FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Pavel Klavík

Extension Properties of Graphs and Structures

Computer Science Institute of Charles University

Supervisor of doctoral thesis:	Prof. RNDr. Jaroslav Nešetřil, DrSc.
Study program:	Informatics
Study branch:	Discrete Models and Algorithms

Prague 2017

Acknowledgements

The results presented in this thesis are based on several conference and journal papers. Below, I give their list in the order of appearence.

Chapters 1 and 2. For these introductory chapters, only small parts and some figures are used from the following papers.

[216] **Pavel Klavík, Jan Kratochvíl, Tomasz Krawczyk, Bartosz Walczak.**
Extending partial representations of function graphs and permutation graphs.
In Algorithms, ESA 2012, volume 7501 of Lecture Notes in Computer Science, pages 671–682, 2012.
The full version at `http://arxiv.org/abs/1204.6391`.

[211] **Pavel Klavík, Jan Kratochvíl, Yota Otachi, Ignaz Rutter, Toshiki Saitoh, Maria Saumell, and Tomáš Vyskočil**.
Extending Partial Representations of Proper and Unit Interval Graphs.
In Algorithm Theory, SWAT 2014, volume 8503 of Lecture Notes in Computer Science, pages 253–264, 2014.

[212] **Pavel Klavík, Jan Kratochvíl, Yota Otachi, Ignaz Rutter, Toshiki Saitoh, Maria Saumell, and Tomáš Vyskočil**.
Extending Partial Representations of Proper and Unit Interval Graphs.
The journal version of [211]. Algorithmica, 77(4):1071–1104, 2017.
The pre-print available at `http://arxiv.org/abs/1207.6960`.

[213] **Pavel Klavík, Jan Kratochvíl, Yota Otachi, Toshiki Saitoh.**
Extending Partial Representations of Subclasses of Chordal Graphs.
In Algorithms and Computation, ISAAC 2012, volume 7676 of Lecture Notes in Computer Science, pages 444–454, 2012.

[214] **Pavel Klavík, Jan Kratochvíl, Yota Otachi, Toshiki Saitoh.**
Extending Partial Representations of Subclasses of Chordal Graphs.
The journal version of [213]. Theoretical Computer Science, 576:85–101, 2015.
The pre-print available at `http://arxiv.org/abs/1207.0255`.

[58] **Steven Chaplick, Radoslav Fulek, and Pavel Klavík.**
Extending Partial Representations of Circle Graphs.
In Graph Drawing, GD 2013, volume 8242 of Lecture Notes in Computer Science, pages 131–142, 2013.

[59] **Steven Chaplick, Radoslav Fulek, and Pavel Klavík.**
Extending Partial Representations of Circle Graphs.
The journal version of [58], submitted, 2015.
The pre-print available at `http://arxiv.org/abs/1309.2399`.

[18] **Martin Balko, Pavel Klavík, and Yota Otachi.**
Bounded Representations of Interval and Proper Interval Graphs.
In Algorithms and Computation, ISAAC 2013, volume 8283 of Lecture Notes in Computer Science, pages 535–546, 2013.

[19] **Martin Balko, Pavel Klavík, and Yota Otachi.**
Bounded Representations of Interval and Proper Interval Graphs.
In preparation, 2017.

Chapter 3.

[216] **Pavel Klavík, Jan Kratochvíl, and Tomáš Vyskočil.**
Extending Partial Representations of Interval Graphs.
In Theory and Applications of Models of Computation, TAMC 2011, volume 6648 of Lecture Notes in Computer Science, pages 276–285, 2011.

[215] **Pavel Klavík, Jan Kratochvíl, Yota Otachi, Toshiki Saitoh, and Tomáš Vyskočil.**
Extending Partial Representations of Interval Graphs.
The journal version of [216], accepted to Algorithmica, 2016.
The pre-print available at `http://arxiv.org/abs/1306.2182`.

Chapter 4.

[222] **Pavel Klavík, and Maria Saumell.**
Minimal Obstructions for Partial Representations of Interval Graphs.
In Algorithms and Computation, ISAAC 2014, volume 8889 of Lecture Notes in Computer Science, pages 401–413, 2014.

[223] **Pavel Klavík, and Maria Saumell.**
Minimal Obstructions for Partial Representations of Interval Graphs.
The journal version of [222], submitted, 2015.
The pre-print available at `http://arxiv.org/abs/1406.6228`.

