both of X_k and X_l, then G is indeed a proper U-graph. Reversely, if none of X_k and X_l exists, then the branching cliques, say C_k and C_l, placed on the nodes b_k and b_l adjacent to the ends of P must share a vertex with the branching cliques placed on the corresponding ends since G is not a proper U_1-graph for some subgraph of U due to the assumption A1. This means that when b_k and b_l are considered, the properness of the induced subgraphs

on the edges incident to the ends of P are guaranteed. This applies to the case when exactly one of X_k and X_l exists. Thus, the lemma holds. $\square$

Lemma 7.7. *Consider a connected proper U-graph G which is chordal in the setting of Procedure 7.4, and assume that the maximal cliques in $\mathcal{C}_f$ can be placed on the branching nodes of the circle of U in some proper U-representation of G. If P is as defined in Step iv, then the following hold:*

1. *There exist at most two maximal cliques $C_j \neq C_k \in \mathcal{C}_f$ which have distinct minimal intersections with C_i.*
2. *At least two such maximal cliques are placed on consecutive nodes in $\mathcal{B}$.*

Proof. We prove these items separately.

1. Assume that $C_i \cap C_j$ and $C_i \cap C_k$ are distinct minimal intersections with C_i by inclusion of the vertices, and also for $C_l \in \mathcal{C}_f$, $C_i \cap C_l$ is another distinct minimal intersection with C_i. Since G is a connected proper U-graph, the induced subgraph of G placed on the circle of U forms a circular-arc graph and its cliques have a circular ordering. However, $C_i \cap C_m \neq \emptyset$ for all $C_m \neq C_i \in \mathcal{C}_f$ in Step iv, thus the existence of at least three distinct minimal intersections with C_i means that the induced subgraph of G placed on the circle of U does not form a circular-arc graph which means that there are at most two maximal cliques $C_j \neq C_k \in \mathcal{C}_f$ which have distinct minimal intersections with C_i.

2. Due to the connectedness of G, and again since the induced subgraph of G placed on the circle of U forms a circular-arc graph, there exists at least one pair of maximal cliques C_j and C_k placed on consecutive nodes of $\mathcal{B}$. $\square$

Lemma 7.8. *Considering a connected proper U-graph G which is chordal and the assignment f in the setting of Procedure 7.4, if there are more than one component of $\mathcal{X}$ placed on P, then G does not have a proper U-representation with respect to f.*

Proof. Assume that f results in a proper U-representation of G but there exists another component X_l of $\mathcal{X}$ placed on an edge of P other than e as every edge of a proper U-representation carries at most one connected component [10], mentioned also in Chapter 5. However, since P is a circle, every maximal clique in $\mathcal{C}_f$ shares a vertex in common with C_i but then, X_l can not be represented properly with respect to G. Thus, G can not have a proper U-representation with respect to f. $\square$

Lemma 7.9. *Considering a connected chordal graph G and the assignment f in the setting of Procedure 7.4, Steps iv and v correctly test whether G is a proper U-graph with respect to f.*

Proof. Note that every edge of a proper U-representation carries at most one connected component due to the properness and the connectedness of G, and an assignment of maximal cliques to the branching nodes of U determines the components between the branching nodes if G is a proper U-graph [10], mentioned also in Chapter 5 and Section 2.8. In both Steps iv and v, P is the circle and to obtain the subtree $Y \subsetneq U$, we discard a special edge of the circle. We emphasize that we return that G is a proper U-graph immediately when the corresponding tests are passed since the defined graph Y^* contains every vertex of G which may affect the properness.

If there is no connected component placed on the circle as in Step iv, then we discard the edge between any pair of consecutive branching nodes b_j and b_{j+1} from b_i such that $C_i \cap C_j$ and $C_i \cap C_{j+1}$ are minimal intersections with C_i. At least one pair of such consecutive branching nodes exist by Lemma 7.7 if G is a proper U-graph. Since we remove the intersection $C_j \cap C_{j+1}$, and there exists no connected components, if Y^* is a proper Y-graph, then G is a proper U-graph as the intersection discarded can always be represented properly between the branching cliques containing them.

Else if there exists exactly one component X_k between the branching nodes b_j and b_{j+1} as in Step v, then we again discard the edge e between them and make two copies of X_k with distinct attachments to the branching cliques C_j and C_{j+1}. The correctness is the same as in the previous case but this time, we also need to guarantee that X_k can really be represented properly. To achieve that, we additionally check whether $G[C_j \cup X_k \cup C_{j+1}]$ is a proper interval graph which is sufficient since X_k has to be placed deterministically on e if G is a proper U-graph and f results in a proper U-representation. By Lemma 7.8, we correctly move on to another assignment f' if there are more than one component of $\mathcal{X}$ placed on P. □

Theorem 7.10. *Procedure 7.4 correctly decides whether a given chordal graph is a proper U-graph in* FPT*-time.*

Proof. We first prove that Procedure 7.4 works correctly. We return that G is not a proper U-graph if the number of rich cliques is not bounded by $\mathcal{O}(|U|^3)$ due to Lemma 7.3 or the number of rich cliques is less than $|\mathcal{B}|$ due to the assumption A1. Since the set of rich cliques contains the

possible branching cliques by Lemma 7.3, we consider the assignments f of rich cliques to the branching nodes B on the circle of U. If there are more than $deg(\mathcal{B}) - |\mathcal{B}|$ connected components in $G - \mathcal{C}_f$, we move on to another assignment as f does not result in a proper U-representation by Lemma 7.5. For each branching node b_i, we consider the defined subgraph P of U, and obtain a subtree $Y \subsetneq U$ by including the other branching nodes b_k such that $C_i \cap C_k$ is non-empty. If P is already a tree, the correctness follows from Lemma 7.6, and otherwise, the correctness follows from Lemma 7.9.

We now prove that Procedure 7.4 works in FPT-time. The set of maximal cliques of G is found in polynomial time using the algorithm given in [51]. Among those maximal cliques, we identify the set $\mathcal{C}$ of rich cliques in FPT-time by Lemma 7.3. Since the number of connected components in $\mathcal{X}$ must be bounded by $|U|$ in a proper U-graph, if $\mathcal{C}$ is bounded, we can try all possible assignments f, and for each of them, we can find the deterministic placements of $\mathcal{X}$ on U in FPT-time. We test in FPT-time whether every obtained Y^* is a proper Y-graph for the tree Y and the proper interval graph recognition using the algorithm given in [21]. Therefore, the overall procedure works correctly and in FPT-time parameterized by $|U|$. □

Since all maximal cliques of graphs with polynomially many maximal cliques can be listed in polynomial time [60], we obtain the following.

Theorem 7.11. *For a fixed unicyclic graph U, one can decide whether an input graph with polynomially many maximal cliques is a proper U-graph in FPT-time parameterized by $|U|$.*

Proof. We first list all maximal cliques of our input graphs in polynomial time [60]. We identify the rich cliques among them efficiently and the number of such cliques in proper U-graphs is bounded by $\mathcal{O}(|U|^3)$ by Lemma 7.3. Thus, if this is not the case, one can correctly return that a given graph is not a proper U-graph. Now, since both the number of rich cliques and the number of branching nodes of the circle of U is bounded by $|U|$, the number of all possible assignments of rich cliques to the branching nodes of the circle of U is bounded by a function of $|U|$. For each such assignment, we use Procedure 7.4 and if at least one assignment returns that the input graph is a proper U-graph, we return its output. Otherwise, all assignments fail and the input graph has no proper U-representation certifying that it is not a proper U-graph. Since Procedure 7.4 works correctly in FPT-time and the number of all assignments is bounded by a function of $|U|$, one

can recognize the input graphs with polynomially many maximal cliques in FPT-time parameterized by $|U|$. □

We now focus on the recognition problem for proper Helly U-graphs, i.e. proper U-graphs which have at least one proper U-representation such that all cliques are Helly. It has been shown that every (proper) Helly H-graph G has at most $|H|+|E(H)|\cdot|G|$ maximal cliques which is linear on the order of the input graph G [22]. Recall that all maximal cliques of graphs with polynomially many maximal cliques can be listed in polynomial time [60]. Under the same assumptions, we modify Procedure 7.4 to decide whether a given graph G is a proper Helly U-graphs as follows:

1. Before Step 1, we start listing the maximal cliques of G using the algorithm of [60]. If the listing procedure outputs at least $1 + |U| + |E(U)|\cdot|G|$ maximal cliques, we terminate the enumeration and return that G is not a proper Helly U-graph.
2. In Step 2, we only consider the assignments f such that $G[\mathcal{C}_f]$ is a Helly circular-arc graph tested using the algorithm of [48].
3. We return either G is a proper Helly U-graph or not analogous to Procedure 7.4.

Theorem 7.12. *The above modifications to Procedure 7.4 result in an algorithm correctly recognizing proper Helly U-graphs in FPT-time.*

Proof. The first modification guarantees that the given graph G has linearly many maximal cliques on the order of G if it is a proper Helly U-graph [22]. For a tree T, every T-representation of a T-graph satisfies the Helly property [22]. Therefore, only the cliques of a proper Helly U-graph placed on the cycle of U can be non-Helly. Thus, the correctness follows from Theorem 7.11, and the first and the second modifications. Also the procedure takes FPT-time again by Theorem 7.11, and since the tests given in the first and the second modifications can be checked in polynomial time [48, 60]. □

7.3 Recognizing proper U-graphs in general

In this section, we assume that the input graphs do not have polynomially many maximal cliques since otherwise we can use Procedure 7.4 by Theorem 7.11. Therefore, we assume that the input graph is not a proper Helly

U-graph and not a chordal graph. Given a graph G, we make use of the following observations:

- If G is a proper U-graph but not a proper Helly U-graph, then every proper U-representation of G contains at least one non-Helly clique C. Since T-graphs satisfy the Helly property where T is a tree, C (along with all such cliques in G) must be placed on the circle in every U-representation if G is a (proper) U-graph. Since being non-Helly may depend on a particular U-representation, our objective is to identify *the set of non-Helly clique vertices.* These vertices belong to the cliques that are non-Helly in every proper U-representation of G.
- If G is not chordal, it contains at least one "hole", i.e. an induced cycle L of length at least 4. Since T-graphs are chordal, L (along with all holes in G) must be placed on the circle in every U-representation if G is a (proper) U-graph. Hence, our goal is to identify the set of all vertices contained in the holes, referred as *the set of hole vertices.*
- The set of all vertices that are placed completely outside of the circle of U in a proper U-representation forms a (possibly disconnected) chordal graph. Here, the term "completely on" $S \subseteq U$ means that a vertex v is represented with some subgraph of (a subdivision of) S. Conversely, if v is placed "outside of" $S \subseteq U$, it is represented with some subgraph of (a subdivision of) $U \setminus S$.

We first give the following lemma which provides *i*) an upper bound on the length of every hole L of a proper U-graph which is placed completely on the circle of U, and *ii*) an upper bound on the length of any hole L.

Lemma 7.13. *For a fixed unicyclic graph U, let G be a proper U-graph which is not a proper Helly U-graph nor a chordal graph. In a proper U-representation of G, if a hole $L \subseteq G$ is placed completely on the circle of U, then the length of L is at most* 6*. Moreover, the length of any hole $L \subseteq G$ is linearly bounded in $|U|$.*

Proof. Note that since G is not a proper Helly U-graph nor a chordal graph, there exist at least one non-Helly clique and at least one hole of G which have to be placed (completely or not) on the circle of U in every proper U-representation.

We first prove that the length of every hole $L \subseteq G$ placed completely on the circle of U is at most 6. We assume that there is no vertex of G covering the whole circle of U since otherwise such L can not be properly represented.

Let C denote a non-Helly clique of G in some proper U-representation of G. Note that if a vertex v of L represented completely on the circle is adjacent to exactly one vertex w of C, then the arc corresponding to v is contained by the arc corresponding to w resulting in a non-proper representation since v is not adjacent to the vertices in C that w is adjacent to. Thus, every vertex of L has at least 2 neighbors from C. Observe that at most 2 vertices of L can have the same pair of neighbors from C due to the properness and since L is a hole. Also, since L is a hole, two vertices of L can have at most one common neighbor from L. Since we aim to determine the maximum length of L, we fix every vertex of L to have exactly 2 neighbors from C. By this construction maximizing the length of L, if C is of size 2, then the length of L is at most 4, and otherwise there is at least one non-Helly triangle of size 3, and therefore the length of L is at most 6.

We now prove that the length of any hole $L \subseteq G$ is linearly bounded in $|U|$. Since L is induced, no three vertices of L are pairwise adjacent. Then, at most 2 vertices of L can go through the same branch on the circle of U and these vertices are consecutive in L. By the previous arguments, the number of vertices of L not going through a branch of U is bounded. Thus, to obtain the maximum length hole L, 2 consecutive vertices of L must go through a branch together and none of them should appear in other branches. Therefore, the length of any hole is bounded by $2|\mathcal{B}|$ where $\mathcal{B} \leq |U|$ denotes the number of branching nodes of U on the circle. □

Note that in proper Helly U-graphs, the length of holes is not bounded since they do not contain non-Helly cliques. However, they can be recognized in `FPT`-time without identifying their holes by Theorem 7.10. On the other hand, one can identify all holes of proper U-graphs (satisfying Helly property or not) by brute-force in `XP`-time by testing all d-tuples of vertices for $4 \leq d \leq |U|$ by Lemma 7.13. Before we show how to compute the set of hole vertices in `FPT`-time, we give the following lemmas which help us identify the set of non-Helly clique vertices of the input graph G given the set of hole vertices.

Lemma 7.14. *For a fixed unicyclic graph U, let G be a proper U-graph and L be a hole of G. Every vertex v of G adjacent to at least 3 vertices of L must be placed (not necessarily completely) on the circle of U in every (proper) U-representation of G.*

Proof. Since at most 2 vertices of L can go through the same branch on the circle of U and every vertex of a U-graph is represented by a connected

subgraph of U, these at least 3 vertices of L are consecutive in L and any vertex v of G adjacent to at least three vertices of L must contain two intersections of such vertices which are either on the circle or on the edges of U other than the circle but incident to two consecutive vertices of U on the circle. Then, the vertex v must be placed on the circle but not necessarily completely meaning that a partial representation on the circle is sufficient. □

Lemma 7.15. *For a fixed unicyclic graph U, let G be a proper U-graph which is not chordal. If G is not a proper Helly U-graph, one can in polynomial time identify the set of non-Helly clique vertices of G given the set $V_L \subseteq V(G)$ of hole vertices of G.*

Proof. Since G is not a proper Helly U-graph, there is at least one non-Helly clique C which has to be placed on the circle in every proper U-representation of G. We assume that no vertex v of C covers the whole circle of U since otherwise C is a Helly clique as v intersect also the intersection of its other two vertices. By the definition of Helly property, for every non-Helly clique of size at least 4, we have a clique of size 3 (which is a subclique of that non-Helly clique) covering the whole circle. Thus, with the upcoming method, we identify the set of non-Helly clique vertices although we do not identify all non-Helly cliques among them which can be exponentially many. Since G is not chordal, all the following hold:

- G contains at least one hole L, and every hole L of G covers the circle of U.
- Every hole L of G has a fixed cyclic ordering (up to reversal) around the circle of U since L is an induced cycle of length at least 4.
- The union of the neighborhoods of the vertices of every hole L of G is a superset of all vertices appearing in every non-Helly clique C of G.

Since $|L| \geq 4$, at least one vertex of C has at least 2 neighbors consecutive in L. Then, for at least one hole $L \in G$, at least two vertices of C have incomparable neighborhoods (by inclusion) with each other in L since there is a cyclic order on L and all vertices of C do not meet at a common point. This holds since otherwise, C can be represented as a Helly clique. By checking the neighborhoods of every clique of size 3 which union of non-empty neighborhoods on V_L equals V_L in polynomial time and comparing their neighborhoods in V_L, one can identify every such clique C appearing as a non-Helly clique in every proper U-representation of G given V_L. □

Recall that a dominating edge of a circular-arc graph is an edge covering the whole circle and having two disconnected intersections. Considering a proper U-graph G, we focus on the dominating edges around the circle of U which may not form dominating edges in G. We refer to the union of end-vertices of all dominating edges covering the circle of U as *the set of dominating edge vertices.*

Lemma 7.16. *For a fixed unicyclic graph U, let G be a proper U-graph which is not chordal. Then, one can identify the set of dominating edge vertices of G in polynomial time given the set V_L of hole vertices of G.*

Proof. If none of two vertices u and v of such a dominating edge e covers the whole circle of U, those vertices have incomparable (by inclusion) neighborhoods in at least one hole and can be identified in polynomial time as there are polynomially many edges. Otherwise, the vertex u dominates the whole circle but may not cover it. Then, the neigborhood of v in the circle is a subset of the neigborhood of u in the circle but in this case, v is actually a vertex of some hole of G, thus already given in V_L. □

We give the following procedure to obtain the set of *hole vertices*, and the possibly empty set of *revealed vertices* which also must be represented (not necessarily completely) on the circle of U if G is a proper U-graph.

Procedure 7.17. Given a connected graph G which is not chordal, we identify the set V_L of hole vertices and (if exists) a set of revealed vertices V_R of $V(G) \setminus V_L$ as follows:

1. Let V_L and V_R denote the initially empty sets of hole vertices and revealed vertices of G, respectively.
2. For each vertex $u \in V(G)$ and each pair of its neighbors $v \neq w \in N_G(v)$ such that v and w are not adjacent to each other:
   - Identify the shortest path P between v and w in $G - (N_G[u] \setminus \{v, w\})$ using Dijkstra's algorithm [29]. If $P \neq \emptyset$, $V_L \leftarrow V_L \cup V(P) \cup \{u, v, w\}$.
3. For each vertex $u \in V(G) \setminus V_L$:
   - If u has at least three neighbors in V_L forming an induced path in G, then $V_R \leftarrow V_R \cup u$. This can be checked by identifying whether there exist three distinct vertices $v, w, x \in (N_G[u] \setminus u) \cap V_L$ such that $\{v, w, x\}$ forms an induced path of length 2 in G.

4. Return V_L and V_R as the sets of hole and revealed vertices, respectively.

Lemma 7.18. *Procedure 7.17 correctly identifies the set V_L of hole vertices and the set V_R of revealed vertices in polynomial time.*

Proof. The correctness and complexity of identifying the set V_L of hole vertices depends on the Dijkstra's algorithm [29], and greedily checking the neighborhood of each vertex in $V(G)$. We now prove that V_R is indeed a revealed set. It is clear that V_R can be computed in polynomial time. Since only the consecutive vertices of holes are connected by an edge, each vertex u having at least three neighbors $v, w, x \in V_L$ appearing as an induced path in G has to be placed on the circle by Lemma 7.14. Also, if u has more that three such neighbors, some triple among them still form an induced path of length 2, and therefore, it is sufficient to check all three neighbors of u in polynomial time. □

Note that the hole vertices in V_L found using Procedure 7.17 lie on the circle of U in every proper U-representation of G if G is a proper U-graph since the rest of G is chordal.