Chapter 5.

[219] **Pavel Klavík, Yota Otachi, and Jiří Šejnoha.**
On the Classes of Interval Graphs of Limited Nesting and Count of Lengths.
In 27th International Symposium on Algorithms and Computation, ISAAC 2016, volume 64 of Leibniz International Proceedings in Informatics (LIPIcs), pages 45:1–45:13, 2016.

[221] **Pavel Klavík, Yota Otachi, and Jiří Šejnoha.**
On the Classes of Interval Graphs of Limited Nesting and Count of Lengths.
The journal version of [219], submitted, 2017.
The pre-print available at `http://arxiv.org/abs/1510.03998`.

Chapter 6. For this introductory chapter, only small parts and some figures are used from the following papers.

[224] **Pavel Klavík and Peter Zeman**.
Automorphism groups of geometrically represented graphs.
In 32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, volume 30 of Leibniz International Proceedings in Informatics (LIPIcs), pages 540–553, 2015.

[225] **Pavel Klavík and Peter Zeman**.
Automorphism groups of geometrically represented graphs.
The journal version of [224], submitted, 2015.
The pre-print available at `http://arxiv.org/abs/1407.2136`.

Chapters 7, 10, and 11.

[118] **Jiří Fiala, Pavel Klavík, Jan Kratochvíl, and Roman Nedela.**
Algorithmic Aspects of Regular Graph Covers with Applications to Planar Graphs.
In Automata, Languages, and Programming, ICALP 2014, volume 8572 of Lecture Notes in Computer Science, pages 489–501, 2014.

[119] **Jiří Fiala, Pavel Klavík, Jan Kratochvíl, and Roman Nedela.**
3-connected Reduction for Regular Graph Covers.
The journal version of the first part of [118], submitted, 2017.
The pre-print available at `http://arxiv.org/abs/1503.06556`.

[120] **Jiří Fiala, Pavel Klavík, Jan Kratochvíl, and Roman Nedela.**
Algorithmic Aspects of Regular Graphs Covers.
The journal version of the second part of [118], submitted, 2017.
The pre-print available at `http://arxiv.org/abs/1609.03013`.

Chapter 8.

[217] **Pavel Klavík, Roman Nedela, and Peter Zeman.**
Jordan-like Characterization of Automorphism Groups of Planar Graphs.
Submitted, 2017. The pre-print available at `http://arxiv.org/abs/1506.06488`.

Chapter 9.

[209] **Pavel Klavík, Dušan Knop, and Peter Zeman.**
Graph Isomorphism Restricted by Lists.
Submitted, 2017. The pre-print available at `http://arxiv.org/abs/1607.03918`.

I would like to thank my supervisor prof. Jaroslav Nešetřil for discussions and generous support during my PhD studies. I would like to thank the students of my classes at Charles University for many ideas and discussions—this is what moves mathematics forward. I am glad that I could supervise Bachelor's and Master's thesis' of Peter Zeman, we did a lot of great collaboration together.

I am also thankful to Zdeněk Hedrlín and his group, namely Jiří Šejnoha, for changing my view of the world. We worked on mind modelling, and sharing and understanding of Big Picture using structural diagrams. I would like to thank about 200 students of Charles University who participated in our education experiment of this method for examination of linear algebra.

I would like to thank Jan Bok, Andrew Goodall, and Peter Zeman for proofreading some parts of this thesis.

Last, but not least, I would like to thank my family and to my friends for supporting me during my studies, especially to my girlfriend Kamila.

Prague, June 28, 2017 Pavel Klavík

Contents

Title: Extension Properties of Graphs and Structures
Author: RNDr. Pavel Klavík
Department: Computer Science Institute of Charles University
Supervisor: Prof. RNDr. Jaroslav Nešetřil, DrSc.
Supervisor's e-mail: nesetril@iuuk.mff.cuni.cz
Keywords: geometric representations of graphs,
partial representation extension,
regular graph covering,
automorphism groups,
graph isomorphism problem

Abstract:

The main motivation for graph drawing and geometric representations is finding ways to visualize graphs efficiently to make their structure as understandable as possible. In this thesis, we are concerned with structural properties which are implied for graphs having certain geometric representations. We study two types of geometric representations: intersection representations in which the vertices are represented by geometric sets while the edges are encoded by their intersections, and planar embeddings of planar graphs which are drawing of graphs into the plane without crossing edges. The existence of geometric representations can be used to deduce additional information about graphs. The main idea of this thesis is to ask what extra information can be deduced from the structure of all possible geometric representations.

In Part I, we study the partial representation extension problems for intersection representations. Aside from the graph, the input also gives a partial representation, which prescribes a representation of an induced subgraph. We ask whether this partial representation can be extended to a full representation of the input graph without altering the predrawn sets. I introduced this problem in 2010 in my Bachelor's thesis. We survey the state-of-the-art results for many graph classes. We concentrate on interval graphs and prove both structural and algorithmic results for the partial representation extension problem.

In Part II, we study algebraic properties of graphs, namely their automorphism groups, the graph isomorphism problem and regular graph covering. The main structural tool is the 3-connected reduction which decomposes each graph G into 3-connected components. We are mostly concerned with planar graphs, but some of our results apply to general graphs. In 1867, Jordan described an inductive characterization of the automorphism groups of trees. We describe the first Jordan-like inductive characterization of the automorphism groups of planar graphs. Also, we study the list restricted graph isomorphism problem for variety of graph classes and parameters. For regular graph covering, we describe all regular quotients of planar graphs and construct an FPT algorithm for testing regular graph covering for planar inputs.

1 Introduction to Geometric Representations of Graphs

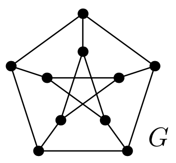
This chapter contains:

- *1.1: Motivation.* We introduce graph drawing and graph representations by three detailed motivations: 1) Tutte's spring embedding of planar graphs using linear algebra and physics, 2) Benzer's study of DNA using interval graphs, 3) circle packings of planar graphs and their relation to Riemann Mapping Theorem in complex analysis.
- *1.2: Definitions.* We define graphs, geometric representations, the recognition problems, and the other main definitions used in this thesis.
- *1.3: Intersection Representations.* We describe the main classes of intersection graphs which are discussed in this thesis.
- *1.4: Planar Embeddings.* We describe planar and spherical embeddings. Also, the special geometric role of 3-connected planar graphs due to Steinitz, Whitney and Mani is discussed.
- *1.5: Results of This Thesis.* We describe the organization of this thesis and give a short overview of the main proved results.

`http://pavel.klavik.cz/orgpad/geom_rep.html`

1.1 Motivation

The study of geometric representations of graphs is as old as graph theory itself and drawings of graphs appear in every introductory book and lecture. We are so used to these drawings that when one says "think of the Petersen graph", we likely think of a figure similar to the one on the left instead of an abstract pair of vertices and edges as on the right. (However, the description on the right is sometimes useful.)



versus

$$\boldsymbol{V}(G) = \binom{\{1,2,3,4,5\}}{2} = \big\{\{x,y\} : x,y \in \{1,2,3,4,5\}, x \neq y\big\}$$

$$\boldsymbol{E}(G) = \big\{uv : u,v \in \boldsymbol{V}(G), u \cap v = \emptyset\big\}$$

The main motivation for graph drawing and geometric representations is finding ways to visualize graphs efficiently to make their structure as understandable as possible; see Fig. 1.1. The issue with this statement is that the notion of efficiency of representations is very vague. In my experience, there are two basic approaches in the field of graph drawing:

- *Theoretical approach* – studying restrictive types of representations of restricted graphs. The representations studied are precisely defined but might not be very helpful (and so non-efficient).
- *Applied approach* – construction of nice representations of large graphs and networks, coming from practice, and construction of algorithms based on heuristics.

Of course, results in the field are based on a combination of these approaches. In this thesis, our approach is theoretical. We are concerned with structural properties which are implied for graphs having certain geometric representations.

Three Applications of Geometric Representations. In the rest of this section, we describe three diverse connections of graph representations to other areas of mathematics and to other sciences.