Lemma 7.19. *Let G be a connected proper U-graph, where U is a fixed unicyclic graph. Consider the following sets of vertices:*

- *V_L: The set of hole vertices found using Procedure 7.17.*
- *V_R: The set of revealed vertices found using Procedure 7.17.*
- *V_C: The union of sets of non-Helly clique vertices and dominating edge vertices found using Lemma 7.15 and Lemma 7.16 given $V_L \cup V_R$.*

Then, both $G[V_L \cup V_R]$ and $G[V_L \cup V_R \cup V_C]$ are circular-arc graphs with fixed cyclic orderings (up to reversal).

Proof. As G is a proper U-graph, the vertices in $V_L \cup V_R$ must be placed (not necessarily completely) on the circle of U by Lemma 7.18, and V_C must be placed (not necessarily completely) on the circle of U since every T-graph satisfies the Helly property for a tree T, both $G[V_L \cup V_R]$ and $G[V_L \cup V_R \cup V_C]$ form circular-arc graphs. In addition, the cyclic order of $G[V_L \cup V_R]$ is fixed (up to reversal) since V_L only contains the vertices of holes and V_R only contains the vertices which neighbors in V_L form induced paths. Since, holes and induced paths have fixed circular orderings, $G[V_L \cup V_R]$ has a fixed circular ordering (up to reversal). Also, each vertex of V_C has neighbors in

$V_L \cup V_R$ which have a fixed ordering in $G[V_L \cup V_R]$. This also means that $G[V_L \cup V_R \cup V_C]$ has a fixed circular ordering (up to reversal). $\square$

We first give the following definition for an input graph G before we show how to recognize whether G is a proper U-graph in general.

Definition 7.20. *Let V_L, V_R and V_C be as in the setting of Lemma 7.19 for some connected graph G. Let $\mathcal{X}$ be the set of components of $G-(V_L \cup V_R \cup V_C)$ where $\mathcal{X}$ has at most $|U|$ components, and each component of $\mathcal{X}$ has at most two upper attachments in $G[V_L \cup V_R \cup V_C]$. Let $\mathcal{B}$ denote the branching nodes on the circle of the fixed graph U. We call a partition of $\mathcal{X}$ into subsets $\mathcal{X}_1, \ldots, \mathcal{X}_{|\mathcal{B}|}$ a* Helly partition *if the following hold:*

1. *For every pair $1 \leq i < j \leq |\mathcal{B}|$, it holds that $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$.*
2. *For each $1 \leq i \leq |\mathcal{B}|$, $\mathcal{X}_i$ contains at most $deg(b_i) - 2$ components with exactly 1 upper attachment where $deg(b_i)$ denotes the degree of $b_i \in \mathcal{B}$.*
3. *Let $\mathcal{F}_1, \ldots, \mathcal{F}_{|\mathcal{B}|}$ be the subfamilies of $\mathcal{F}$ such that for every $1 \leq i \leq |\mathcal{B}|$, $\mathcal{F}_i$ consists of exactly 1 upper attachment of each component $X_j \in \mathcal{X}_i$ with respect to the fixed cyclic order of $G[V_L \cup V_R]$.*
4. *For each $b_i \in \mathcal{B}$, it holds that $|\mathcal{X}_i| = |\mathcal{F}_i| \leq deg(b_i)$.*
5. *The union $C_{\mathcal{F}_i}$ of all vertices of G contained in each $\mathcal{F}_i$ corresponds to a clique in G.*
6. *The circular-arc graph $G[V_L \cup V_R \cup V_C]$ has a circular-ones property with $C_{\mathcal{F}_1}, \ldots, C_{\mathcal{F}_{|\mathcal{B}|}}$ appearing in this order.*
7. *For every pair $1 \leq i \neq j \leq |\mathcal{B}|$, there exists no $X_k \in \mathcal{X}_i$ such that any of the (at most 2) upper attachments of X_k has an intersection with $C_{\mathcal{F}_j}$ which is a strict superset of the intersection of the upper attachment of some $X_l \in \mathcal{X}_j$.*

Lemma 7.21. *Considering Definition 7.20, the number of Helly partitions of $\mathcal{X}$ with distinct choices of the subfamilies $\mathcal{F}_1, \ldots, \mathcal{F}_{|\mathcal{B}|}$ is at most $(|U| + 4)^{|U|}$, and they can be computed in* FPT*-time.*

Proof. Since $\mathcal{X}$ has at most $|U|$ connected components and $|\mathcal{B}| < |U|$, the number of distinct partitions of $\mathcal{X}$ into $\mathcal{X}_1, \ldots, \mathcal{X}_{|\mathcal{B}|}$ is upper bounded by $|U|^{|\mathcal{B}|} \leq |U|^{|U|}$. In addition, the number of upper attachments is bounded by $2|U|$ since each connected component has at most two upper attachments. Thus, for each partition, there are at most $4^{|\mathcal{B}|} \leq 4^{|U|}$ distinct choices of such

subfamilies as each $\mathcal{X}_i$ contains at most two connected components with two upper attachments. Overall, we have at most $(|U|+4)^{|U|}$ partitions $\mathcal{X}$ with distinct choices of the subfamilies of $\mathcal{F}$.

We can in polynomial time check whether a set of vertices corresponds to a clique of G, and a given subset $C_{\mathcal{F}_1}, \ldots, C_{\mathcal{F}_{|\mathcal{B}|}}$ of cliques placed on the circle can appear in this order using the PC-tree of the circular-arc graph $G[V_L \cup V_R \cup V_C]$ as in [26]. We also note here that this check is not the same as partial representation extension with predefined sets $C_{\mathcal{F}_1}, \ldots, C_{\mathcal{F}_{|\mathcal{B}|}}$ which has been shown to be NP-complete [32] since we do not specify the endpoints of the arcs for the corresponding vertices. □

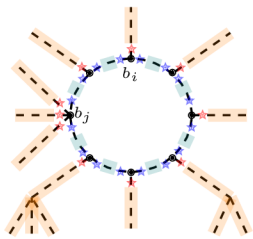


Figure 7.4: After the sets V_L, V_R and V_C described in Lemma 7.19 are found, teal and orange components are obtained without their deterministic placements on U. Teal and orange components have two and one upper attachments, marked with blue and red stars, respectively. Teal components must and reversely orange components can not be placed on the circle of U.

We now give Procedure 7.22 to recognize general proper U-graphs.

Procedure 7.22. Given a connected graph G on n vertices and a fixed unicyclic graph U, we decide whether G is a proper U-graph as follows:

1. If G is chordal, then run Procedure 7.4 and return its output.
2. Check whether G is a proper Helly U-graph using Procedure 7.4 with the modifications proven in Theorem 7.12. If it returns that G is a proper Helly U-graph, return its output.
3. Find the set V_L of hole vertices and the set V_R of revealed vertices using Procedure 7.17.
4. Using Lemma 7.15 and Lemma 7.16 given the set $V_L \cup V_R$, identify the union V_C of sets of non-Helly clique vertices and dominating edge

vertices. If $G[V_L \cup V_R \cup V_C]$ is not a circular-arc graph, return that G is not a proper U-graph by Lemma 7.19.

5. Find the set $\mathcal{X}$ of connected components of $G - (V_L \cup V_R \cup V_C)$. If $\mathcal{X}$ has more than $|U|$ components, return that G is not a proper U-graph.

6. Identify the upper attachments of each component of $\mathcal{X}$ in $G[V_L \cup V_L \cup V_C]$ with respect to the fixed cyclic order of $G[V_L \cup V_R]$. If there is a connected component with at least 3 upper attachments, return that G is not a proper U-graph.

7. Every connected component in $\mathcal{X}$ has at most 2 upper attachments. Let $\mathcal{B}$ denote the branching nodes on the circle of U, $deg(\mathcal{B})$ be the sum of degrees of nodes in $\mathcal{B}$ and $\mathcal{F}$ be the family of all upper attachments. If there are more than $deg(\mathcal{B})$ such attachments in $\mathcal{F}$, return that G is not a proper U-graph.

8. To identify the Helly cliques placed on $\mathcal{B}$ (if G is a proper U-graph), compute every Helly partition of $\mathcal{X}$ into $\mathcal{X}_1, \ldots, \mathcal{X}_{|\mathcal{B}|}$ and the cliques $\{C_{\mathcal{F}_1}, \ldots, C_{\mathcal{F}_{|\mathcal{B}|}}\}$ for distinct choices of subfamilies $\{\mathcal{F}_1, \ldots, \mathcal{F}_{|\mathcal{B}|}\}$.

9. For each Helly partition $\mathcal{X}_1, \ldots, \mathcal{X}_{|\mathcal{B}|}$ and for each choice of subfamilies of $\mathcal{F}$, use Procedure 7.4 from Step (a) with $\mathcal{C}_f = \{C_{\mathcal{F}_1}, \ldots, C_{\mathcal{F}_{|\mathcal{B}|}}\}$ and the connected components $\mathcal{X}$ of $G - \mathcal{C}_f$. ▷ *Here, we may additionally identify a set of maximal Helly-cliques containing the corresponding Helly-cliques by Lemma 7.23.*

   (a) If Procedure 7.4 returns that G is a proper U-graph, return its output.

10. Return that G is not a proper U-graph.

Lemma 7.23. *Considering the setting in Step* 8 *of Procedure 7.22, one can in* FPT*-time identify a set of maximal Helly-cliques to be used in Procedure 7.22 as commented in Step* 9 *if such a representation exists.*

Proof. By Lemma 7.21, we have a bounded number of partitions $\mathcal{X}_1, \ldots, \mathcal{X}_{|\mathcal{B}|}$ with distinct choices of the subfamilies $\mathcal{F}_1, \ldots, \mathcal{F}_{|\mathcal{B}|}$, and by conditions d, e and g of the definition of a Helly partition, for each $1 \leq i \leq j \leq |\mathcal{B}|$, any $X_k \in \mathcal{X}_i$ and its upper attachment contains a maximal clique C of G which can not be obtained from any other $\mathcal{X}_j$ for $j \neq i$ as the connected components in $\mathcal{X}$ are disjoint. Since any set $C_{\mathcal{F}_1}, \ldots, C_{\mathcal{F}_{|\mathcal{B}|}}$ of cliques obtained by a Helly partition of $\mathcal{X}$ has a circular-ones ordering, any non-maximal $C_{\mathcal{F}_i}$ can be extended to a maximal clique if there exists a component $X_j \in \mathcal{X}_i$ such that

the upper attachment of X_j is a superset of the upper attachment of every other component in $C_{\mathcal{F}_i}$. Otherwise, either $C_{\mathcal{F}_i}$ is already a maximal clique, or its vertices appear in some Helly-clique and it can not be extended to a maximal clique to be placed on a branch. On the other hand, every clique in $C_{\mathcal{F}_1}, \ldots, C_{\mathcal{F}_{|\mathcal{B}|}}$ which can be extended to a maximal clique can be computed in FPT-time and used in Procedure 7.4 from Step (a) as the current bijection $\mathcal{C}_f$. □

We shortly mention that the proof of Lemma 7.23 follows from the bound on the number of connected components. Thus, we get the following.

Theorem 7.24. *Procedure 7.22 correctly decides whether a given graph is a proper U-graph in FPT-time parameterized by $|U|$.*

Proof. We first prove that Procedure 7.22 works correctly. The correctness of Step 1 to 4 follows from Theorem 7.10, Theorem 7.12, Lemma 7.15 and Lemma 7.18. In Step 5, if there are more than $|U|$ connected components, then G can not be a proper U-graph since there exists no proper U-representation in this case [10].

In Step 6, the cyclic order of $G[V_L \cup V_R]$ is fixed (up to reversal) by Lemma 7.19, and every component must have at most 2 upper attachments since they are either placed on the circle of U having exactly 2 distinct upper attachments due to the existence of a proper U-representation, or placed not on the circle of U having exactly 1 upper attachment due to the connectedness under our assumptions if G is a proper U-graph. Then, in Step 7, there are at most $|\mathcal{B}|$ connected components placed on the circle of U with exactly two distinct upper attachments, and at most $deg(\mathcal{B}) - 2|\mathcal{B}|$ connected components with one upper attachment placed not on the circle of U. Thus, if there are more than $2|\mathcal{B}| + deg(\mathcal{B}) - 2|\mathcal{B}| = deg(\mathcal{B})$ upper attachments, G can not be a proper U-graph.

In Step 8, even though we do not know the exact placements of the connected components on the edges of U, we try all possible placements. Due to the necessity of having a proper U-representation if G is a proper U-graph, the intersections of the upper attachments of the connected components placed on incident edges with a branching clique must be maximal, and since $G[V_L \cup V_R \cup V_C]$ is a circular-arc graph by Lemma 7.19, the branching cliques $C_{\mathcal{F}_1}, \ldots, C_{\mathcal{F}_{|\mathcal{B}|}}$ have a circular-ones ordering consistent with $G[V_L \cup V_R \cup V_C]$. Note here that we do not check whether $G[V_L \cup V_R \cup V_C]$ is a proper circular-arc graph since properness can be achieved on the branches of U.

In Step 9, we try each Helly partition as at least one of them passes the test of Procedure 7.4 if G is a proper U-graph using the identified Helly cliques. In Step 10, no partition gives a proper U-representation meaning that G is not a proper U-graph.

We now prove that Procedure 7.22 works in FPT-time. Steps 1 to 4 takes FPT-time due to Theorem 7.10, Theorem 7.12, Lemma 7.15 and Lemma 7.18, and Steps 5, 6 and 7 takes trivially polynomial time. In Step 8, the number of Helly partitions of $\mathcal{X}$ into $\mathcal{X}_1, \ldots, \mathcal{X}_{|\mathcal{B}|}$ with distinct choices of subfamilies $\mathcal{F}_1, \ldots, \mathcal{F}_{|\mathcal{B}|}$ is bounded by our parameter $|U|$ and can be computed in FPT-time by Lemma 7.21. Finally, we use the FPT-approach given as Procedure 7.4 for each of Helly partition of $\mathcal{X}$ with a distinct choice of subfamilies of $\mathcal{F}$ in Step 9 and use its result. Therefore, the overall procedure works correctly and in FPT-time. □

7.4 Isomorphism of proper U-graphs

In this section, we show how to test the isomorphism of two proper U-graphs in FPT-time parameterized by $|U|$. We modify Procedure 7.4 and Procedure 7.22, and have the previous assumptions given on the corresponding sections. Here, we assume connectedness also since the isomorphism of each pair of components of given disconnected graphs can be checked pairwise. We first modify Procedure 7.4 as follows:

1. On the input, we are given the fixed unicyclic graph U, and two connected proper U-graphs G_1 and G_2 on n vertices which are both chordal or proper Helly U-graphs.

2. Let $\mathcal{C}_i$ be the isomorphism invariant set of rich cliques of G_i for $i \in \{1, 2\}$ by Lemma 7.3. In Step 1, we check whether $|\mathcal{C}_1| = |\mathcal{C}_2|$ holds.

3. Using Procedure 7.4, we fix a proper U-representation of G_1 where $f : \mathcal{C}_1 \to \mathcal{B}$ denotes the assignment of branching cliques on the circle. Let $\mathcal{C}_1^f \subseteq \mathcal{C}_1$ be those cliques and C_1^i be the maximal clique of $\mathcal{C}_1^f$ placed on the branching node $b_i \in \mathcal{B}$ in the current assignment f.

4. We denote each assignment $\mathcal{C}_2 \to \mathcal{B}$ by g, and only consider those g which match to f as being in the case of Step *iii*, *iv* and *v* of Procedure 7.4. Therefore, P is the same for both graphs in every step. Let $\mathcal{C}_2^g \subseteq \mathcal{C}_2$ be the branching cliques on the circle w.r.t. g, and C_2^i be the maximal clique of $\mathcal{C}_2^g$ placed on the branching node $b_i \in \mathcal{B}$ in the

current assignment g. Here, we require also that $|C_1^i| = |C_2^i|$ for every $b_i \in \mathcal{B}$.

5. For each branching node b_i of U, we compute the subtree $Y_1 \subsetneq U$ and the graph $\mathcal{Y}_1^*$ obtained by G_1 defined as in Procedure 7.4. Let $Y_2 \subsetneq U$ be the corresponding subtree for G_2 and $\mathcal{Y}_2^*$ be the graph obtained by G_2 in the current assignment g.

6. We first check whether $Y_1 \simeq Y_2$, and instead of checking for proper Y-graph recognition, we test whether $G_1[\mathcal{Y}_1^*] \simeq G_2[\mathcal{Y}_2^*]$ using the proper Y-graph isomorphism testing for the tree $Y_1 \simeq Y_2$ given in [10].

7. In Step *iv*, we additionally check whether $|C_j^1 \cap C_{j+1}^1|$ and $|C_j^2 \cap C_{j+1}^2|$ are equal.

8. In Step *v*, we check whether the defined interval graphs are isomorphic proper interval graphs using the algorithm given in [17].

9. We return either $G_1 \simeq G_2$ or $G_1 \not\simeq G_2$ analogous to Procedure 7.4.

Theorem 7.25. *The above modifications to Procedure 7.4 result in a correct* FPT*-time isomorphism testing for given two proper U-graphs which are both proper Helly U-graphs or chordal graphs.*

Proof. We prove that our modifications correctly cover the cases in the Steps *iii*, *iv* and *v* of Procedure 7.4. Precisely, we show that any attachment ignored in one step of the modified procedure is checked in some other step, thus the neighborhoods of bijected vertices can also be bijected to each other.

In Step *iii*, Y_1 and Y_2 do not contain at least one branching node b_j. Therefore, $C_i^k \cap G_i[\mathcal{Y}_i^*] = \emptyset$ and the attachment of C_i^k in $G_i[\mathcal{Y}_i^*]$ is ignored for $i \in \{1, 2\}$. Due to our assumptions, C_i^k for $i \in \{1, 2\}$ has a non-empty attachment in $G_i[\mathcal{Y}_i^*]$ which results in having b_k together with b_{k-1} and b_{k+1}, and possibly some other branching nodes of the circle, in at least one other $Y_i' \neq Y_i$ even though $Y_i' \simeq Y_i$ may hold, thus the attachments of any C_i^k ignored in this step are checked in another step.

In Step *iv*, there is no components on the edges of P and we consider the furthest branching nodes b_j and b_{j+1} from b_i. Since there is no ignored attachment in this case and no component between b_i and b_{j+1}, having $|C_1^j \cap C_1^{j+1}| = |C_2^j \cap C_2^{j+1}|$ is sufficient.

In Step *v*, there is exactly one component on the edge e of P. Similar to Step *iv*, no attachments are ignored and proper interval graph isomorphism

testing for copied components and their immediate attachments guarantees a bijection between the same vertices.