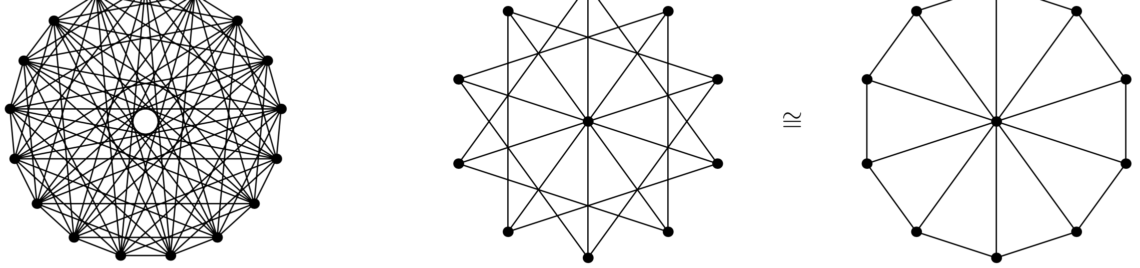


Figure 1.1: On the left, the drawing is not very understandable since the graph contains too many edges. The drawing in the middle is also hard to understand, while the structure of the same graph is immediately apparent from the drawing on the right.

Section 1.1.1: Tutte's Spring Embedding of planar graphs [343] and its connection to linear algebra and physics. Tutte proved that every 3-connected planar graph can be drawn by pinning vertices of the outer face into a convex polygon, replacing edges by springs and letting the system of springs come to rest. By the laws of physics, the system in the rest minimizes the total potential energy of springs, and the energy-minimizing state can be computed by solving linear systems. Having over 1000 citations in Google Scholar, this work greatly influenced the study of graph drawing and embeddings of planar graphs.

Section 1.1.2: Benzer's application of interval graphs in his study [24] of the structure of DNA and genes from 1959. Each mutation modifies a small connected part of DNA. Benzer constructed experiments which tested for each pair of mutations whether they overlapped. Therefore, he obtained information about intersections in the data and he pioneered the mathematical study of the possible topologies which can be inferred from the data. For geometric representations of graphs, his work is historically important because it had a profound influence on the study of interval graphs and many other classes of intersection graphs.

Section 1.1.3. Thurston's application of circle packings of planar graphs [309] to the Riemann Mapping Theorem [302] for conformal mappings in complex analysis. Ensured by the existence of complex derivatives, conformal mappings are locally rigid since they map infinitesimal circles to infinitesimal circles. The Riemann Mapping Theorem states that they are globally flexible: any simply connected region can be mapped by a conformal bijection into the unit disk. (For instance, the unit square can be mapped to the unit disk while locally preserving angles and ratios of lengths.) The Riemann Mapping Theorem proves existence, but explicitely finding conformal bijections is a difficult problem. Thurston's method is an iterative approach to compute approximative conformal bijections using circle packings.

I choose these applications based on several criteria. Most importantly, they should be beautiful and surprising, but of course this criterion is very subjective. They should be important and influential. I really love their connection to linear algebra and complex analysis, which are both subjects I am very fond of. (And I am very thankful to Charles University that I could teach them for a few years.)

I also wanted to include applications which are not widely known, i.e., which are not a part of introductory texts and lectures on graph theory or even geometric representations. (This was one of the reasons not to include Euler's formula for the number of vertices, edges, and faces of polyhedra.) From these three applications, Tutte's Spring Embedding is likely the most widely known.

Lastly, these applications should be important with respect to the rest of the thesis. Therefore, they are concerned with interval graphs (key in Part I) and planar graphs (key in Part II). In particular, Section 1.1.1 sheds some light on the structure of 3-connected planar graphs. Complex analysis, described in Section 1.1.3, is important for describing symmetries of the sphere, which play a key role in Chapters 8, 10, and 11.

It is important to note that *the rest of this section may be freely skipped* and only some small details are occasionally referred to in the remainder of this thesis. On the other hand, I believe that these three applications greatly spice up this thesis. In the remainder of this chapter, i.e., Sections 1.2, 1.3, 1.4, and 1.5, we give an overview of important geometric representations of graphs and describe the main goals and results of this thesis.

1.1.1 Tutte's Spring Embedding and Spectral Graph Drawing

Planar graphs are graphs which can be drawn in the plane without edges crossing, see Section 1.4 for more details. In 1963, Tutte [343] described an elegant way to construct embeddings of 3-connected planar graphs based on linear algebra and physics.