Since the proper T-graph isomorphism for a tree T takes FPT-time [10] and proper interval graph isomorphism takes linear time [17], the overall proper U-graph isomorphism approach for given two proper Helly U-graphs or two chordal graphs works correctly in FPT-time. □

With analogous modifications to Procedure 7.22, we get the following.

Theorem 7.26. *The isomorphism problem for proper U-graphs can be solved in FPT-time parameterized by $|U|$.*

Proof. We first give the following modifications to Procedure 7.22 for general proper U-graph isomorphism in addition to the modifications applied to Procedure 7.4:

1. Let V_{L_i} and V_{R_i} be the sets of hole and revealed vertices, and V_{C_i} be the union of sets of non-Helly clique and dominating edge vertices of G_i for $i \in \{1, 2\}$. We additionally check whether $G[V_{L_1} \cup V_{R_1} \cup V_{C_1}] \simeq H[V_{L_2} \cup V_{R_2} \cup V_{C_2}]$ using the algorithm of [47].
2. Let $\mathcal{X}^i$ be the set of components of $G_i - (V_{L_i} \cup V_{R_i} \cup V_{C_i})$, and $\mathcal{F}^{\mathcal{X}^i}$ be the family of all upper attachments of $\mathcal{X}^i$ for $i \in \{1, 2\}$. We additionally check whether the number of upper attachments in $\mathcal{F}^{\mathcal{X}^1}$ and $\mathcal{F}^{\mathcal{X}^2}$ are the same.
3. Let $\mathcal{X}_1^1, \ldots, \mathcal{X}_s^1$ and $\mathcal{X}_1^2, \ldots, \mathcal{X}_t^2$ be the Helly partitions of $\mathcal{X}^1$ and $\mathcal{X}^2$, respectively. We additionally check whether $s = t$ and $|\mathcal{X}_i^1| = |\mathcal{X}_i^2|$ for all $1 \leq i \leq s$.
4. Let $\mathcal{F}_1^{\mathcal{X}^i}, \ldots, \mathcal{F}_s^{\mathcal{X}^i}$ be a distinct choice of the subfamilies of $\mathcal{F}^{\mathcal{X}^i}$ for $i \in \{1, 2\}$. We additionally check whether $|\mathcal{F}_i^{\mathcal{X}^1}| = |\mathcal{F}_i^{\mathcal{X}^2}|$ for all $1 \leq i \leq s$.
5. Let $C_{\mathcal{F}_j^{\mathcal{X}^i}}$ be the union of all vertices of G_i contained in $\mathcal{F}_j^{\mathcal{X}^i}$ for $i \in \{1, 2\}$ and $1 \leq j \leq s$.

We now prove the correctness and the complexity. By Lemma 7.15, Lemma 7.16 and Lemma 7.18 showing the sets of hole vertices, revealed vertices, non-Helly clique vertices and dominating edge vertices can be identified in FPT-time, Lemma 7.21 showing the number of all possible partitions of $\mathcal{X}^i$ into $\mathcal{X}_1^i, \ldots, \mathcal{X}_{|\mathcal{B}|}^i$ is bounded by a function of the parameter $|U|$, by Theorem 7.24

showing that the Procedure 7.4 can be extended to recognize proper U-graphs in general, and by Lemma 7.25 showing that the given modifications to Procedure 7.4 results in a correct FPT-time isomorphism testing for proper U-graphs which are both proper Helly U-graphs or chordal graphs, the isomorphism problem for proper U-graphs in general can be tested in FPT-time. We also note that we can use Lemma 7.23 to identify every set of maximal Helly-cliques possible to be used in isomorphism testing. Since the number of components is bounded in G and H, here we get a bounded number of choices for such sets. □

7.5 GI-completeness for H-graphs and proper H-graphs

In this section, we prove that if H is not a tree or a unicyclic graph, then the isomorphism problem is GI-complete on proper H-graphs, i.e., as hard as the general graph isomorphism problem. Note that, for every H, the class of proper H-graphs is a subclass of H-graphs. Thus, this implies the GI-completeness of the isomorphism problem for H-graphs as well. The key is the following theorem where B denotes the 5-vertex graph consisting of two triangles sharing precisely one vertex.

Theorem 7.27. *The isomorphism problem for the class of all proper B-graphs is GI-complete.*

Note that if a graph H is a minor of a graph H', then every H-graph is an H'-graph. This immediately gives the following.

Corollary 7.28. *If H is not unicyclic, then the isomorphism problem for the class of all proper H-graphs and the class of all H-graphs is GI-complete.*

To prove Theorem 7.27, we first give several lemmas.

Lemma 7.29. *If G is a disjoint union of stars, then its complement $\overline{G}$ is a proper circular-arc graph.*

Proof. Let $X \subseteq V(G)$ be the vertices of degree 1 in G, and let $Y = V(G) \setminus X$. For every vertex $v \in Y$, $N_G(v)$ forms an equivalence class of true twins in $\overline{G}$ where true twins are a set of vertices with the same neighborhood which are pairwise adjacent, i.e. the vertices with the same closed neighborhood. The quotient graph of $\overline{G}$ with respect to this equivalence relation is exactly

the complement of a matching $K_{2n} - nK_2$, which is a well-known proper circular-arc graph. Therefore, $\overline{G}$ is also a proper circular-arc graph. $\square$

From a connect graph $G = (V, E)$, we construct a new graph G^* as follows:

1. Let $G' = (V \cup V_i, E')$ be the graph resulting from subdividing each edge of G.
2. Let $G'' = (V \cup V_i \cup V_j, E'')$ be the graph resulting from subdividing each edge of G'.
3. Let G''' be the graph obtained from G'' by adding all the edges between V and V_i.
4. Finally, we set the graph G^* to be the complement of G'''.

Lemma 7.30. *Let G be a connected graph. Then, G^* is a proper B-graph.*

Proof. First note that the induced subgraphs $G''[V \cup V_j]$ and $G''[V_i \cup V_j]$ are disjoint unions of stars. Thus, by Lemma 7.29, the induced subgraphs $G^*[V \cup V_j]$ and $G^*[V_i \cup V_j]$ are circular-arc graphs with representations $\mathcal{R}_i \colon V \cup V_j \to \mathcal{S}(C_i)$ and $\mathcal{R}_j \colon V_i \cup V_j \to \mathcal{S}(C_j)$ where $\mathcal{S}(C_i)$ and $\mathcal{S}(C_j)$ denote the set of all connected subgraphs of some cycles C_i and C_j, respectively. Moreover, both representations can be chosen so that there are points $p_i \in V(C_i)$ and $p_j \in V(C_j)$ such that

$$p_i \in \bigcap_{v \in V_j} \mathcal{R}_i(v), \quad p_i \notin \bigcup_{v \in V} \mathcal{R}_i(v), \quad \text{and} \quad p_j \in \bigcap_{v \in V_j} \mathcal{R}_j(v), \quad p_j \notin \bigcup_{v \in V_i} \mathcal{R}_j(v). \tag{7.1}$$

Let H be the graph obtained from C_i and C_j by identifying the points p_i and p_j. Clearly H is a subdivision of B. Let $\mathcal{R} \colon V \cup V_i \cup V_j \to \mathcal{S}(H)$ be the mapping defined by

$$\mathcal{R}(v) = \begin{cases} \mathcal{R}_i(v) & \text{if } v \in V, \\ \mathcal{R}_i(v) \cup \mathcal{R}_j(v) & \text{if } v \in V_j, \\ \mathcal{R}_j(v) & \text{if } v \in V_i. \end{cases}$$

Since there are no edges between V and V_i in G^*, it follows from (7.1) that $\mathcal{R}$ is an H-representation of G^* and, therefore, G^* is an B-graph.

It remains to note that both $\mathcal{R}_i$ and $\mathcal{R}_j$ are proper representations of the corresponding circular-arc graphs. Thus, the resulting representation $\mathcal{R}$ of G^* is also a proper B-representation. $\square$

Lemma 7.31. *Let G_1 and G_2 be connected graphs with minimum degree at least three. Then, G_1 and G_2 are isomorphic if and only if G_1^* and G_2^* are isomorphic.*

Proof. Since subdividing each edge and taking complements preserve the isomorphism relation, it follows that $G_1 \cong G_2$ if and only if $G_1^{''} \cong G_2^{''}$, and, $G_1^{'''} \cong G_2^{'''}$ if and only if ${G_1}^* \cong {G_2}^*$. Thus, it suffices to prove that $G_1^{''} \cong G_2^{''}$ if and only if $G_1^{'''} \cong G_2^{'''}$.

Let $f\colon G_1 \to G_2$ be an isomorphism and let $f^{''}\colon G_1^{''} \to G_2^{''}$ be the isomorphism such that $f^{''}|_{V(G_1)} = f$, i.e., the restriction of $f^{''}$ to $V(G_1)$ equals to f. We have

$$f^{''}(V(G_1)) = V(G_2) \quad \text{and} \quad f^{''}(V_i(G_1)) = V_i(G_2). \tag{7.2}$$

The graph $G_1^{'''}$ is constructed by adding all edges between $V(G_1)$ and $V_i(G_1)$. Likewise, the graph $G_2^{'''}$ is constructed by adding all edges between $V(G_2)$ and $V_i(G_2)$. From (7.2) it follows that $f^{''}$ is also an isomorphism from $G_1^{'''}$ to $G_2^{'''}$.

For the reverse implication, first note that for any vertex v of $G_1^{'''}$, we have

$$deg_{G_1^{'''}}(v) = \begin{cases} deg_{G_1}(v) + |V'(G_1)| & \text{if } v \in V(G_1), \\ 2 + |V(G_1)| & \text{if } v \in V'(G_1), \\ 2 & \text{if } v \in V''(G_1). \end{cases}$$

Note that the degree is constant on $V'(G_1)$ and on $V''(G_1)$. An analogous formula is true for every vertex in $G_2^{'''}$. Since the minimum degree in G_1 is three, we have

$$2+|V(G_1)| \geq 6 \quad \text{and} \quad deg_{G_1}(v)+|V'(G_1)| \geq 3+3/2|V(G_1)| > 2+|V(G_1)|.$$

Again, analogous inequalities hold in $G_2^{'''}$. It follows that an isomorphism $f^{'''}\colon G_1^{'''} \to G_2^{'''}$ satisfies

$$f^{'''}(V(G_1)) = V(G_2), \quad f^{'''}(V'(G_1)) = V'(G_2) \quad \text{and} \quad f^{'''}(V''(G_1)) = V''(G_2).$$

Thus, $f^{'''}$ is also an isomorphism $G_1^{''} \to G_2^{''}$. □

Proof of Theorem 7.27. ${G_1}^*$ can be constructed from G_1 in polynomial time. Moreover, by Lemma 7.30, ${G_1}^*$ is a proper B-graph. It is well-known that the isomorphism problem for the class of all graphs with minimum degree

three is GI-complete. By Lemma 7.31, G_1 and G_2 with minimum degree three are isomorphic if and only if ${G_1}^*$ and ${G_2}^*$ are isomorphic. It follows that the isomorphism problem for the class of all proper B-graphs is GI-complete. $\square$

Chapter 8

Recognizing L-graphs, strictly cyclic U-graphs, and Helly $\mathcal{U}$-graphs

In this chapter, we study the parameterized complexity of the recognition problem on general U-graphs. This is a simplified exposition of our preprint [7] to omit further technical details for the sake of simplicity, based on our multi-author paper [8]. We first show that for every fixed graph H containing two different cycles, recognition of H-graphs is NP-complete. We then focus on L-graphs, and prove that they can be recognized in polynomial time. We finally show that for a fixed unicyclic graph U, the recognition of U-graphs can be solved in XP-time parameterized by the size of U when the input graphs are strongly cyclic. We also show that Helly $\mathcal{U}$-graphs can be recognized in polynomial time where $\mathcal{U}$ is the class of all unicyclic graphs.

8.1 NP-completeness reduction to H-graphs when H contains at least two cycles sharing a vertex

In this section, we show that the recognition problem is NP-complete for H-graphs when H has at least two cycles intersecting in one vertex. A *2-track graph* is a graph $G = (V, E)$ such that $E(G)$ can be partitioned into E_1, E_2 so that both $(V(G), E_1)$ and $(V(G), E_2)$ are interval graphs. The problem Bipartite 2-Track is to decide whether a given bipartite graph G is 2-track.

The key is the following theorem where B denotes the 5-vertex graph consisting of two triangles sharing one vertex.

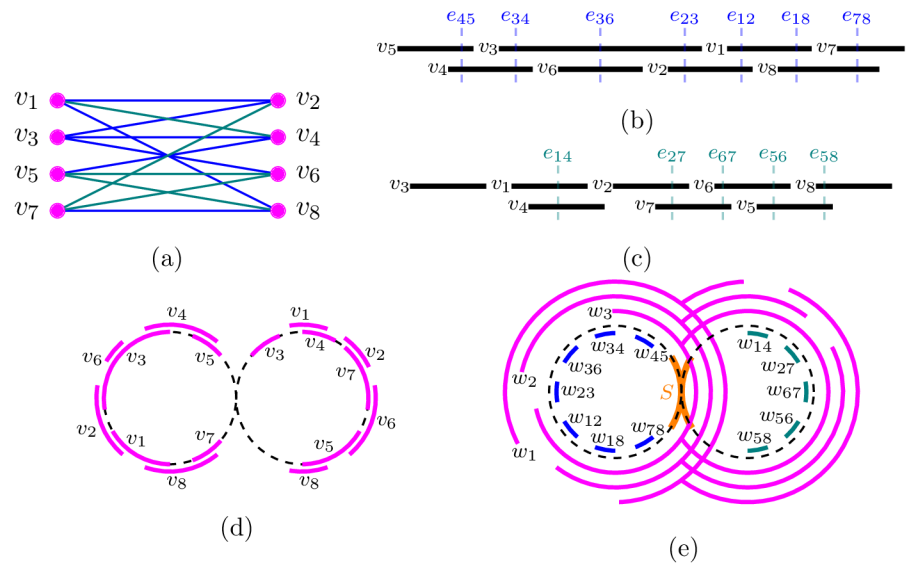


Figure 8.1: An illustration of the reduction given in the proof of Theorem 8.1.

Theorem 8.1. *The recognition problem for B-graphs is* NP*-complete.*

Proof. It is easy to see NP-membership of B-graph recognition [23], and to prove its NP-hardness, we reduce from Bipartite 2-Track which was shown to be NP-hard [39].

From a bipartite graph G, we will construct a graph G' that is a B-graph if and only if G is 2-track as follows:

1. Let S be the graph of order 9 obtained by subdividing every edge of the star $K_{1,4}$ once.
2. The vertex set $V(G')$ consists of $V(S)$, a vertex w_i for every vertex $v_i \in V(G)$, and an *edge-vertex* w_{ij} for every edge $e_{ij} = \{v_i, v_j\} \in E(G)$.
3. The edge set $E(G')$ consists of $\binom{V(G)}{2}$, $E(S)$ and the edges which make w_{ij} adjacent to $v_k \in V(G) \setminus \{v_i, v_j\}$ for every edge $e_{ij} \in E(G)$.
4. Finally, we add every edge between $V(G)$ and $V(S)$.

Figure 8.1 illustrates above construction. In Figure 8.1a, we see a bipartite 2-track graph G with $E(G)$ partitioned into blue edge set E_1 and teal edge set E_2. Both subgraphs $(V(G), E_1)$ and $(V(G), E_2)$ form interval graphs and they are given in Figure 8.2b and Figure 8.1c, respectively. These two interval graphs can be embedded on B as in Figure 8.1d where B is shown by dashed black graph and the embeddings of $(V(G), E_1)$ and $(V(G), E_2)$ are shown with magenta intervals on the first and the second circles of B, respectively. Figure 8.1e shows the constructed B-graph G' where S is shown by orange X shape in the middle of B, the representations of the edge vertices for the edges in E_1 and E_2 are shown by blue and teal arcs, respectively, and for the sake of simplicity, the representations of the first three vertices v_1, v_2 and v_3 in $V(G)$ are shown by magenta.

Given a bipartite graph G, it is obvious that G' can be constructed in polynomial time. It remains to show that G is 2-track if and only if G' is a B-graph.

($\Rightarrow$) We show that G' is a B-graph if the bipartite graph G is 2-track. We achieve this by constructing a B-representation of G' as follows:

- Let $\mathcal{R}$ be the B-representation of G'.
- Fix any interval representation $\mathcal{R}_1$ and $\mathcal{R}_2$ of $(V(G), E_1)$ and of $(V(G), E_2)$, respectively. Let $\mathcal{R}_i(v_j)$ denote the interval representing v_j in $(V(G), E_i)$ with respect to $\mathcal{R}_i$ for $i \in \{1, 2\}$.
- Recall that B denotes the 5-vertex graph consisting of two triangle with one vertex x identified. Embed $\mathcal{R}_1$ and $\mathcal{R}_2$ on the first circle and the second circle of B, respectively, without passing through x. Note that neither the vertices of the interval graphs $(V(G), E_1)$ and of $(V(G), E_2)$ nor their representations in $\mathcal{R}_1$ and $\mathcal{R}_2$ are a part of G' but these embeddings will solely be used to identify the representations of the vertices in G'.
- The vertices of S are represented in the center of B where the representation of only the unique degree-4 vertex of S passes through the vertex x of B.
- For $v_i \in V(G)$, represent w_i by the set $\mathcal{R} \backslash (\mathcal{R}_1(v_i) \cup \mathcal{R}_2(v_i))$. Note that since $(\mathcal{R}_1(v_i)$ and $\mathcal{R}_2(v_i))$ are intervals, the representation of every w_i passes through x of B.
- For each $\{v_i, v_j\} = e_{ij} \in E_k$, represent w_{ij} by the set $\mathcal{R}_k(v_i) \cap \mathcal{R}_k(v_j)$ for $k \in \{1, 2\}$.

($\Leftarrow$) We show that the bipartite graph G is 2-track if G' is a B-graph. We achieve this by tracing back to G from the obtained B-representation of G' as follows:

- Since G is bipartite, it contains no clique of size more than 2. Then, for any two distinct edges $e_{ij}, e_{kl} \in E(G)$ and for $k \in \{1, 2\}$, $(\mathcal{R}_k(v_i) \cap \mathcal{R}_k(v_j)) \cap (\mathcal{R}_k(v_k) \cap \mathcal{R}_k(v_l)) = \emptyset$. Therefore, the edge vertices in G' form an independent set.

- For $v_i \in V(G)$, the representation of every w_i passes through x of B. Moreover, every vertex in S either passes through x or is connected to a vertex passing through x. Therefore, in the representation $\mathcal{R}$ of G', all intervals not passing through x and not being connected to a vertex passing through x are the edge vertices. The edge vertices on the first circle and the second circle of B will be the edges in E_1 and E_2 of G, and we denote them by V_{E_1} and V_{E_2} in G', respectively.