Laplacian Matrices. We start with a small detour into applications of linear algebra in graph theory; for more detail, see [325, 66, 151]. The central idea is to study graph properties from algebraic properties of an associated matrix. For instance, to an n-vertex graph G with $\boldsymbol{V}(G) = \{1, \ldots, n\}$, we can associate two n-by-n matrices: the adjacency matrix A_G and the Laplacian matrix L_G, defined as

$$(A_G)_{i,j} = \begin{cases} 1 & \text{if } ij \in \boldsymbol{E}(G), \\ 0 & \text{else,} \end{cases} \qquad \text{and} \qquad (L_G)_{i,j} = \begin{cases} d(i) & \text{if } i = j, \\ -1 & \text{if } ij \in \boldsymbol{E}(G), \\ 0 & \text{else,} \end{cases}$$

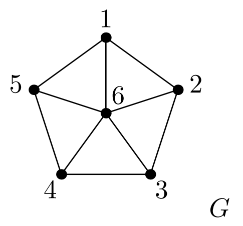
where $d(i)$ is the degree of the vertex i. Figure 1.2 depicts an example.

By choosing different matrix representations of G, different graph properties of G are revealed. Why? The key concept of linear algebra is that an m-by-n real matrix can be viewed in two different ways: as an m-by-n table of data, or as a representation of a linear mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$. There is no difference between A_G and L_G when we look at them as tables of data since they contain the same information. However, they represent completely different linear transformations from $\mathbb{R}^n$ to $\mathbb{R}^n$.

We can think of vectors $\boldsymbol{x} \in \mathbb{R}^n$ as of evaluations of the vertices $\boldsymbol{V}(G)$, i.e., x_i is the value of the vertex i. For $\boldsymbol{y} = A_G \cdot \boldsymbol{x}$ and $\boldsymbol{z} = L_G \cdot \boldsymbol{x}$, we get

$$\begin{aligned} y_i &= \sum_{ij \in \boldsymbol{E}(G)} x_j, \\ z_i &= d(i) \cdot x_i - \sum_{ij \in \boldsymbol{E}(G)} x_j = d(i) \cdot \left(x_i - \frac{1}{d(i)} \sum_{ij \in \boldsymbol{E}(G)} x_j \right). \end{aligned} \tag{1.1}$$

By iterating A_G, we distribute the values of vertices in $\boldsymbol{x}$ throughout the graph which can be used for instance in probabilistic models of random walks. We are concerned



$$A_G = \begin{pmatrix} & 1 & & & 1 & 1 \\ 1 & & 1 & & & 1 \\ & 1 & & 1 & & 1 \\ & & 1 & & 1 & 1 \\ 1 & & & 1 & & 1 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}, \qquad L_G = \begin{pmatrix} 3 & -1 & & & -1 & -1 \\ -1 & 3 & -1 & & & -1 \\ & -1 & 3 & -1 & & -1 \\ & & -1 & 3 & -1 & -1 \\ -1 & & & -1 & 3 & -1 \\ -1 & -1 & -1 & -1 & -1 & 5 \end{pmatrix}.$$

Figure 1.2: The adjacency matrix A_G and the Laplacian matrix L_G of the graph G.

with the Laplacian matrix L_G for which the value z_i is the difference between x_i and the average x_j over all neighbors j of i, and this difference is multiplied by $d(i)$.

The Laplacian matrix L_G is the discretization of the Laplace operator Δ, which is a differential operator used in physics to model diffusion, electrostatic potential, etc. In the Cartesian coordinates in $\mathbb{R}^2$, it is defined as

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

The value $\Delta f(x, y)$ is the difference between $f(x, y)$ and the average value of f over infinitesimal circle centered at (x, y) [287, Section 12.II.3]. Equivalently, the Laplace operator Δ is the divergence of the gradient: $\Delta = \nabla \cdot \nabla$. We note that this corresponds to the discrete analog $L_G = I_G I_G^\mathsf{T}$ where I_G is the incidence matrix of the graph G; see [333] for more detail.

Spring Embedding. Let G be a 3-connected planar graph. We fix positions of k vertices of an outer face in such a way that they form a convex k-gon. Tutte [343] proved that L_G can be used to construct the rest of the planar embedding.

Let $1, \dots, k$ be the vertices of the outer face and let $k+1, \dots, n$ be the remaining vertices. We want to compute vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of positions such that the vertex i is placed in the position (x_i, y_i). The fixed positions of $1, \dots, k$ prescribe the values $x_1, \dots, x_k$ and $y_1, \dots, y_k$. For each vertex $k+1, \dots, n$, we require that it is placed in the barycenter of the position of its neighbors, and we call it a *barycentric representation.*

By (1.1), this requirement is equivalent to

$$(L_G \cdot \boldsymbol{x})_j = 0 \qquad \text{and} \qquad (L_G \cdot \boldsymbol{y})_j = 0, \qquad \forall j = k+1, \dots, n.$$