- Identify S as follows:

  - Since only the unique degree-4 vertex of S and every vertex representing the original vertices of G pass through x, every vertex of $V(G') \setminus (V_{E_1} \cup V_{E_2})$ not passing through x is contained in S. Let $V(S')$ be the set of vertices obtained this way.

  - Recall that to obtain G', every edge between $V(G)$ and $E(S)$ was added. Identify the unique degree-4 vertex of S as the vertex in $V(G') \setminus (V_{E_1} \cup V_{E_2})$ which passes through x and not adjacent to some vertex in $V(S')$.

- Recall that for $v_i \in V(G)$, w_i is represented by the set $\mathcal{R} \setminus (\mathcal{R}_1(v_i) \cup \mathcal{R}_2(v_i))$. For each $w_{ij} \in V_{E_k}$ for $k \in \{1, 2\}$, identify i and j as follows:

  - Take the complement $\overline{\mathcal{R}(w_l)}$ of the representation of every w_l in $V(G') \setminus (V_{E_1} \cup V_{E_2} \cup V(S))$.

  - If there exists a pair (w_l, w_m) such that the intersection $\overline{\mathcal{R}(w_l)} \cap \overline{\mathcal{R}(w_m)}$ on the first circle of B equals $\mathcal{R}(w_{ij})$, then $\{l, m\} \in E(G)$ and $\{l, m\} \in E_1$, and $i = l$ and $j = m$.

  - If there exists a pair (w_l, w_m) such that the intersection $\overline{\mathcal{R}(w_l)} \cap \overline{\mathcal{R}(w_m)}$ on the second circle of B equals $\mathcal{R}(w_{ij})$, then $\{v_, v_m\} \in E(G)$ and $\{v_l, v_m\} \in E_2$, and $i = l$ and $j = m$.

- Finally, since the unions $\bigcup \overline{\mathcal{R}(w_l)}$ on the first circle and on the second

circle of B form interval graphs as no vertex in those unions passes through x of B, G forms a 2-track graph with its partition into E_1 and E_2 analogously to the partitioning of the edge vertices in G' into V_{E_1} and V_{E_2}.

□

8.2 Recognizing L-graphs

In this section, we prove that L-graphs can be recognized in polynomial time. Analogously to Chapter 7, we assume that if the input graph G is an L-graph, then it is not a L'-graph G for some minor or subgraph L' of L since both circular-arc graph recognition [54] and S_3-graph recognition [22] were already shown to be solvable in polynomial time. We obtain our result using three key theorems.

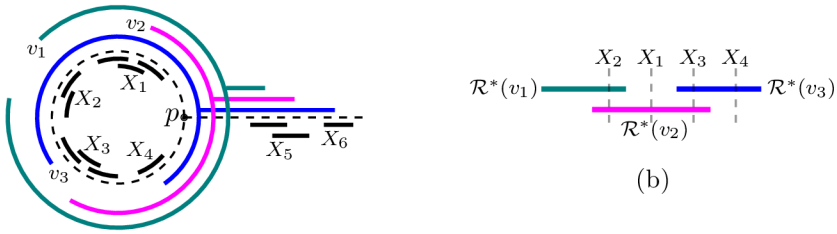


Figure 8.2: For an L-graph G and its clique on the vertices v_1, v_2 and v_3, (a) a (C, I)-centered L-representation of G where all v_1, v_2 and v_3 passing through the intersection point p of the unique circle and the unique stick of L, and (b) the interval graph I on the vertices $\mathcal{R}^*_O(v_1)$, $\mathcal{R}^*_O(v_2)$ and $\mathcal{R}^*_O(v_3)$.

Before we state our key theorems, we define C-centered L-graphs and (C, I)-centered L-graphs. See Figure 8.2 for an illustration of these definitions.

Let G be an L-graph, $\mathcal{R}$ be any L-representation of G, and C be the clique of G placed on the intersection point p of the unique circle and the unique stick of L in $\mathcal{R}$. Then, $\mathcal{R}$ is called a *C-centered* L-representation, and G is called a *C-centered* L-graph.

For a C-centered L-graph G, and $\mathcal{R}$ be any L-representation of G with C on the intersection point p of L in $\mathcal{R}$, and $\mathcal{X}$ be the connected components in $G \setminus C$. With respect to $\mathcal{R}$, let $\mathcal{R}_O$ denote the subdivision of the circle of

L resulting in this L-representation $\mathcal{R}$ of G, and $\mathcal{R}_O(v_i)$ denote the representation of the vertex $v_i \in C$ around the circle of L. Moreover, let $\mathcal{R}^*_O(v_i)$ be the representation consisting of the complement $\overline{\mathcal{R}_O(v_i)}$ of $\mathcal{R}_O(v_i)$ in $\mathcal{R}_O$ and every component $X_j \in \mathcal{X}$ if $\overline{\mathcal{R}_O(v_i)} \cap \mathcal{R}(v_k) \supseteq \mathcal{R}(v_k)$ for some vertex $v_k \in X_j$, i.e. v_i is not adjacent to v_k in G. Finally, let I be the interval graph with the vertex set $C' = \{v_i \in C : \mathcal{R}^*_O(v_i) \neq \emptyset\}$ with the interval representation given by the intervals $\{\mathcal{R}^*_O(v_i) : v_i \in C'\}$. Then, $\mathcal{R}$ is called a (C, I)-*centered* L-representation, and G is called a (C, I)-*centered* L-graph. Here, I can be informally though as "negating" the representations of the vertices of C in the circle of L.

8.2.1 The first key theorem

Our first key theorem is stated as follows.

Theorem 8.2. *Given a graph G, one can in polynomial time either decide that G is an L-graph or output a set $\mathcal{C}$ of cliques of G such that the size of $\mathcal{C}$ is polynomial in $V(G)$ and if G is an L-graph, then G is a C-centered L-graph for some $C \in \mathcal{C}$.*

We first give the following procedure which is used in Theorem 8.2 and then prove its correctness and complexity.

Procedure 8.3. Given a graph G on n vertices, we either decide that G is an L-graph or output the mentioned set $\mathcal{C}$ of cliques in Theorem 8.2 as follows:

1. Let V_O and V_- be initially empty sets of vertices which have to be placed in the circle and which might be placed on the stick of L, respectively, if G is an L-graph.
2. Identify V_O and V_- as follows:
   (a) Use Procedure 7.17 given in Chapter 7 obtaining the sets V_L and V_R, and let $V_O \leftarrow V_L \cup V_R$.
   (b) Let $V_- \leftarrow V(G) \setminus V_O$.
3. Let $\mathcal{C}$ be an initially empty set of candidate cliques of G, and $\mathcal{X}_-$ denote the components in $G[V_-]$ where each of them is a chordal graph.
4. For each $X \in \mathcal{X}_-$, let $\mathcal{C}_X$ denote the set of all maximal cliques of $G[X \cup N_O(X)]$, and add $\mathcal{C}_X$ to $\mathcal{C}$.
5. For each $C \in \mathcal{C}$:

(a) Output C, and let $\mathcal{X}$ denote the components of $G \setminus C$.

(b) For every component $X \in \mathcal{X}$:

i. Let $C' = \{v_i \in C : v_i \text{ has a neighbor in } X\}$. Return that G is an L-graph if $G[C' \cup X]$ is a circular-arc graph with a representation of C' as a Helly clique [9] and $G \setminus X$ forms an interval graph with C as its leftmost clique.

ii. If $G[X]$ forms an interval graph, then for every maximal clique C_X of X, output the clique $C'_X = C' \cup \{v_i \in C_X : C' \subseteq N(v_i)\}$.

Proof of Theorem 8.2. We prove that Procedure 8.3 works correctly and in polynomial time. We first show its correctness.

Given an L-graph G, we first argue G has an L-representation $\mathcal{R}$ such that:

- There exists a clique $C \in \mathcal{C}$ such that C is placed on the stick of L, does not pass through p of L and C is as close to p as possible in $\mathcal{R}$.
- If C' is the clique placed on p in $\mathcal{R}$, then every component $X \in \mathcal{X} = G \setminus C'$ is either placed on the circle or the stick of L, and no component placed on the stick passes through p of L.

Given an L-representation $\mathcal{R}$ of a graph G, we now argue that G is a C-centered L-graph for some $C \in \mathcal{C}$:

- If p is not covered by the representations of the vertices in some $X \in \mathcal{X}$, then $\mathcal{R}$ can be turned into C-centered L-representation of G since C is added to $\mathcal{C}$.
- Otherwise, p is covered by the representations of the vertices in some $X \in \mathcal{X}$. Then, if the procedure returns that G is an L-graph, then X is the only component on the circle of L which forms a circular-arc graph and $G \setminus X$ forms an interval graph with C as its leftmost clique. Otherwise, if two components from $\mathcal{X}$ are represented on the circle of L, then all the components in $\mathcal{X}$ form interval graphs, and $\mathcal{R}$ can be turned into C-centered L-representation of G since C'_X is added to $\mathcal{C}$.

We now prove its complexity. First of all, Procedure 7.17 given in Chapter 7 works correctly and in polynomial time by Lemma 7.18. The number of components in $\mathcal{X}_-$ is bounded by the number of vertices in G, thus at most n. Since each $X \in \mathcal{X}_-$ forms a chordal graph, each of them has linearly many

maximal cliques that can be listed in polynomial time [59]. Therefore, the size of the set $\mathcal{C}$ is at most $\mathcal{O}(n^2)$.

We return that G is an L-graph if $G[C' \cup X]$ is a circular-arc graph with a representation of C' as a Helly clique [9] and $G \setminus X$ forms an interval graph with C as its leftmost clique. To check the former, we use our polynomial time algorithm for the OrderedHellyCliquesCarc problem with $k = 1$ that is defined as follows: Given a circular-arc graph and a set $\mathcal{C} = \{C_1, \ldots, C_k\}$ of cliques of G, the goal is to decide whether there exists a circular-arc representation of G such that all cliques in $\mathcal{C}$ are represented as Helly cliques in the given order. Our preprint [9] proving this result is currently in preparation. To check the latter, we use the linear time algorithm of Booth and Lueker that constructs the PQ-tree of $G \setminus X$ that corresponds to all interval representations of $G \setminus X$ [17]. □

8.2.2 The second key theorem

Assuming that Theorem 8.2 returns a set $\mathcal{C}$ of candidate cliques, it remains to efficiently decide whether G is a C-centered L-graph for some $C \in \mathcal{C}$ which is stated as follows.

Theorem 8.4. *Given a graph G and a clique C of G, one can in polynomial time output a set $\mathcal{I}$ of interval graphs such that the size of $\mathcal{I}$ is polynomial in $V(G)$ and if G is a C-centered L-graph, then G is a (C, I)-centered L-graph for some $I \in \mathcal{I}$.*

Assume a C-centered L-graph G. Let $\mathcal{R}$ be a C-centered L-representation of G, and $\mathcal{X}$ be the set of components in $G \setminus C$ where every $G[X]$ forms an interval graph for $X \in \mathcal{X}$. With respect to $\mathcal{R}$, let $\mathcal{X}_O$ and $\mathcal{X}_-$ be the components of $\mathcal{X}$ which are placed in the circle and the stick of L, respectively.

A component in $\mathcal{X}$	**Its type w.r.t. C**
X_1	ambiguous non-simple
X_2	ambiguous simple
X_3	circle
X_4	ambiguous simple
X_5	ambiguous non-simple
X_6	ambiguous simple

Table 8.1: The partition of the components $X_1, \ldots, X_7$ of the C-centered L-graph G for the clique C on the vertices $\{v_1, v_2, v_3\}$ given in Figure 8.2a.

Given a graph G and a clique C of G, we partition the components in $\mathcal{X} = G \setminus C$. For $X \in \mathcal{X}$:

- If $G[C \cup X]$ does not have an interval representation with C as its leftmost clique, then X is called *circle*. Informally, this means that if G is a C-centered L-graph, then X has to be placed on the circle of L in every C-centered L-representation of G.
- Otherwise, X is called *ambiguous*. Informally, this means that if G is a C-centered L-graph, then X might be placed on the stick of L in some C-centered L-representation of G.

We partition the ambiguous components further as follows:

- If $N(v_i) \cap C = N(v_j) \cap C$ for every $v_i, v_j \in X$, X is called *ambiguous simple*.
- Otherwise, X is called *ambiguous non-simple*.

We denote the sets of circle, ambiguous, ambiguous simple and ambiguous non-simple components by $\mathcal{X}_c$, $\mathcal{X}_a$, $\mathcal{X}_{as}$ and $\mathcal{X}_{ans}$, respectively. Table 8.1 shows the partition of the components of C-centered L-graph G given in Figure 8.2.

Recall that for a (C, I)-centered L-graph, I is informally defined as "negating" the representations of the vertices of C in the circle of L. For $X_i \in \mathcal{X}$, let $C(X_i) = \{v \in C : v \text{ is not adjacent to some vertex in } X_i\}$. We assume that $C(X_i) \neq C(X_j)$ for every two ambiguous simple components $X_i \neq X_j$ since otherwise we may simply consider the input graph G without X_j. We define an order $\subset_t$ of $\mathcal{X}$ with respect to the sets $C(X_i)$ for $X_i \in \mathcal{X}$ and the partitions $\mathcal{X}_c$, $\mathcal{X}_a$, $\mathcal{X}_{as}$ and $\mathcal{X}_{ans}$ where $X_i \subset_t X_j$ if one of the following holds:

- $C(X_i) \subsetneq C(X_j)$, or
- $C(X_i) = C(X_j)$ and $X_i \in \mathcal{X}_c$ and $X_j \in \mathcal{X}_a$, or
- $C(X_i) = C(X_j)$ and $X_i \in \mathcal{X}_{ans}$ and $X_j \in \mathcal{X}_{as}$.

Then, $(\mathcal{X}, \subseteq_t)$ is a poset where $\subseteq_t$ is the reflexive closure of $\subset_t$.

For an element x_i of a poset $(\mathcal{X}, \subseteq_t)$, we define the *downset* of x_i as the set $\{x_j \in \mathcal{X} : x_j \subset x_i\}$. Now, we are ready to state the following lemma which helps us give a procedure to compute the set $\mathcal{I}$ of mentioned in Theorem 8.4.

Lemma 8.5. *Let G be an L-graph, C be a clique of G, $\mathcal{R}$ be a C-centered L-representation of G, $\mathcal{X}$ be the set of components of $G \setminus C$, and with respect*

to $\mathcal{R}$, $\mathcal{X}_- \subseteq \mathcal{X}$ be the components that are placed on the stick of L, and $\mathcal{X}_O = \mathcal{X} \setminus \mathcal{X}_-$ be the components that are placed on the circle of L, respectively. Then, the followings hold:

1. *$\mathcal{X}_- \subseteq \mathcal{X}_a$, and $\mathcal{X}_-$ forms a chain in $(\mathcal{X}, \subseteq_t)$.*
2. *The order of the components in $\mathcal{X}_-$ on the stick of L with respect to their distance from C coincides with the partial order $\subseteq_t$ of $\mathcal{X}_-$.*
3. *The downset of every component in $\mathcal{X}_O$ has width at most 2 in the partial order of $(\mathcal{X}_O, \subseteq_t)$ and at most 3 in the partial order of $(\mathcal{X}, \subseteq_t)$, respectively.*
4. *There are no two circle components $X_i, X_j \in \mathcal{X}_c \subseteq \mathcal{X}_O$ such that $C(X_i) \subseteq C(X_j)$. Therefore, if distinct $X_i, X_j \in \mathcal{X}$ are such that $X_i \in \mathcal{X}_c$ and $C(X_i) \subseteq C(X_j)$, then X_j is placed on the stick of L, thus $X_j \in \mathcal{X}_-$ in $\mathcal{R}$.*

Proof. We prove these items separately.

1. Since the set $\mathcal{X}_c$ consists of all components in $G \setminus C$ that have to be represented on the circle of U, $\mathcal{X}_-$ has to be a subset of $\mathcal{X}_a = \mathcal{X} \setminus \mathcal{X}_c$. To show that $\mathcal{X}_-$ forms a chain in $(\mathcal{X}, \subseteq_t)$, it is enough to show that no two components in $\mathcal{X}_-$ form an antichain. For a contradiction, assume that $X_i \neq X_j \in \mathcal{X}_-$ forms an antichain. Then, C must be placed between X_i and X_j but this means that $\mathcal{R}$ is not a C-centered L-representation of G, or one of X_i and X_j is not placed on the stick of L, thus is not contained in the set $\mathcal{X}_-$. Thus, $\mathcal{X}_-$ forms a chain in $(\mathcal{X}, \subseteq_t)$.

2. Let $\prec_{\mathcal{R}}$ denote the linear order of the components in $\mathcal{X}_-$ on the stick of L with respect to their distance from C. For two components $X_i \neq X_j \in \mathcal{X}_-$, we denote by $X_i \prec_{\mathcal{R}} X_j$ that X_i is closer to C than X_j in $\mathcal{R}$.

Since $\mathcal{X}_-$ forms a chain in $(\mathcal{X}, \subseteq_t)$ by the previous item, they have a fixed linear order in $(\mathcal{X}, \subseteq_t)$ and therefore, in $(\mathcal{X}_-, \subseteq_t)$. Clearly, if $C(X_i) \subsetneq C(X_j)$ for $X_i \neq X_j \in \mathcal{X}_-$, then $X_i \subseteq_t X_j$, and X_i has to be placed closer to C on the stick of L with respect to $\mathcal{R}$, thus $X_i \prec_{\mathcal{R}} X_j$. Otherwise, $C(X_i) = C(X_j)$, $X_i \in \mathcal{X}_{ans}$ and $X_j \in \mathcal{X}_{as}$ as $\mathcal{X}_- \subseteq \mathcal{X}_a$, and therefore $X_i \subseteq_t X_j$. Thus, X_i has to be placed closer to C on the stick of L with respect to $\mathcal{R}$, thus $X_i \prec_{\mathcal{R}} X_j$. Therefore, if $X_i \subseteq_t \ldots \subseteq_t X_j$ is the linear order of $\mathcal{X}_-$, then $X_i \prec_{\mathcal{R}} \ldots \prec_{\mathcal{R}} X_j$ is the order of the components in $\mathcal{X}_-$ on the stick of L in $\mathcal{R}$.

3. Let $\prec_{\mathcal{R}}$ denote the linear order of the components in $\mathcal{X}_O$ on the circle of L with respect to their distance from C when we traverse the circle of L starting

from C in $\mathcal{R}$. For two components $X_i \neq X_j \in \mathcal{X}_O$, we denote by $X_i \prec_{\mathcal{R}} X_j$ that X_i is encountered before X_j when the circle of L is traversed in clockwise order starting from C in $\mathcal{R}$. To prove that the downset of every component $X_i \in \mathcal{X}_O$ has width at most 2 in the partial order of $(\mathcal{X}_O, \subseteq_t)$, we show that for every $X_i \in \mathcal{X}_0$, both sets $\mathcal{X}_i^a = \{X_j \in \mathcal{X}_O : X_j \subseteq_t X_i \text{ and } X_j \prec_{\mathcal{R}} X_i\}$ and $\mathcal{X}_i^b = \{X_j \in \mathcal{X}_O : X_j \subseteq_t X_i \text{ and } X_i \prec_{\mathcal{R}} X_j\}$ form chains in $(\mathcal{X}, \subseteq_t)$.