Since the values $x_1, \dots, x_k$ and $y_1, \dots, y_k$ are fixed, we can substitute them into these linear systems, and we get linear systems

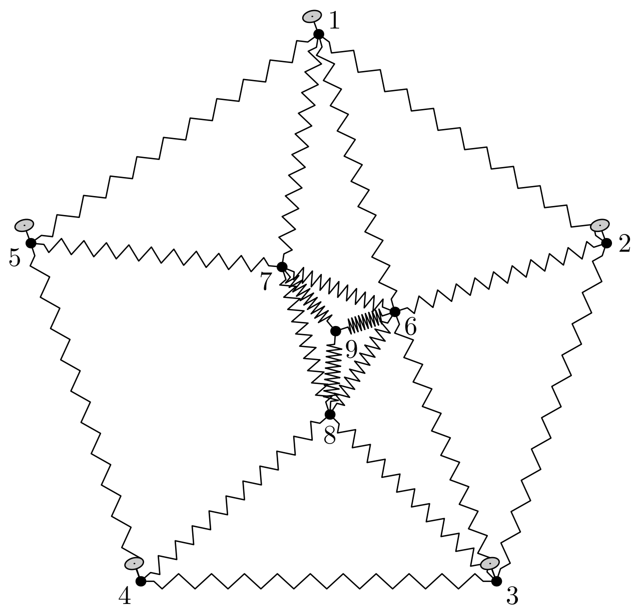
$$\widehat{L_G} \cdot \hat{\boldsymbol{x}} = \boldsymbol{b} \qquad \text{and} \qquad \widehat{L_G} \cdot \hat{\boldsymbol{y}} = \boldsymbol{c}, \tag{1.2}$$

where $\widehat{L_G}$ is the minor of L_G created by removing the first k rows and columns, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ without the first k coordinates, and $\boldsymbol{b}$ and $\boldsymbol{c}$ are the right-hand sides determined by the positions of the vertices $1, \dots, k$.

Tutte [343] proved that $\widehat{L_G}$ is regular, so there exist unique solutions to the linear systems in (1.2), and when edges are represented by straight-line segments, the barycentric representation is a planar embedding of G.

Even though it was not described in [343], there is the following well-known physics interpretation, which explains why it is often called a *spring embedding*; see [231] for a survey. Figure 1.3 shows an example. Suppose that we pin vertices $1, \dots, k$ to their positions. For every edge, we place a spring. The springs corresponding to edges sharing a vertex are attached to each other, and for the edges incident with the vertices $1, \dots, k$, we attach their springs to pinned positions. We let the system of springs come to rest, and let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the vectors of positions of the vertices $1, \dots, n$.

Ignoring constants, the potential energy of a spring between i and j is $(x_i - x_j)^2 + (y_i - y_j)^2$. Assuming that all springs are the same and ideally elastic, the total



$$\left(\begin{array}{rrrr|c} 6 & -1 & -1 & -1 & x_1+x_2+x_3 \\ -1 & 5 & -1 & -1 & x_1+x_5 \\ -1 & -1 & 5 & -1 & x_3+x_4 \\ -1 & -1 & -1 & 3 & 0 \end{array}\right)$$

$$\left(\begin{array}{rrrr|c} 6 & -1 & -1 & -1 & y_1+y_2+y_3 \\ -1 & 5 & -1 & -1 & y_1+y_5 \\ -1 & -1 & 5 & -1 & y_3+y_4 \\ -1 & -1 & -1 & 3 & 0 \end{array}\right)$$

Figure 1.3: Suppose that the vertices $1,\dots,5$ are pinned in the depicted positions. The remaining vertices $6,\dots,9$ are placed in such a way that the total potential energy of the system of springs is minimized. Their positions $x_6,\dots,x_9$ and $y_6,\dots,y_9$ are computed by solving the depicted linear systems $\widehat{L_G}\cdot\hat{\boldsymbol{x}}=\boldsymbol{b}$ and $\widehat{L_G}\cdot\hat{\boldsymbol{y}}=\boldsymbol{c}$, and they are placed at the barycenters of their neighbors.

potential energy of the system is

$$E_p=\sum_{ij\in\boldsymbol{E}(G)}(x_i-x_j)^2+(y_i-y_j)^2=\boldsymbol{x}^{\mathsf{T}}L_G\,\boldsymbol{x}+\boldsymbol{y}^{\mathsf{T}}L_G\,\boldsymbol{y}.$$