For $X_i \in \mathcal{X}_0$, let $X_j, X_k \in \mathcal{X}_i^a$ and $X_j \prec_{\mathcal{R}} X_k$. Then, $C(X_j) \subseteq C(X_k) \subseteq C(X_i)$ since $C(X_j) \subseteq C(X_i)$, $C(X_k) \subseteq C(X_i)$, and $X_j \prec_{\mathcal{R}} X_k$. Clearly, if $C(X_j) \subsetneq C(X_k)$, then $X_i \subseteq_t X_j$. Otherwise, $C(X_j) = C(X_k)$. If $X_j \in \mathcal{X}_{ans}$ and $X_k \in \mathcal{X}_{as}$, then clearly $X_j \subseteq_t X_k$. Else, $X_j \in \mathcal{X}_c$ and $X_k \in \mathcal{X}_a$ but in that case, again $X_j \subseteq_t X_k$ since X_j must be placed closer to X_i than X_k. Therefore, $\mathcal{X}_i^a$ forms a chain in the partial order of $(\mathcal{X}_O, \subseteq_t)$. By analogous arguments, we get that $\mathcal{X}_i^b$ forms a chain in the partial order of $(\mathcal{X}_O, \subseteq_t)$.

Finally, since $\mathcal{X}_-$ forms a chain in $(\mathcal{X}, \subseteq_t)$ by item 1, and the downset of every component in $\mathcal{X}_O$ has width at most 2 in the partial order of $(\mathcal{X}_O, \subseteq_t)$, the downset of every component in $\mathcal{X}_O$ has width at most 3 in the partial order of $(\mathcal{X}, \subseteq_t)$.

4. Since $X_i \in \mathcal{X}_c$, $G[C \cup X_i]$ does not have an interval representation with C as its leftmost clique. Then, there exists a vertex $u \in C$ adjacent to a vertex $u_i \in X_i$ but not adjacent to all vertices in X_i, and there exists another vertex $v \in C$ adjacent to a vertex $v_i \in X_i$ other than u_i but not adjacent to all vertices in X_i. This means that $\{u, v\} \subseteq C(X_i)$, and the set $(u \cup V(X_i) \cup v)$ covers the circle of L.

Assume, for a contradiction, that there is some $X_j \in \mathcal{X}_c$ such that $C(X_i) \subseteq C(X_j)$. Since X_i and X_j are distinct components of $G \setminus C$ and the set $(u \cup V(X_i) \cup v)$ covers the circle of L, all the vertices in X_j must intersect u or v, thus $u \notin C(X_j)$ or $v \notin C(X_j)$. If $C(X_i) = C(X_j)$, then $\{u, v\} \subseteq C(X_i) = C(X_j)$, and otherwise, $C(X_i) \subsetneq C(X_j)$, thus, $\{u, v\} \subsetneq C(X_j)$. Both of these are contradicted by that at least one of $u \notin C(X_j)$ and $v \notin C(X_j)$ holds.

This also proves that there are no two distinct circle components $X_i, X_j \in \mathcal{X}_c$ such that $C(X_i) \subseteq C(X_j)$. □

We give the following procedure which is used in Theorem 8.4 and then prove its correctness and complexity.

Procedure 8.6. Given a graph G on n vertices and a clique C of G, we output the mentioned set $\mathcal{I}$ of interval graphs in Theorem 8.4 as follows:

1. Let $\mathcal{X}$ be the set of components in $G \setminus C$, and $\mathcal{I}$ be the initially empty set of interval graphs.
2. For each $X_i \in \mathcal{X}$, identify the set $C(X_i)$.
3. Identify the partition $\{\mathcal{X}_c, \mathcal{X}_a\}$ of $\mathcal{X}$, and the partition $\{\mathcal{X}_{as}, \mathcal{X}_{ans}\}$ of $\mathcal{X}_a$.
4. Let $\mathcal{D}$ be the initially empty set of components that have to be placed on the stick of L in every C-centered representation of G.
5. Looping through each $X_i \in \mathcal{X}_c$ and $X_j \neq X_i \in \mathcal{X}$, if $C(X_i) \subseteq C(X_j)$, then $\mathcal{D} \leftarrow \mathcal{D} \cup X_j$.
6. Construct the partially order set $(\mathcal{X}, \subseteq_t)$.
7. Looping through the elements in $(\mathcal{X}, \subseteq_t)$, if the downset of $X_i \in \mathcal{X}$ has width at least 4 in $(\mathcal{X}, \subseteq_t)$, then $\mathcal{D} \leftarrow \mathcal{D} \cup X_i$.
8. For an antichain $\mathcal{A}$ in $(\mathcal{X}, \subseteq_t)$, let $I(\mathcal{A})$ denote the graph where
   - $V(I(\mathcal{A})) = \bigcup\{C(X_i) : X_i \in \mathcal{A}\}$, and
   - $E(I(\mathcal{A})) = \{\{u, v\} : u, v \in C(X_i) \text{ for some } X_i \in \mathcal{A}\}$.
9. Let $\mathcal{X}' = \mathcal{X} \setminus \mathcal{D}$, and $\mathcal{A}'$ be the maximal elements in $(\mathcal{X}', \subseteq_t)$ which forms an antichain.
10. If $I(\mathcal{A}')$ forms an interval graph, then $\mathcal{I} \leftarrow \mathcal{I} \cup I(\mathcal{A}')$.
11. Looping through each $X_i \in \mathcal{A}' \setminus \mathcal{X}_c$, if X_i is comparable with all components in $\mathcal{D}$, then for each antichain $\mathcal{A}^i$ in the downset of X_i in $(\mathcal{X}, \subseteq_t)$, add $I((\mathcal{A}' \setminus X_i) \cup \mathcal{A}^i)$ to $\mathcal{I}$ if:
    - The set $(\mathcal{A}' \setminus X_i) \cup \mathcal{A}^i$ forms an antichain in $(\mathcal{X}, \subseteq_t)$, and
    - $I((\mathcal{A}' \setminus X_i) \cup \mathcal{A}^i)$ forms an interval graph.
12. Output the set $\mathcal{I}$.

Proof of Theorem 8.4. We prove that Procedure 8.6 works correctly and in polynomial time. We first show its correctness.

The correctness of the steps from 1 to 9 follow from Lemma 8.5. The steps 10 and 11 follow from the fact that since $\mathcal{A}'$ is an antichain and $\mathcal{D}$ forms a chain in $(\mathcal{X}', \subseteq_t)$, their intersection has at most one element of $(\mathcal{X}', \subseteq_t)$ in common. Therefore, considering the antichain $I(\mathcal{A}')$ when their intersection is empty, and considering the antichains $(\mathcal{A}' \setminus X_i) \cup \mathcal{A}^i$ when their intersection is a

single element X_i covers all possible cases. Therefore, for a (C, I)-centered L-graph, the set $\mathcal{I}$ that we output is correct.

We now prove its complexity. The steps from 1 to 9 clearly take $\mathcal{O}(n^2)$ time. Step 10 also takes linear time as interval graph recognition takes linear time [17]. Since no component $X_i \in \mathcal{D}$ is in $\mathcal{X}'$ and thus, X_i is not an element of $\mathcal{A}'$ in step 11, the width of the downset of every component considered in step 11 is at most 3 which means that there are $\mathcal{O}(n^3)$ antichains which can be enumerated in polynomial time. $\square$

8.2.3 The third key theorem

Since Theorem 8.4 returns a set $\mathcal{I}$ of candidate interval graphs, it remains to efficiently decide whether G is a (C, I)-centered L-graph for some $I \in \mathcal{I}$ which is stated as follows.

Theorem 8.7. *Given a graph G, a clique C of G and an interval graph I on a subset C' of C, one can in polynomial time decide if G is a (C, I)-centered L-graph.*

The rest of this subsection is dedicated to prove Theorem 8.7, and we state the claims marked with a star ($\star$) without a proof to avoid many additional technical definitions.

Assume a (C, I)-centered L-graph G, a (C, I)-centered L-representation $\mathcal{R}$ of G, and the set $\mathcal{X}$ of components in $G \setminus C$ where every $G[X]$ forms an interval graph for $X \in \mathcal{X}$. With respect to $\mathcal{R}$, let $\mathcal{X}_O$ and $\mathcal{X}_-$ be the components of $\mathcal{X}$ which are placed in the circle and the stick of L, respectively. Therefore, with respect to $\mathcal{R}$, the components in $\mathcal{X}_O$ and the components in $\mathcal{X}_-$ have a fixed linear order when the circle of L is traversed in clockwise order starting from C and from C to the degree 1 vertex of the stick of L, respectively.

Let the sets $\mathcal{X}_a$, $\mathcal{D}$, $\mathcal{X}'$ and $\mathcal{I}$ be defined as in Procedure 8.6 given the graph G and the clique C. Now, our aim is to find a partition of $\mathcal{X}'$ into $\mathcal{X}'_O$ and $\mathcal{X}'_-$, and a linear order $\prec_{\mathcal{R}}$ of $\mathcal{X}'_O$ such that there exists an C-centered L-representation $\mathcal{R}$ of G in which the components in $\mathcal{X}'_O$ are placed on the circle of L in the order $\prec_{\mathcal{R}}$, and the components in $\mathcal{X}'_-$ are placed on the stick of L.

For $X_i \in \mathcal{X}$, let $C^s(X_i) = \{v \in C : v \text{ is adjacent to no vertex in } X_i\}$. Clearly, for every $X_i \in \mathcal{X}$, $C^s(X_i) \subseteq C(X_i)$. We define an order $\sqsubset$ of $\mathcal{X}_a$ with respect to the sets $C^s(X_i)$ for $X_i \in \mathcal{X}$ where $X_i \sqsubset X_j$ if $C(X_i) \subseteq C^s(X_j)$. Then, $(\mathcal{X}_a, \sqsubseteq)$ is a poset where $\sqsubseteq$ is the reflexive closure of $\sqsubset_t$. To check

whether $\mathcal{X}'_{-}$ can be placed on the stick of L, we simply check whether $(\mathcal{X}'_{-} \cup \mathcal{D})$ forms a chain in $(\mathcal{X}_a, \sqsubseteq)$. The correctness and the sufficiency of this check is analogous to the proof of item 1 in Lemma 8.5.

To check if $\mathcal{X}'_O$ can be placed on the circle of L in the order $\prec$, we need the notion of PQ-trees from Chapter 2 that is introduced in [17] to recognize interval graphs. We call such an order $(\mathcal{X}'_O, \prec)$ *good* for the root $\mathbb{R}$ of the PQ-tree T of I.

Let $\mathcal{L}$ be the set of all leaves in T and $X_i \in \mathcal{X}'$, and assume that the maximal cliques in I appear in the order $\mathbb{L}_1, \ldots, \mathbb{L}_p$. Then, for $\mathbb{L}_i \in \mathcal{L}$, let the following be defined as follows:

- All *inner components*, *inner components* placed on the circle of L and *inner components* placed on the stick of L:
  - $inner(\mathbb{L}_i) = \{X \in \mathcal{X}' : C(X) = V(\mathbb{L}_i)\}$.
  - $inner_O(\mathbb{L}_i) = inner(\mathbb{L}_i) \cap \mathcal{X}_O$ with respect to $\mathcal{R}$.
  - $inner_{-}(\mathbb{L}_i) = inner(\mathbb{L}_i) \cap \mathcal{X}_{-}$ with respect to $\mathcal{R}$.
- The left *zone*, the right *zone* and whole *zone* placed on the circle of L:
  - $zone_O^{left}(\mathbb{L}_i) = \{X_j \in \mathcal{X}'_O : C(X_j) \subsetneq V(\mathbb{L}_i)$ and X_j preceeds every $X_k \in inner(\mathbb{L}_i)\}$ with respect to $\mathcal{R}$.
  - $zone_O^{right}(\mathbb{L}_i) = \{X_j \in \mathcal{X}'_O : C(X_j) \subsetneq V(\mathbb{L}_i)$ and X_j succeeds every $X_k \in inner(\mathbb{L}_i)\}$ with respect to $\mathcal{R}$.
  - $zone_O(\mathbb{L}_i) = zone_O^{left}(\mathbb{L}_i) \cup inner_O(\mathbb{L}_i) \cup zone_O^{right}(\mathbb{L}_i)$ with respect to $\mathcal{R}$.

For an internal node $\mathbb{N}$ of T, let $zone_O(\mathbb{N}) = \bigcup_{\mathbb{L}_i \in \mathcal{L}(\mathbb{N})} zone_O(\mathbb{L}_i)$ where $\mathcal{L}(\mathbb{N})$ is the set of all leaves in $\mathcal{L}$ which descend $\mathbb{N}$ in T.

If $\mathcal{X}'_O$ can be placed on the circle of L in the order $\prec$, we call $(\mathcal{X}'_O, \prec)$ a *good order* of the root $\mathbb{R}$ of T. In addition, for a subset $\mathcal{K} \subseteq \mathcal{X}'_O$, we define a *good order* $(\mathcal{K}, \prec)$ with respect to a node $\mathbb{N}$ of the PQ-tree T and some left and right borders (B_L, B_R) which form cliques in H.

Recall that $\mathbb{L}_1, \ldots, \mathbb{L}_p$ are the leaves of the PQ-tree T of I which correspond to the set of all maximal cliques in I. We assume that $\mathbb{L}_0 = \emptyset$ and $\mathbb{L}_{p+1} = \emptyset$, and then the followings hold for every $i \in \{1, \ldots, p\}$:

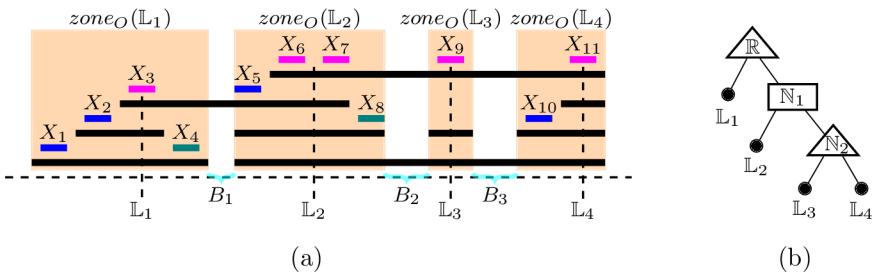


Figure 8.3: An illustration of the zones of the nodes of a PQ-tree. (a) An interval graph I on black intervals with four maximal cliques $\mathbb{L}_1, \ldots, \mathbb{L}_4$, and blue, magenta and teal components $X_1 \ldots, X_{11}$ are the left zone, inner vertices and the right zone, respectively, and orange rectangles are the zones. For instance, $zone_O^{left}(\mathbb{L}_1) = \{X_1, X_2\}$ while $zone_O^{left}(\mathbb{L}_3) = \emptyset$, and $zone_O(\mathbb{N}_2) = zone_O(\mathbb{L}_3) \cup zone_O(\mathbb{L}_4) = \{X_9\} \cup \{X_{10}, X_{11}\} = \{X_9, X_{10}, X_{11}\}$. (b) The PQ-tree of I which has the maximal cliques $\mathbb{L}_1, \ldots, \mathbb{L}_4$ on its leaves.

1. $zone_O^{left}(\mathbb{L}_i)$ forms a chain in $(\mathcal{X}_a, \sqsubseteq)$, $(zone_O^{left}(\mathbb{L}_i), \sqsubset)$ is equal to $(zone_O^{left}(\mathbb{L}_i), \prec)$, and $zone_O^{left}(\mathbb{L}_i)$ *"respects"* the border $\mathbb{L}_{i-1} \cap \mathbb{L}_i$ which means $\mathbb{L}_{i-1} \cap \mathbb{L}_i \sqsubset zone_O^{left}(\mathbb{L}_i)$.

2. $zone_O^{right}(\mathbb{L}_i)$ forms a chain in $(\mathcal{X}_a, \sqsubseteq)$, $(zone_O^{right}(\mathbb{L}_i), \sqsubset)$ is equal to $(zone_O^{right}(\mathbb{L}_i), \prec)$, and $zone_O^{right}(\mathbb{L}_i)$ *"respects"* the border $\mathbb{L}_i \cap \mathbb{L}_{i+1}$ which means $\mathbb{L}_i \cap \mathbb{L}_{i+1} \sqsubset zone_O^{right}(\mathbb{L}_i)$.

The next step is to define when $(\mathcal{K}, \prec)$ is a good order for a node $\mathbb{N}$ of T and the borders (B_L, B_R). Roughly speaking, $(\mathcal{K}, \prec)$ is defined such that it allows to derive an admissible order $\mathbb{L}_1, \ldots, \mathbb{L}_p$ of the leaves in $\mathcal{L}(\mathbb{N})$ and to define the zones for $\mathbb{L}_i$ in $(\mathcal{K}, \prec)$ for every $i \in \{1, \ldots, p\}$. Moreover, it is required that the left zone of $\mathbb{L}_1$ respects the border B_L, the right zone of $\mathbb{L}_p$ respects the border B_R, and the right zone of $\mathbb{L}_i$ and the left zone of $\mathbb{L}_{i+1}$ respect the border $\mathbb{L}_i \cap \mathbb{L}_{i+1}$ for $i \in \{1, \ldots, p-1\}$. For example, in Figure 8.3 (a), we have an interval graph I with its zones $\mathbb{I}_1, \ldots, \mathbb{I}_4$ and their inner nodes identified. The borders B_1, B_2 and B_3 between zones correspond to cliques in I. With respect to the example interval representation of I, let $\prec$ be the order of the components $X_1, \ldots, X_{11}$ from left to right. Then, Table 8.2 shows the good orders of the nodes in T with respect to $\mathcal{R}$ and some borders.

For a node of T and the borders	An example good order
$\mathbb{L}_1$ and $(\emptyset, B_1)$	$(zone_O(\mathbb{L}_1), \prec_{\mathcal{R}})$
$\mathbb{L}_2$ and (B_1, B_2)	$(zone_O(\mathbb{L}_2), \prec_{\mathcal{R}})$
$\mathbb{L}_3$ and (B_2, B_3)	$(zone_O(\mathbb{L}_3), \prec_{\mathcal{R}})$
$\mathbb{L}_4$ and $(B_3, \emptyset)$	$(zone_O(\mathbb{L}_4), \prec_{\mathcal{R}})$
$\mathbb{N}_1$ and $(B_1, \emptyset)$	$(zone_O(\mathbb{N}_1), \prec_{\mathcal{R}})$
$\mathbb{N}_2$ and $(B_2, \emptyset)$	$(zone_O(\mathbb{N}_2), \prec_{\mathcal{R}})$
$\mathbb{R}$ and $(\emptyset, \emptyset)$	$(zone_O(\mathbb{R}), \prec_{\mathcal{R}})$

Table 8.2: Example good orders for the nodes of the PQ-tree given in Figure 8.3(b) with respect to the left to right order of the components $X_1, \ldots, X_{11}$ on the interval graph given in Figure 8.3(a).

Definition 8.8. *Given a graph G, a clique C of G, an interval graph I on a subset C' of C, the sets $\mathcal{X}_a$, $\mathcal{D}$, $\mathcal{X}'$ and $\mathcal{I}$ defined as in Procedure 8.6 for G, C and the components in $G \setminus C$, and the PQ-tree T of I, let $\mathbb{N}$ be a node of T and $\mathcal{X}^* \subseteq \mathcal{X}'$. A triple $(\mathcal{X}'_O, \mathcal{X}'_-, \prec)$ is called a* good triple *for $\mathcal{X}^*$, $\mathbb{N}$ and the borders (B_L, B_R) if all the followings hold:*

1. *The set $\{\mathcal{X}'_O, \mathcal{X}'_-\}$ is a partition of $\mathcal{X}'$.*
2. *$(\mathcal{X}'_O, \prec)$ is a good order for $\mathbb{N}$ and (B_L, B_R).*
3. *$\mathcal{X}'_- \cup \mathcal{D}$ forms a chain in $(\mathcal{X}_a, \sqsubseteq)$.*

Finally, we get the following theorem.

Theorem 8.9. *Let G be a (C, I)-centered L-graph where $\mathcal{X} = G \setminus C$, $\mathcal{D} \subseteq \mathcal{X}$ is the set of components that have to be placed on the stick of L in every C-centered representation of G, and $\mathcal{X}' = \mathcal{X} \setminus \mathcal{D}$. Then, the followings hold:*

1. *For every (C, I)-centered L-representation $\mathcal{R}$ of G, the triple $(\mathcal{X}_O, \mathcal{X}_- \setminus \mathcal{D}, \prec_{\mathcal{R}})$ is good for $\mathcal{X}'$, the root $\mathbb{R}$ of the PQ-tree T of I, and the borders $(B_L, B_R) = (\emptyset, \emptyset)$.*
2. *For every triple $(\mathcal{Y}_O, \mathcal{Y}_-, \prec)$ that is good for $\mathcal{X}'$, the root $\mathbb{R}$ of the PQ-tree T of I, and the borders $(B_L, B_R) = (\emptyset, \emptyset)$, there exists a (C, I)-centered L-representation $\mathcal{R}$ of G such that $(\mathcal{Y}_O, \prec) = (\mathcal{X}_O, \prec_{\mathcal{R}})$ and $\mathcal{Y}_- \cup \mathcal{D} = \mathcal{X}_-$.*

Given the above, our remaining task is to efficiently test whether there exists a good triple for the set for $\mathcal{X}'$ and the root $\mathbb{R}$ of the PQ-tree T of the interval graph I which witnesses that the input graph G has a (C, I)-

centered L-representation. To achieve this, we need the following technical definitions.

For a vertex $v \in V(I)$, let $\ell(v) = \{\mathbb{L} \in \mathcal{L} : v \in V(\mathbb{L})\}$. Then, for each node $\mathbb{N}$ in T, we first define the following four subsets of $V(I)$:

1. The set of vertices in $V(I)$ that are *"universal"* in $\mathbb{N}$:

$$I^{=}(\mathbb{N}) = \{v \in V(I) : \ell(v) = \mathcal{L}(\mathbb{N})\}$$

2. The set of vertices in $V(I)$ that are *"internal"* in $\mathbb{N}$:

$$I^{\subset}(\mathbb{N}) = \{v \in V(I) : \ell(v) \subsetneq \mathcal{L}(\mathbb{N})\}$$

3. The set of vertices in $V(I)$ that are *"contained"* in $\mathbb{N}$:

$$I^{\subseteq}(\mathbb{N}) = I^{=}(\mathbb{N}) \cup I^{\subset}(\mathbb{N})$$

4. The set of vertices in $V(I)$ that *"surpass"* $\mathbb{N}$:

$$I^{\supset}(\mathbb{N}) = \{v \in V(I) : \mathcal{L}(\mathbb{N}) \subsetneq \ell(v)\}$$

For a linear order $(\mathcal{X}, \preceq)$ and $\mathcal{Y} \subseteq \mathcal{X}$, $\mathcal{Y}$ is called an *interval* in $(\mathcal{X}, \preceq)$ if for every X_i, X_j, X_k such that $X_i \prec X_j \prec X_k$ and $X_i, X_k \in \mathcal{Y}$, then $X_j \in \mathcal{Y}$. We call an interval $\mathcal{Y}$ *upper* if $X_j \in \mathcal{Y}$ for every $X_i \prec X_j$ and $X_i \in \mathcal{Y}$.

Using the above, we define the following sets $inner(\mathbb{N}) \subseteq \mathcal{X}'$, $middle(\mathbb{N}) \subseteq \mathcal{X}'$, $outer(\mathbb{N}) \subseteq \mathcal{X}' \cap \mathcal{X}_a$ and $comp(\mathbb{N}) \subseteq \mathcal{X}' \cap \mathcal{X}_a$ for each $\mathbb{N}$ in T as follows:

1. The set of *"inner"* components in $\mathcal{X}'$ for $\mathbb{N}$:

   - If $\mathbb{N} = \mathbb{L}$ where $\mathbb{L}$ is a leaf node in T, this has already been defined as:

   $$inner(\mathbb{L}) = \{X \in \mathcal{X}' : C(X) = V(\mathbb{L})\}$$

   - Otherwise:

   $$inner(\mathbb{N}) = \{X \in \mathcal{X}' : C(X) \cap I^{\subset}(\mathbb{N}) \neq \emptyset \text{ and } I^{=}(\mathbb{N}) \cup I^{\supset}(\mathbb{N}) \subsetneq C(X)\}$$

   By the above definition of inner components, we get that for every pair $\mathbb{N}_i \neq \mathbb{N}_j$ in T:

- ⋆ If $\mathbb{N}_i$ is a descendant of $\mathbb{N}_j$, then $inner(\mathbb{N}_i) \subseteq inner(\mathbb{N}_j)$.
- ⋆ If $\mathbb{N}_i$ and $\mathbb{N}_j$ are incomparable in T, i.e. none of them is a descendant of the other, then $inner(\mathbb{N}_i) \cap inner(\mathbb{N}_j) = \emptyset$.

Consider a (C, I)-centered L-representation $\mathcal{R}$ and the sets of components $\mathcal{X}_-$ and $\mathcal{X}_O$ that are placed on the stick and the circle of L, respectively. Then, we get the followings:

- ⋆ All nodes $\mathbb{N}$ of T such that $inner(\mathbb{N}) \cap \mathcal{X}_- \neq \emptyset$ lie on a single root-to-leaf path in T.
- ⋆ For every internal node $\mathbb{N}$ in T, $inner(\mathbb{N}) \cap \mathcal{X}_O \neq \emptyset$ and $inner(\mathbb{N}) \cap \mathcal{X}_O$ is an interval in $(\mathcal{X}_O, \prec)$.
- ⋆ If $inner(\mathbb{N}) \cap \mathcal{X}_- \neq \emptyset$, then $inner(\mathbb{N}) \cap \mathcal{X}_-$ is an upper interval in the chain $(\mathcal{X}_- \setminus \mathcal{D}, \sqsubseteq)$.

2. The set of *"middle"* components in $\mathcal{X}'$ for $\mathbb{N}$:

   - If $\mathbb{N} = \mathbb{R}$ where $\mathbb{R}$ is the root of T:

   $$middle(\mathbb{N}) = \mathcal{X}' \setminus inner(\mathbb{N})$$

   - Otherwise, if $\mathbb{N}$ is a child of $\mathbb{N}_i$:

   $$middle(\mathbb{N}) = \{X \in inner(\mathbb{N}_i) \setminus inner(\mathbb{N}) : C(X) \cap I^{\subseteq}(\mathbb{N}) \neq \emptyset\}$$

   By the above definition of middle components, we get that:

   - ⋆ For every internal node $\mathbb{N}$ in T and every pair $\mathbb{N}_i \neq \mathbb{N}_j$ of its children, $(inner(\mathbb{N}_i) \cup midlle(\mathbb{N}_i)) \cap (inner(\mathbb{N}_j) \cup midlle(\mathbb{N}_j)) = \emptyset$.
   - ⋆ For any node $\mathbb{N}$ in T and distinct X_i and X_j, if $X_i \in inner(\mathbb{N})$ and $X_j \in middle(\mathbb{N})$ are comparable in $\sqsubseteq$, then $X_j \sqsubset X_i$.

   Consider a (C, I)-centered L-representation $\mathcal{R}$ and the set of components $\mathcal{X}_O$ that are placed on the circle of L with respect to $\mathcal{R}$. Then, the following holds:

   - ⋆ For any node $\mathbb{N}$ in T, $(inner(\mathbb{N}) \cup middle(\mathbb{N})) \cap \mathcal{X}_O$ is an interval in $(\mathcal{X}_O, \prec)$.

3. The set of *"outer"* components in $\mathcal{X}'$ for $\mathbb{N}$:

- If $\mathbb{N} = \mathbb{R}$ where $\mathbb{R}$ is the root of T or $\mathbb{N}$ is a child of a P-node in T:

$$outer(\mathbb{N}) = \emptyset$$

- Otherwise, $\mathbb{N}$ is a Q-node. Let $\mathbb{N}_1, \ldots, \mathbb{N}_s$ be the set of its children which have the admissible orderings $\mathbb{N}_1, \ldots, \mathbb{N}_s$ and its reverse $\mathbb{N}_s, \ldots, \mathbb{N}_1$. Consider the set $rest(\mathbb{N}) = inner(\mathbb{N}) \setminus \bigcup_{i \in \{1,\ldots,s\}} (inner(\mathbb{N}_i) \cup middle(\mathbb{N}_i))$. In contrast to P-nodes, this set might be non-empty, and, it will be used to identify the *"outer"* components for the Q-node $\mathbb{N}$. Consider the following sets:

$$I^{\subset^*}(\mathbb{N}) = I^{\subset}(\mathbb{N}) \setminus \bigcup_{i \in \{1,\ldots,s\}} V(\mathbb{N}_i)$$

$$J(v) = \{j \in \{1, \ldots, s\} : \mathcal{L}(\mathbb{N}_j) \subseteq \ell(v)\} \quad \text{for every } v \in I^{\subset^*}(\mathbb{N})$$

$$J(X) = \bigcap J(v) : v \in C(X) \cap I^{\subset^*}(\mathbb{N}) \quad \text{for every } X \in rest(\mathbb{N})$$

For $X \in rest(\mathbb{N})$, $J(X) = \{j_1, \ldots j_2\}$ is a strict subinterval of $\{1, \ldots, s\}$. We shortly denote $J(X)$ as $[j_1, j_2]$, and define the *"outer"* components for the children $\mathbb{N}_{j_1}$ and $\mathbb{N}_{j_2}$ of $\mathbb{N}$ as follows. If $j_1 = j_2$, then we add X to $outer(\mathbb{N}_{j_1})$. Otherwise, $j_1 < j_2$. In that case, we add X to $outer(\mathbb{N}_{j_1})$ if $\{j_1 - 1, j_1\} \subseteq J(v)$ for no $v \in I^{\subset^*}(\mathbb{N}) \setminus C(X)$, and we add X to $outer(\mathbb{N}_{j_2})$ if $\{j_2, j_2 + 1\} \subseteq J(v)$ for no $v \in I^{\subset^*}(\mathbb{N}) \setminus C(X)$.

By the above definition of outer components, we get that:

- ⋆ Since $[j_1, j_2]$ is a strict subinterval of $\{1, \ldots, s\}$ when $\mathbb{N}$ is a Q-node, X is added to exactly one of $outer(\mathbb{N}_{j_1})$ and $outer(\mathbb{N}_{j_2})$.
- ⋆ Since every node $\mathbb{N}$ in T is either a child of some other node or the root $\mathbb{R}$ of T, every node is assigned a set of "outer" components above.

4. Finally, we define the set $comp(\mathbb{N}) = inner(\mathbb{N}) \cup middle(\mathbb{N}) \cup outer(\mathbb{N})$ for every $\mathbb{N}$.

Now, for every $\mathbb{N}$ in T, all the followings hold:

- ⋆ If $\mathbb{N} = \mathbb{R}$, then $comp(\mathbb{N}) = \mathcal{X}'$ where $\mathbb{R}$ is the root of T.
- ⋆ The set $\{inner(\mathbb{N}), middle(\mathbb{N}), outer(\mathbb{N})\}$ is a partition of $comp(\mathbb{N})$.

- ⋆ If $\mathbb{N}$ is an internal node in T, then $\{comp(\mathbb{N}_1), \ldots, comp(\mathbb{N}_s)\}$ is a partition of $inner(\mathbb{N})$ where $\mathbb{N}_1, \ldots, \mathbb{N}_s$ are the children of $\mathbb{N}$ in T.
- ⋆ Consider a (C, I)-centered L-representation $\mathcal{R}$ and the sets of components $\mathcal{X}_O$ and $\mathcal{X}_-$ that are placed on the circle and the stick of L, respectively. Then, for an internal node $\mathbb{N}$ regardless if it is a P- or a Q-node and its child $\mathbb{N}_i$, $comp(\mathbb{N}_i) \cap \mathcal{X}_O$ is an interval in $(\mathcal{X}_O, \prec)$, and $comp(\mathbb{N}_i) \cap \mathcal{X}_-$ is an interval in $(\mathcal{X}_- \setminus \mathcal{D}, \sqsubseteq)$.

Therefore, our aim is to find the good triples for $comp(\mathbb{N}) = \mathcal{X}'$ where:

1. A good triple for $inner(\mathbb{N})$ consists of good triples for $comp(\mathbb{N}_1)$, ..., $comp(\mathbb{N}_s)$ where $\mathbb{N}_1, \ldots, \mathbb{N}_s$ are the children of $\mathbb{N}$ in T.
2. A good triple for $comp(\mathbb{N})$ consists of a good triple for $inner(\mathbb{N})$, and a 3-chain partition $\{M_L, M_R, M_D\}$ of $middle(\mathbb{N}) \cup outer(\mathbb{N})$ such that $M_D \cup \mathcal{D}$ can be placed on the stick of L, M_L is to the left of $inner(\mathbb{N})$, and M_R is to the right of $inner(\mathbb{N})$ on the circle of L.

Finally, to decide whether an input graph G is a (C, I)-centered L-graph, our final task is to decide whether there exists a good triple for the root $\mathbb{R}$ of the root of the PQ-tree T of I and the set $comp(\mathbb{R}) = \mathcal{X}'$ as suggested by Theorem 8.9. To achieve this, we first compute the set $comp(\mathbb{N})$ for each node (N) in T which can be easily done in polynomial time. Then, we compute the polynomially many *"fingerprints"* of the good triples for $comp(\mathbb{N})$ using dynamic programming since there can be exponentially many good triples for $comp(\mathbb{N})$ specifically when $\mathbb{N}$ is an internal node of T. We skip the definition of fingerprints to avoid further technical notions, mention that they capture the *"traces"* of good triples, and note that the number of fingerprints of $comp(\mathbb{N})$ is bounded by $\mathcal{O}(n^6)$. To test whether there exists a 3-chain-partition of $middle(\mathbb{N}) \cup outer(\mathbb{N})$ with the computed fingerprints, we reduce to s-list coloring problem for co-comparability graphs for $s = 3$.

A *comparability graph* is an undirected graph that is obtained by replacing every directed edge in a poset with an undirected edge. The *co-comparability graph* of a comparability graph G is the complement of G. Given a graph G and a number s, a set S of colors with $|S| \leq s$, and a list $L(v) \subseteq S$ of colors for each vertex $v \in V(G)$, *s-list coloring* is the problem to decide whether there exists a coloring $f : V(G) \to S$ such that for every $v \in V(G)$, $f(v) \in L(v)$ and for every edge $\{u, v\} \in E(G)$, $f(u) \neq f(v)$. It was shown that that problem can be solved in $\mathcal{O}(n^{s^2+1}s^3)$ in [23].

We reduce 3-chain-partition $\{M_L, M_R, M_D\}$ of $middle(\mathbb{N}) \cup outer(\mathbb{N})$ to 3-

list coloring of the poset $(middle(\mathbb{N}) \cup outer(\mathbb{N}) \cup \mathcal{D}, \sqsubseteq)$ with a list of colors $L(X) \subseteq \{M_L, M_R, M_D\}$ for each component X since partitioning a poset into chains is equivalent to coloring its co-comparability graph.

8.3 Recognizing strongly cyclic U-graphs and Helly $\mathcal{U}$-graphs

In this section, we prove that strongly cyclic U-graphs can be recognized in XP-time where a *strongly cyclic* graph is a graph that contains a hole. Then, a *strongly cyclic U-graph* is a U-graph which is not chordal. As in Chapter 7, we assume that the input graph G is connected, and is not a proper U^*-graph for some subgraph or minor U^* of U. We first give the following procedure to recognize connected strongly cyclic U-graphs, and then prove its correctness and complexity. We then show that Helly $\mathcal{U}$-graphs can be recognized in polynomial time where $\mathcal{U}$ is the class of all unicyclic graphs.

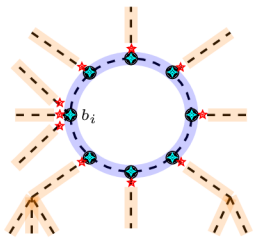


Figure 8.4: The sets V_O and V_T in Procedure 8.10 are shown with blue and orange components, respectively. Each red star corresponds to the maximal clique C_e of $\mathcal{C}_T$ that is assigned to the edge e of U. Each black dot on the circle of U correspond some clique C of $G[V_O]$ that is placed on the branching node b of U. Each cyan star inside a clique C is a subset of C and must be represented as a Helly clique, and also is a subset of the union of every C_e which is assigned on an edge e incident to b. For example, for the node b_i, the cyan clique C' must be Helly, it is a subset of the clique shown by a black dot, and if C_{b_i} is the union of all three cliques shown by red stars on the left of b_i, then C' is a subset of C_{b_i}.

Procedure 8.10. Given a connected strongly cyclic graph G on n vertices and a fixed unicyclic graph U, we decide whether G is a U-graph as follows:

1. Let $\mathcal{B}$ be the set branching nodes on the circle of U.
2. For each $b_i \in \mathcal{B}$, let E_i be the set of all edges e in $E(U)$ that is incident

to b_i and a node $v \in V(U) \setminus \mathcal{B}$.