The system in the rest minimizes the total potential energy E_p, subject to the boundary constraints given by the fixed positions $x_1,\dots,x_k$ and $y_1,\dots,y_k$. The minimization then leads to

$$\min_{\hat{\boldsymbol{x}}}\ \hat{\boldsymbol{x}}^{\mathsf{T}}\widehat{L_G}\,\hat{\boldsymbol{x}}+2\hat{\boldsymbol{x}}^{\mathsf{T}}\boldsymbol{b}+\min_{\hat{\boldsymbol{y}}}\ \hat{\boldsymbol{y}}^{\mathsf{T}}\widehat{L_G}\,\hat{\boldsymbol{y}}+2\hat{\boldsymbol{y}}^{T}\boldsymbol{c}+d, \tag{1.3}$$

where d corresponds to the total potential energy of strings between the pinned vertices $1,\dots,k$. Since $\widehat{L_G}$ is symmetric positive definite, by [334, Section 6.4] and [333, Section 1.4], this minimization problem is equivalent with solving the linear systems in (1.2).

Spectral Graph Drawing. A similar approach was pioneered by Hall [173] for drawing general graphs; see also [233, 234]. Suppose that we want to construct a barycentric representation for a general graph G. To place every vertex at the barycenter of its neighbors, the positions have to satisfy $L_G\cdot\boldsymbol{x}=\boldsymbol{0}$ and $L_G\cdot\boldsymbol{y}=\boldsymbol{0}$ which is only possible for degenerate drawings placing all vertices at the same point.

Therefore, we relax the conditions to an *almost barycentric representation*, satisfying that both $L_G\cdot\boldsymbol{x}$ and $L_G\cdot\boldsymbol{y}$ are small. To avoid degenerate drawings, we want $\boldsymbol{x}$ and $\boldsymbol{y}$ to be orthogonal to each other and to the vector $(1,1,\dots,1)$. To control the proportions, let $\|\boldsymbol{x}\|=\|\boldsymbol{y}\|$. If G is connected, then the Laplacian matrix L_G is symmetric positive semidefinite with eigenvalues

$$0=\lambda_n<\lambda_{n-1}\le\lambda_{n-2}\le\cdots\le\lambda_1$$

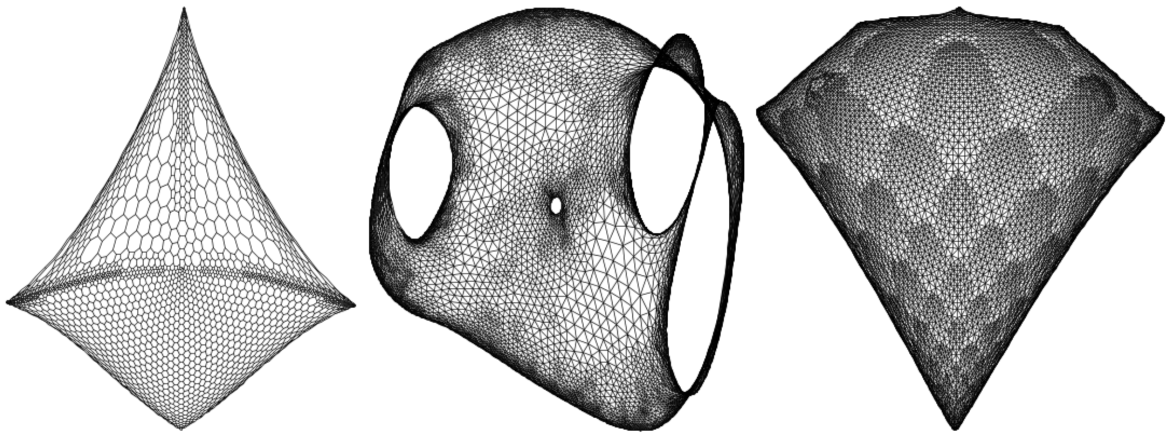


Figure 1.4: This figure is taken from [233, 234]. Spectral graph drawings of three graphs using eigenvectors of the Laplacian matrices. Notice that almost every vertex is placed very close to the barycenter of its neighbors.

and eigenvectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n$, i.e., $L_G \cdot \boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$. By the Spectral Decomposition Theorem [334], there exists an orthogonal basis of eigenvectors.