3. Let $E = \bigcup_{b_i \in \mathcal{B}} E_i$.
4. For each $b_i \in \mathcal{B}$, let $\mathcal{T}_i$ be the forest obtained by subdividing every $e \in E_i$ with a new vertex v_e obtaining the graph U', and then removing every e from U'.
5. Let T_e be the tree in $\mathcal{T}_i$ obtained as above by removing the edge e.
6. Let $\mathcal{T} = \bigcup_{b_i \in \mathcal{B}} \mathcal{T}_i$.
7. Using Procedure 7.17 given in Chapter 7, compute the sets V_O and V_T of vertices of G which have to be placed in the circle and which might be placed on the sticks and trees of U, respectively, if G is a U-graph.
8. Let $\mathcal{X}_T$ be the set of connected components in $G[V_T]$.
9. For each $X \in \mathcal{X}_T$, let $N_O(X) = N(X) \cap V_O$, and let $N_O(\mathcal{X}_T) = \bigcup_{X \in \mathcal{X}_T} N_O(X)$.
10. Let $\mathcal{C}_T$ be the set of all maximal cliques in $G[\mathcal{X}_T \cup N_O(\mathcal{X}_T)]$.
11. Looping through each assignment $f : \mathcal{C}_T \to E$:
    (a) Let $C_e \in \mathcal{C}_T$ be the maximal clique assigned to the edge $e \in E$ with respect to f, and $C'_e = C_e \cap V_O$.
    (b) For each $b_i \in \mathcal{B}$, let $\mathcal{C}_{b_i} = \bigcup_{e \in E_i} C_e$ and $\mathcal{C}'_{b_i} = \bigcup_{e \in E_i} C'_e$.
    (c) Finally, let $G_O = (G \setminus \mathcal{C}_{b_i}) \cup \mathcal{C}'_{b_i}$.
    (d) If G_O is a circular-arc graph where all cliques in $\mathcal{C}'_{b_i}$ are Helly, and $G \setminus G_O$ is a $\mathcal{T}$-graph where each maximal clique C_e appear as a leaf clique in the corresponding T_e-graph, we return that G is a U-graph.
12. Otherwise, we return that G is not a U-graph.

Lemma 8.11. *Let the sets V_O, V_T, $\mathcal{X}_T$ and $N_O(X)$ be as in Procedure 8.10 for $X \in \mathcal{X}_T$. Then, $N_O(X)$ is a clique in $G[V_O]$ and $G[X \cup N_O(X)]$ is chordal.*

Proof. Firstly, assume for a contradiction that $N_O(X)$ is not a clique. Then, it contains two vertices u and v that are not adjacent to each other. Since both $u, v \in N_O(X)$, there exist two vertices $u_i, v_i \in X$ such that u_i is adjacent to u, and v_i is adjacent to v. Since X is connected, u_i and v_i are

connected, and there exists a shortest path P from u_i to v_i which forms an induced path. Now, if u_i is also adjacent to v, $\{u, u_i, v\}$ forms an induced path of length 2 in G, and u_i would be contained in V_O by Procedure 7.17 which is a contradiction. Otherwise, $\{u, u_i, P, v_i, v\}$ forms an induced path of length at least 4 in G, and both u_i and v_i would be contained in V_O by Procedure 7.17 which is a contradiction. Therefore, $N_O(X)$ is a clique.

Now, assume for a contradiction that $G[X \cup N_O(X)]$ is not chordal. Then, $G[X \cup N_O(X)]$ contains a hole L where $V(L) \in V_O$. Since X is chordal, $L \notin X$, and since $X_O \cap X_T = \emptyset$, $L \in N_O(X)$. However, as $N_O(X)$ is a clique, it does not contain any hole L. Therefore, $G[X \cup N_O(X)]$ is chordal. □

Theorem 8.12. *Given a connected strongly cyclic graph G and a fixed unicyclic graph U, one can decide whether G is a U-graph in* `XP`*-time parameterized by $|U|$.*

Proof. Analogously to Section 8.2.1, using the polynomial time Procedure 7.17 given in Chapter 7, we compute the sets V_O and V_T of vertices which have to be placed in the circle and which might be placed on the sticks and trees of U, respectively, if G is a U-graph. Let $\mathcal{X}_T$ be the set of components in $G[V_T]$, and $\mathcal{C}_T$ be the set of all maximal cliques in $G[V_T]$. Since $G[V_T]$ is chordal as it contains no hole, chordal graphs have linearly many maximal cliques which can be listed in linear time [59] and the number of components is at most $\mathcal{O}(n)$ if G is of order n, we can compute both sets $\mathcal{X}_T$ and $\mathcal{C}_T$ in polynomial time.

Now, if G is a U-graph, then $G[V_O]$ must be a circular-arc graph where $N_O(X_i)$ for every $X_i \in \mathcal{X}_T$ is a Helly clique, and every X_i must be placed on a stick or a tree of U attached to the node b_j of the circle where $N_O(X_i)$ is represented as a Helly clique. To test the first condition that G_O must be a circular-arc graph where all cliques in $\mathcal{C}'_{b_i}$ are Helly, we use the `XP`-algorithm of [28] where the parameter is the number $\mathcal{B}$ of branching nodes of B. To test the second condition that $G \setminus G_O$ must be a $\mathcal{T}$-graph where each maximal clique C_e appear as a leaf clique in the corresponding T_e-graph, we used the `XP`-algorithm of [23] where the parameter is the size of the $\mathcal{T}$ which is less than $|U|$. □

We finally move on to Helly $\mathcal{U}$-graphs and give the following.

Theorem 8.13. *Helly $\mathcal{U}$-graphs can be recognized in polynomial time where $\mathcal{U}$ is the class of all unicyclic graphs.*

Proof. For an input graph G, let the sets V_O and V_T of G be found using Procedure 7.17 given in Chapter 7. Recall that V_O and V_T are the sets of vertices of G which have to be placed in the circle and which might be placed on the sticks and trees of U, respectively, if G is a U-graph.

We first claim that G is a Helly $\mathcal{U}$-graph if and only if $G[V_O]$ is a Helly circular-arc graph. Assume that G is a Helly $\mathcal{U}$-graph. Since every vertex in V_O is placed on the unique circle of some $U \in \mathcal{U}$, $G[V_O]$ forms a Helly circular-arc graph. Now, assume that $G[V_O]$ is a Helly circular-arc graph. Then, $G[V_O]$ has a circular-arc representation such that every (maximal) clique in $G[V_O]$ is a Helly clique. Since $G[V_T]$ is chordal, G has a Helly U-representation for some $U \in \mathcal{U}$, thus a Helly $\mathcal{U}$-graph. Therefore, our claim holds.

Finally, since Helly circular-arc graph recognition takes linear time [48], and the sets V_O and V_T of G can be identified in polynomial time by Lemma 7.18, Helly $\mathcal{U}$-graphs can be recognized in polynomial time. $\square$

Chapter 9

A Purely Combinatorial Reduction for Set-Family Automorphisms

In this chapter, we add a rather simple and fully combinatorial polynomial time reduction from the automorphisms of set families of bounded antichain size to the automorphisms of posets of bounded width. This brings us one step closer to the computation of the latter in FPT-time without utilizing the group-theoretical tools that were introduced before, and it is a possible interesting result on its own. Our result is summarized as follows:

Theorem 9.1 (A reduction from set-family automorphism to colored poset automorphism problems)**.** *Let $\mathcal{U}$ be a finite set family of width d, i.e., $\mathcal{U}$ contains no $d+1$ sets pairwise incomparable by inclusion. One can in polynomial time compute a colored poset $(P_{\mathcal{U}}, \preceq)$ of width at most $d + d^d$, and $X \subseteq P_{\mathcal{U}}$, such that the automorphism group $\mathrm{Aut}(\mathcal{U})$ of colored $\mathcal{U}$ equals the restriction of the automorphism group $\mathrm{Aut}(P_{\mathcal{U}})$ of colored $P_{\mathcal{U}}$ to the subset X.*

Recall the definition of bounded color multiplicity graphs and how they relate to posets of bounded width from Chapter 2. The above theorem will then yield to the following.

Corollary 9.2. *Given a poset $(P, \preceq)$ of width d, and a coloring $c : P \to \mathbb{N}$, the (color-preserving) automorphism group of $(P, \preceq)$ can be computed in FPT-time parameterized by d. Note that c is not required to give bounded multiplicity color classes.*

Let $\mathcal{U} \subseteq 2^S$ be a set family of maximum antichain size d over a finite ground set S (and possibly a a coloring $c : \mathcal{U} \to \mathbb{N}$). Recall from Chapter 3 that the cardinality Venn diagram of $\mathcal{U}$ is the integer vector $\big(\ell_{\mathcal{U},\mathcal{U}_1} : \emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}\big)$ such that $\ell_{\mathcal{U},\mathcal{U}_1} := |L_{\mathcal{U},\mathcal{U}_1}|$ where $L_{\mathcal{U},\mathcal{U}_1} = \bigcap_{A \in \mathcal{U}_1} A \setminus \bigcup_{B \in \mathcal{U} \setminus \mathcal{U}_1} B$. Here, we shortly refer to $L_{\mathcal{U},\mathcal{U}_1}$ as $L_{\mathcal{U}_1}$ (and therefore, $\ell_{\mathcal{U},\mathcal{U}_1}$ as $\ell_{\mathcal{U}_1}$) since we will not consider any $L_{\mathcal{U}',\mathcal{U}_1}$ for $\mathcal{U}' \neq \mathcal{U}$. By Lemma 4.5, we have that a bijection $f : \mathcal{U} \to \mathcal{U}$ is an automorphism of $\mathcal{U}$ if and only if $\ell_{\mathcal{U}_1} = \ell_{\varrho(\mathcal{U}_1)}$ holds for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$.

Therefore, to prove Theorem 9.1, we observe the cardinalities of the Venn diagram cells of $\mathcal{U}$, and we encode these cardinalities with colors of the poset $P_{\mathcal{U}}$ sought in the reduction. However, we have to be careful with such an encoding in order to keep $P_{\mathcal{U}}$ of bounded width with respect to the maximum antichain size d of $\mathcal{U}$. That is why we proceed in the following slightly complicated way.

Let $\mathcal{V} \subseteq 2^S$ denote the family of all nonempty (and pairwise disjoint) Venn diagram cells of $\mathcal{U}$, except "the outer" $S \setminus \bigcup \mathcal{U}$. Formally:

$$\mathcal{V} := \{L_{\mathcal{U}_1} : \emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}\} \setminus \{\emptyset\}$$

Note that for each $V \in \mathcal{V}$ where $V = L_{\mathcal{U}_1}$, the family $\mathcal{U}_1$ consists of precisely those sets of $\mathcal{U}$ which contain the cell V.

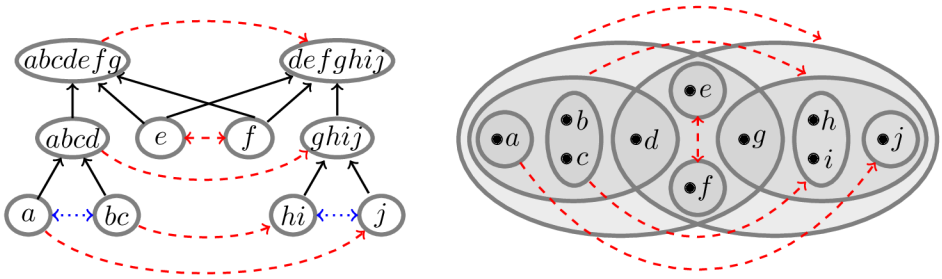
Let $\mathcal{U}_2 \subseteq \mathcal{U}_1$ be the subfamily of inclusion-minimal sets of $\mathcal{U}_1$. Therefore, $|\mathcal{U}_2| \leq d$ by the maximum antichain size of $\mathcal{U}$. For $A \in \mathcal{U}_2$, let the number of sets in $\mathcal{U}$ containing A be defined as:

$$n_A := |\{X \in \mathcal{U} : X \supseteq A\}|$$

Since $\mathcal{U}_2$ is determined by $\mathcal{U}_1$, and $\mathcal{U}_1$ in turn is determined by $V \in \mathcal{V}$, we may correctly define $N[V] := \{n_A : A \in \mathcal{U}_2\}$ as a multiset of integers. We comment that the multisets $N[V]$ are going to be used next to "classify" all nonempty cells $V \in \mathcal{V}$ into groups sharing an equal value of $N[V]$, and we will later show that the size of each group will be at most d^d.

This "classification" of the cells of $\mathcal{V}$ is formalized in the following way. Let p be the number of distinct multisets $N[V]$ over all $V \in \mathcal{V}$. We define pairwise disjoint subfamilies $\mathcal{V}_i \subseteq \mathcal{V}$, where $i \in \{1, \ldots, p\}$, such that $\mathcal{V} = \mathcal{V}_1 \cup \ldots, \mathcal{V}_p$ and the following holds: for $V \in \mathcal{V}_i$ and $V' \in \mathcal{V}_{i'}$, we have $N[V] = N[V']$ if and only if $i = i'$. Based on this, we define a new set family $\mathcal{Y} \subseteq 2^S$ consisting of the following sets, as in Figure 9.2:

- for $i = 1, \ldots, p$, the sets $Y_i := \bigcup_{j=1}^{i} \bigcup_{V \in \mathcal{V}_i} V \in \mathcal{Y}$, and $Y_0 = \emptyset$, and



(a) P

(b) $\mathcal{U}$

Figure 9.1: An illustration of the difference among poset and set-family automorphisms that map (poset) elements and sets to each other, respectively. In (a), the letters, circles and directed straight lines correspond to the ground set elements, (poset) elements and partial order, respectively. In (b), the dots (also marked with the letters) and ellipses correspond to the ground set elements and sets, respectively. To obtain an automorphism, mapping of the colored unidirectional arrows must be applied together while the mapping of colored bi-directional arrows can be applied individually.

- for $i = 1, \ldots, p$ and $j = 1, \ldots, q_i = |\mathcal{V}_i|$, where $\mathcal{V}_i = \{V_1^i, \ldots, V_{q_i}^i\}$, the sets $Z_j^i := Y_{i-1} \cup V_j^i \in \mathcal{Y}$.

We furthermore color each set Y_i by color $c(Y_i) := 0$, and each set Z_j^i by color $c(Z_j^i) := |V_j^i|$. If $\mathcal{U}$ was colored as well, we redefine the color sets of $\mathcal{U}$ and of $\mathcal{Y}$ to be disjoint. The following lemma refines Theorem 9.1:

Lemma 9.3. *Let the (colored) set family $\mathcal{U}$ and the colored set family $\mathcal{Y}$ be as above, and $\mathcal{W} := \mathcal{U} \uplus \mathcal{Y}$ be their disjoint union. Let $P_{\mathcal{U}} := (\mathcal{W}, \subseteq)$ be the poset defined on $\mathcal{W}$ by inclusion. Then:*

1) The maximum antichain size of the set family $\mathcal{W}$ is at most $d + d^d$.

2) A bijection $f : \mathcal{U} \to \mathcal{U}$ is an automorphism of the (colored) set family $\mathcal{U}$, if and only if there exists a bijection $f^+ : \mathcal{W} \to \mathcal{W}$ such that the restriction of f^+ is equal to f, and f^+ is an automorphism of the colored poset $P_{\mathcal{U}}$.

Proof. 1) If $\mathcal{X} \subseteq \mathcal{W}$ is an antichain, then $|\mathcal{X} \cap \mathcal{U}| \leq d$ since the maximum antichain size of $\mathcal{U}$ is at most d. Let us examine $\mathcal{X} \cap \mathcal{Y}$. It follows from the definition that for $i > i'$ and any j we have $Z_j^{i'} \supseteq Y_i \supseteq Z_j^i$, and so $\mathcal{X} \cap \mathcal{Y} \subseteq \mathcal{V}_i$ for some $i \in \{1, \ldots, p\}$. Hence it is enough to prove $|\mathcal{V}_i| = q_i \leq d^d$ for all

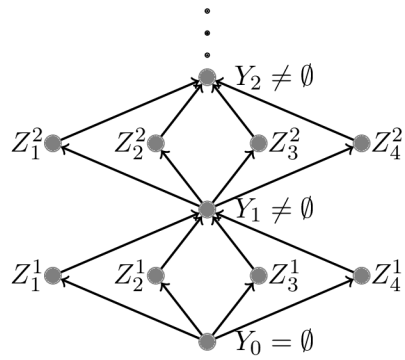


Figure 9.2: An illustration of the inclusion diagram of the set family $\mathcal{Y}$ defined for Lemma 9.3.

$i \in \{1, \ldots, p\}$.

Since the family $\mathcal{U}$ has maximum antichain size $\leq d$, by Dilworth's theorem we can partition $\mathcal{U} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_d$ where each $\mathcal{C}_j$, $j = 1, \ldots, d$, is a chain in inclusion. For each $V \in \mathcal{V}_i$ such that $V = L_{\mathcal{U}_1}$ for (unique) $\mathcal{U}_1 \subseteq \mathcal{U}$, we now denote by $\mathcal{M}[V]$ the subfamily $\mathcal{U}_2 \subseteq \mathcal{U}_1$ of the inclusion-minimal sets of $\mathcal{U}_1$, and we write down these sets as $M[V] = \mathcal{U}_2 = (M_1, \ldots, M_b)$ ($b = |N[V]| \leq d$) ordered such that $n_{M_1} \leq \ldots \leq n_{M_b}$. Then, we define a mapping $s_V : \{1, \ldots, d\} \to \{1, \ldots, d\}$ as follows; $s_V(a) = b$ iff $M_a \in \mathcal{C}_b$. If $|\mathcal{V}_i| > d^d$, then we find two distinct cells $V, T \in \mathcal{V}_i$ such that $s_V \equiv s_T$. So, for $M[V] = (M_1, \ldots, M_b)$ and $M[T] = (M'_1, \ldots, M'_b)$, and for any $j \in \{1, \ldots, b\}$, we have that, at the same time, $M_j, M'_j \in \mathcal{C}_{s_V(j)}$ and $n_{M_j} = n_{M'_j}$. By the above definition of n_A, the latter implies that $M_j = M'_j$, and since this holds for all j, we necessarily conclude $V = T$. This is a contradiction. Therefore, $|\mathcal{V}_i| \leq d^d$ and we are finished with the combined bound of $d + d^d$.

2) If f is an automorphism of the (colored) set family $\mathcal{U}$, then f trivially induces a cardinality-preserving bijection $g : \mathcal{V} \to \mathcal{V}$ on the Venn diagram cells. Since g must preserve our "classification" values $N[V] = N[g(V)]$ for $V \in \mathcal{V}$, and so g respects the subfamilies $\mathcal{V}_1, \ldots, \mathcal{V}_p$, we get that g in turn defines a color-preserving bijection $g' : \mathcal{Y} \to \mathcal{Y}$. So, $f^+ := f \cup g'$ naturally is an automorphism of the colored poset $P_{\mathcal{U}} = (\mathcal{W}, \subseteq)$.

Conversely, assume that f^+ is an automorphism of the colored poset $P_{\mathcal{U}}$, and set f is the restriction of f^+ to $\mathcal{U}$. Then, since the color sets on $\mathcal{U}$ and $\mathcal{Y}$ are disjoint, $f : \mathcal{U} \to \mathcal{U}$ is a bijection, and f preserves the inclusion relation

on $\mathcal{U}$ by the subposet $(\mathcal{U}, \subseteq)$ of $P_{\mathcal{U}}$. By Lemma 4.5, it suffices to verify that f preserves the cardinalities of the Venn diagram cells, i.e., of the sets from $\mathcal{V}$ as defined by $L_{\mathcal{U}_1} = V \in \mathcal{V}$. Observe that, necessarily, $f^+(Y_i) = Y_i$ for $i = 0, 1, \ldots, p$, and so $f(Z_j^i) = Z_{j'}^i$ for each j and some corresponding j'. Moreover, since f preserves the colors on $\mathcal{V}$, $c(Z_j^i) = c(Z_{j'}^i)$, which means that $|V_j^i| = |(Z_j^i \setminus Y_{i-1}| = |(Z_{j'}^i \setminus Y_{i-1}| = |V_{j'}^i|$. Since this holds for every nonempty cell $V_j^i \in \mathcal{V}$, by Lemma 4.5 we get that f is an automorphism of the set family $\mathcal{U}$. $\square$

Since the construction of the family $\mathcal{W}$ of Lemma 9.3 is easily done in polynomial time, we have also finished the proof of Theorem 9.1.

Chapter 10

Conclusions

In this chapter, we first state the future work which is in preparation and open questions that we would like to consider in the future. Then, we conclude this chapter with author's publication.

10.1 Future work and open problems

We first start with the current future work.

In Chapter 6, we showed that the isomorphism problem for T-graphs can be solved in FPT-time parameterized by $|T|$, this result extents to an FPT-approach for chordal graphs with bounded leafage ℓ parameterized by ℓ, and the isomorphism problem is GI-hard already for vertex leafage 2 graphs. The first result was published in [11], and its journal version containing latter two results and our FPT-approach for set family automorphisms which uses group theory only as a black-box (introduced in Chapter 9) is in preparation.

In Chapter 8, we showed that the recognition of L-graphs can be solved in polynomial time, and used our polynomial time algorithm to solve OrderedHellyCliquesCarc problem that was also defined in that chapter. Therefore, our second future work is to complete our manuscript on this result which is in preparation [9].

We now continue with the remaining open problems.

In Chapter 3, we showed that the automorphism group of a (colored and uncolored) set family can be solved in FPT-time using Babai's group theoretical machinery parameterized by the maximum antichain size among the input

set family. A prior approach to an equivalent problem [3] (as mentioned in the introduction) utilizes heavy group machinery, too.

In Chapters 4, 6 and 7, we applied that result to S_d-graphs, T-graphs, and proper U-graphs, respectively, and obtained FPT-algorithms for the same problem. In Chapter 9, we gave a fully combinatorial polynomial time reduction from the automorphisms of set families to the automorphisms of posets which brings us one step closer to the computation of the latter in FPT without utilizing group-theoretical tools. Therefore, we first ask the following:

Open problem 10.1. *Can the automorphism group of posets of bounded width be computed by a fully combinatorial FPT-approach?*

In Chapter 6, we gave a polynomial time approach to canonically decompose T-graphs into levels where each level contains at most $|T|$ components. This canonical decomposition and our FPT-approach from Chapter 3 helped us compute the automorphism group of a T-graph in FPT-time parameterized by $|T|$. On the other hand, the recognition problem for T-graphs is NP-hard [42], and Chaplick et al. gave an XP-approach to recognize them parameterized by $|T|$ [22]. Therefore, we give the following two questions:

Open problem 10.2. *Can the polynomial time canonical decomposition method given in Chapter 6 be used to recognize T-graphs in FPT-time parameterized by $|T|$?*

Open problem 10.3. *Can T-graphs be recognized in FPT-time parameterized by $|T|$, or is it W[1]-hard?*

In Chapter 3, we gave an FPT-approach to compute the automorphism group of a colored clique family of an interval graph which results from the automorphisms of the underlying interval graph. We used this result to prove that isomorphism of T-graphs can be tested in FPT-time parameterized by $|T|$. Considering (proper) U-graphs from Chapter 7 and Chapter 8, we give the following problem:

Open problem 10.4. *Can the automorphism group of a colored clique family of a (proper) circular-arc graph which results from the automorphisms of the underlying (proper) circular-arc graph be computed in FPT- or XP-time?*

In Chapter 8, we showed that L-graphs can be recognized in polynomial time using a very involved approach, and for a fixed unicyclic graph U,

strongly cyclic U-graphs can be recognized in XP-time parameterized by $|U|$. Therefore, we first give the following:

Open problem 10.5. *Can L-graphs be recognized in polynomial time using a simpler approach?*

Since both K_3-graph (i.e., circular-arc graph) isomorphism and S_3-graph isomorphism can be solved in polynomial time, we give the following as the fixed L is the minimal graph that contains both K_3 and S_3:

Open problem 10.6. *Can L-graph isomorphism be tested in polynomial time?*

Chapter 7, we showed how to recognize proper chordal U-graphs in FPT-time, and in Chapter 8, we showed how to recognize strongly cyclic (i.e., non-chordal) U-graphs in XP-time, both parameterized by $|U|$. Considering general U-graphs that are not strongly cyclic, we give the following:

Open problem 10.7. *Can chordal U-graphs be recognized in XP-time parameterized by $|U|$?*

Finally, since circular-arc graph isomorphism can be tested in polynomial time, T-graph isomorphism and proper U-graph isomorphism can be tested in FPT-time parameterized by $|T|$ and $|U|$, respectively, we ask the following:

Open problem 10.8. *Can U-graphs be tested for isomorphism in FPT- or XP-time parameterized by $|U|$?*

If we can answer Open problem 10.4 positively, then we would like to consider the following:

Open problem 10.9. *Can the automorphism group of a U-graph which is not a T-graph for some tree T be computed in FPT- or XP-time parameterized by $|U|$?*

10.2 Author's publications

Paper	Summary of the paper
[5] Isomorphism problem for S_d-graphs Appeared in *MFCS 2020*	We showed that S_d-graph isomorphism can be solved in `FPT`-time parameterized by d.
[13] Unit disk visibility graphs Appeared in *EUROCOMB 2021*	We introduced unit disk visibility graphs, and showed that 3-coloring problem is NP-complete for unit disk segment visibility graphs and unit disk visibility graphs of polygons with holes.
[10] Efficient isomorphism for S_d-graphs and T-graphs Appeared in *ALGORITHMICA 2022*	In addition to our results from [5], we showed that T-graph isomorphism can be solved in `XP`-time parameterized by $\|T\|$ for a fixed tree T, and proper S_d- and proper T-graph isomorphisms can be solved in `FPT`-time parameterized by d and $\|T\|$, respectively.
[11] Isomorphism testing for T-graphs in FPT Appeared in *WALCOM 2022*	We showed that T-graph isomorphism can be solved in `FPT`-time parameterized by $\|T\|$, for a fixed tree T.
[12] Recognition and isomorphism of proper U-graphs in FPT-time Submitted to *WALCOM 2024*	We showed that proper U-graph recognition and isomorphism can be solved in `FPT`-time parameterized by $\|U\|$, for a fixed unicyclic graph U. We also showed that (proper) $\mathcal{U}$-graph recognition is `NP`-complete where $\mathcal{U}$ is the class of all unicyclic graphs, and (proper) H-graph isomorphism is `GI`-complete when H contains two cycles sharing a vertex.
[8] Recognizing H-graphs - Beyond circular-arc graphs Appeared in *MFCS 2023*	We showed that H-graph recognition is `NP`-complete when H contains two cycles sharing a vertex. We then proved that L-graphs can be recognized in polynomial time where L is the graph consisting of one cycle and an edge attached to it. We also showed that, for a fixed unicyclic graph U, U-graph recognition is in `XP` parameterized by $\|U\|$ when the input graph is strictly cyclic, and Helly $\mathcal{U}$-graph recognition can be solved in polynomial time where $\mathcal{U}$ is the class of all unicyclic graphs.

Bibliography

[1] D. Ağaoğlu. *Parameterized Algorithms for Geometric Intersection Graphs*. PhD dissertation proposal, Masaryk University, 2021.

[2] D. Ağaoğlu and P. Hliněný. Isomorphism testing for T-graphs in FPT. *CoRR*, abs/2111.10910, 2021.

[3] V. Arvind, B. Das, J. Köbler, and S. Toda. Colored hypergraph isomorphism is fixed parameter tractable. *Algorithmica*, 71(1):120–138, 2015.

[4] V. Arvind, R. Nedela, I. Ponomarenko, and P. Zeman. Testing isomorphism of chordal graphs of bounded leafage is fixed-parameter tractable. *CoRR*, abs/2107.10689, 2021.

[5] D. Ağaoğlu and P. Hliněný. Isomorphism problem for S_d-graphs. In J. Esparza and D. Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 4:1–4:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[6] D. Ağaoğlu and P. Hliněný. Automorphism group of marked interval graphs in FPT. *CoRR*, abs/2202.12664, 2022.

[7] D. Ağaoğlu Çağırıcı, O. Çağırıcı, J. Derbisz, T. A. Hartmann, P. Hliněný, J. Kratochvíl, T. Krawczyk, and P. Zeman. Beyond circular-arc graphs – Recognizing lollipop graphs and medusa graphs. *arXiv e-prints*, page arXiv:2212.05433, Dec. 2022.

[8] D. Ağaoğlu Çağırıcı, O. Çağırıcı, J. Derbisz, T. A. Hartmann, P. Hliněný, J. Kratochvíl, T. Krawczyk, and P. Zeman. Recognizing H-Graphs - Beyond Circular-Arc Graphs. In J. Leroux, S. Lombardy, and D. Peleg, editors, *48th International Symposium on Mathematical*

Foundations of Computer Science (MFCS 2023), volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:14, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[9] D. Ağaoğlu Çağırıcı, J. Derbisz, G. Gutowski, and T. Krawczyk. Circular-arc graphs and the Helly property, II. *in preparation.*

[10] D. Ağaoğlu Çağırıcı and P. Hliněný. Efficient isomorphism for S_d-graphs and T-graphs. *Algorithmica*, 2022.

[11] D. Ağaoğlu Çağırıcı and P. Hliněný. Isomorphism testing for T-graphs in FPT. In *WALCOM*, volume 13174 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2022.

[12] D. Ağaoğlu Çağırıcı and P. Zeman. Recognition and isomorphism of proper U-graphs in FPT-time. *CoRR*, abs/2206.13372, 2022.

[13] D. Ağaoğlu and O. Çağırıcı. Unit disk visibility graphs. In J. Nešetřil, G. Perarnau, J. Rué, and O. Serra, editors, *Extended Abstracts EuroComb 2021*, pages 390–396, Cham, 2021. Springer International Publishing.

[14] L. Babai. Monte Carlo algorithms in graph isomorphism testing. *Tech. Rep. 79-10, Université de Montréal*, 1979. 42 pages.

[15] L. Babai and E. M. Luks. Canonical labeling of graphs. In D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, and J. I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 171–183. ACM, 1983.

[16] M. Biró, M. Hujter, and Z. Tuza. Precoloring extension. i. interval graphs. *Discrete Mathematics* ***100***, pages 267–279, 1992.

[17] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976.

[18] A. Bouland, A. Dawar, and E. Kopczyński. On tractable parameterizations of graph isomorphism. In D. M. Thilikos and G. J. Woeginger, editors, *Parameterized and Exact Computation - 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012.*

Proceedings, volume 7535 of *Lecture Notes in Computer Science*, pages 218–230. Springer, 2012.

[19] A. Brandstädt, V. B. Le, and J. Spinrad. Graph classes: A survey. *SIAM Monographs on Discrete Mathematics and Applications, ISBN 0-89871-432-X*, 1999.

[20] H. Breu and D. G. Kirkpatrick. On the complexity of recognizing intersection and touching graphs of disks. In F. Brandenburg, editor, *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20-22, 1995, Proceedings*, volume 1027 of *Lecture Notes in Computer Science*, pages 88–98. Springer, 1995.

[21] S. Chaplick, P. A. Golovach, T. A. Hartmann, and D. Knop. Recognizing proper tree-graphs. In Y. Cao and M. Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 8:1–8:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[22] S. Chaplick, M. Töpfer, J. Voborník, and P. Zeman. On H-topological intersection graphs. In H. L. Bodlaender and G. J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, volume 10520 of *Lecture Notes in Computer Science*, pages 167–179. Springer, 2017.

[23] S. Chaplick, M. Töpfer, J. Voborník, and P. Zeman. On H-topological intersection graphs. *Algorithmica*, 83(11):3281–3318, 2021.

[24] F. R. K. Chung. On the cutwidth and the topological bandwidth of a tree. *SIAM J. Alg. Discr. Meth.*, 6:268–277, 1985.

[25] C. J. Colbourn and K. S. Booth. Linear time automorphism algorithms for trees, interval graphs, and planar graphs. *SIAM J. Comput.*, 10(1):203–225, 1981.

[26] A. R. Curtis, M. C. Lin, R. M. McConnell, Y. Nussbaum, F. J. Soulignac, J. P. Spinrad, and J. L. Szwarcfiter. Isomorphism of graph classes related to the circular-ones property. *Discret. Math. Theor. Comput. Sci.*, 15(1):157–182, 2013.

[27] X. Deng, P. Hell, and J. Huang. Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM J. Comput.*, 25(2):390–403, 1996.

[28] J. Derbisz and T. Krawczyk. Circular-arc graphs and the Helly property. *in preparation.*

[29] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[30] I. Dinur, O. Regev, and C. D. Smyth. The hardness of 3-uniform hypergraph coloring. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 33. IEEE Computer Society, 2002.

[31] S. Evdokimov and I. N. Ponomarenko. Isomorphism of coloured graphs with slowly increasing multiplicity of jordan blocks. *Comb.*, 19(3):321–333, 1999.

[32] J. Fiala, I. Rutter, P. Stumpf, and P. Zeman. Extending partial representations of circular-arc graphs. *CoRR*, abs/2108.13076, 2021.

[33] S. Földes and P. L. Hammer. Split graphs. *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing (Louisiana State Univ., Baton Rouge, La., 1977).Congressus Numerantium, No. XIX, Utilitas Math., Winnipeg, Man.,*, pages 311–315, 1977.

[34] M. L. Furst, J. E. Hopcroft, and E. M. Luks. Polynomial-time algorithms for permutation groups. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 36–41. IEEE Computer Society, 1980.

[35] M. L. Furst, J. E. Hopcroft, and E. M. Luks. A subexponential algorithm for trivalent graph isomorphism. In *Proc. 11th Southeastern Conf. Combinatorics, Graph Theory, and Computing, Congressum Numerantium 3*, 1980.

[36] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

[37] M. C. Golumbic. Algorithmic graph theory and perfect graphs. *Academic Press, ISBN 0-12-289260-7, MR 0562306*, 1980.

[38] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., NLD, 2004.

[39] D. Gonçalves and P. Ochem. On star and caterpillar arboricity. *Discrete Mathematics*, 309(11):3694–3702, 2009.

[40] P. L. Hammer and B. Simeone. The splittance of a graph. *Comb.*, 1(3):275–284, 1981.

[41] J. M. Keil. Finding hamiltonian circuits in interval graphs. *Inf. Process. Lett.*, 20(4):201–206, 1985.

[42] P. Klavík, J. Kratochvíl, Y. Otachi, and T. Saitoh. Extending partial representations of subclasses of chordal graphs. *Theor. Comput. Sci.*, 576:85–101, 2015.

[43] J. Kratochvíl. String graphs. II. recognizing string graphs is np-hard. *J. Comb. Theory, Ser. B*, 52(1):67–78, 1991.

[44] J. Kratochvíl. A special planar satisfiability problem and a consequence of its np-completeness. *Discret. Appl. Math.*, 52(3):233–252, 1994.

[45] J. Kratochvíl and J. Matousek. Intersection graphs of segments. *J. Comb. Theory, Ser. B*, 62(2):289–315, 1994.

[46] S. Kratsch and P. Schweitzer. Isomorphism for graphs of bounded feedback vertex set number. In H. Kaplan, editor, *Algorithm Theory - SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010. Proceedings*, volume 6139 of *Lecture Notes in Computer Science*, pages 81–92. Springer, 2010.

[47] T. Krawczyk. Testing isomorphism of circular-arc graphs - Hsu's approach revisited. *CoRR*, abs/1904.04501, 2019.

[48] M. C. Lin and J. L. Szwarcfiter. Characterizations and linear time recognition of helly circular-arc graphs. In D. Z. Chen and D. T. Lee, editors, *Computing and Combinatorics, 12th Annual International Conference, COCOON 2006, Taipei, Taiwan, August 15-18, 2006, Proceedings*, volume 4112 of *Lecture Notes in Computer Science*, pages 73–82. Springer, 2006.

[49] M. C. Lin and J. L. Szwarcfiter. Characterizations and recognition of circular-arc graphs and subclasses: A survey. *Discret. Math.*, 309(18):5618–5635, 2009.

[50] D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. *SIAM J. Comput.*, 46(1):161–189, 2017.

[51] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In T. Hagerup and J. Katajainen, editors, *Algorithm Theory - SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, Humlebaek, Denmark, July 8-10, 2004, Proceedings*, volume 3111 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 2004.

[52] R. Mathon. A note on the graph isomorphism counting problem. *Inf. Process. Lett.*, 8(3):131–132, 1979.

[53] Y. Matsui, R. Uehara, and T. Uno. Enumeration of the perfect sequences of a chordal graph. *Theor. Comput. Sci.*, 411(40-42):3635–3641, 2010.

[54] R. M. McConnell. Linear-time recognition of circular-arc graphs. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 386–394. IEEE Computer Society, 2001.

[55] G. B. Mertzios, I. Sau, and S. Zaks. The recognition of tolerance and bounded tolerance graphs. In J. Marion and T. Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*, volume 5 of *LIPIcs*, pages 585–596. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[56] G. L. Miller. Isomorphism testing for graphs of bounded genus. In R. E. Miller, S. Ginsburg, W. A. Burkhard, and R. J. Lipton, editors, *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 225–235. ACM, 1980.

[57] D. Neuen. Graph Isomorphism for Unit Square Graphs. In P. Sankowski and C. Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 70:1–70:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[58] I. N. Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. Journal of Soviet Mathematics 55, pages 1621–1643, 1991.

[59] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5(2):266–283, 1976.

[60] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.*, 6(3):505–517, 1977.

[61] R. Uehara. Simple geometrical intersection graphs. In S.-i. Nakano and M. S. Rahman, editors, *WALCOM: Algorithms and Computation*, pages 25–33, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[62] R. Uehara, S. Toda, and T. Nagoya. Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. *Discret. Appl. Math.*, 145(3):479–482, 2005.

[63] V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich. Graph isomorphism problem. *J. of Soviet Mathematics*, 29:1426–1481, 1985.