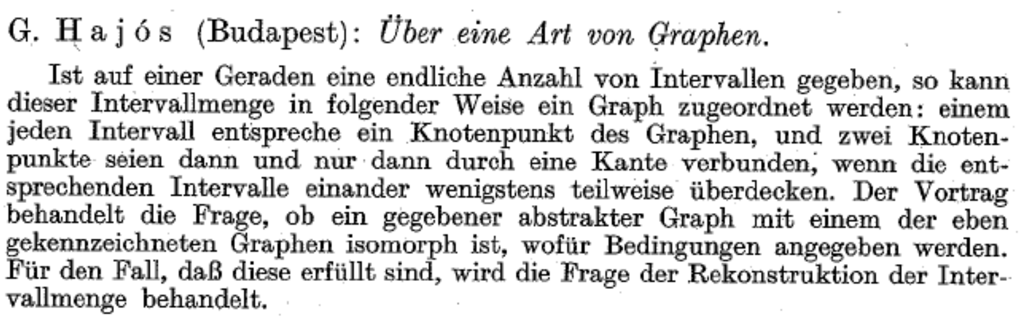
We have $\boldsymbol{u}_n = c \cdot (1, 1, \dots, 1)$. To get a spectral graph drawing, we choose $\boldsymbol{x} = \boldsymbol{u}_{n-1}$ and $\boldsymbol{y} = \boldsymbol{u}_{n-2}$ such that $\|\boldsymbol{x}\| = \|\boldsymbol{y}\|$, which minimizes the error in almost barycentric representations. When λ_{n-1} and λ_{n-2} are suitably close to zero, the drawing is very close to a barycentric one, as can be observed in Fig. 1.4.

1.1.2 Benzer's Study of the Structure of DNA

Interval graphs were first mentioned in 1957 in a talk of the Hungarian mathematician Hajós [172]; see Fig. 1.5 for his abstract. Unfortunately, it is not clear what was his motivation to introduce this class of graphs and which results he presented. In 1959, Benzer [24] independently derived interval graphs in his experimental study of local structures in DNA. His paper greatly motivated the research of interval graphs and related classes of geometrically represented graphs.

History of DNA. The origin of genetics lies in the breeding experiments of Mendel between 1856 and 1863. He noticed inheritance patterns of certain traits in pea plants and described simple rules with some traits being dominant and others being recessive. But the hereditary mechanisms remained a mystery till the 20th century. Already in 1869, Miescher made a remarkable discovery of DNA (deoxyribonucleic acid) molecules in the nuclei of human white blood cells; see [360]. In 1910, Morgan identified that genes reside in chromosomes. In the 1940's, it was identified that the gene's information is encoded in DNA.

In 1953, Watson and Crick [353] published the famous double-helix model of DNA, by building on the previously known result in the field [299]. Levene [250] identified in 1919 components of DNA, in particular four nucleobases: adenine (A), guanine (G), cytosine (C), and thymine (T). Chargaff [62] showed in 1950 that the structure of nucleobases' varies among species and that the amount of A is the same as the amount of C and the amount of G is the same as the amount of T, which is



G. H a j ó s (Budapest): *Über eine Art von Graphen.*

Ist auf einer Geraden eine endliche Anzahl von Intervallen gegeben, so kann dieser Intervallmenge in folgender Weise ein Graph zugeordnet werden: einem jeden Intervall entspreche ein Knotenpunkt des Graphen, und zwei Knotenpunkte seien dann und nur dann durch eine Kante verbunden, wenn die entsprechenden Intervalle einander wenigstens teilweise überdecken. Der Vortrag behandelt die Frage, ob ein gegebener abstrakter Graph mit einem der eben gekennzeichneten Graphen isomorph ist, wofür Bedingungen angegeben werden. Für den Fall, daß diese erfüllt sind, wird die Frage der Rekonstruktion der Intervallmenge behandelt.

Figure 1.5: The abstract of Hajós' talk [172] in the 4th Austrian Mathematical Congress, and our translation:

G. Hajós (Budapest): About a type of graphs

If a finite number of intervals is given on a straight line, then a graph can be assigned to this set of intervals: each interval corresponds to a node of the graph, and two vertices are connected by an edge if and only if the corresponding intervals at least partially cover each other. The lecture deals with the question whether a given abstract graph is isomorphic with one of the graphs just described, for which conditions are specified. When these are fulfilled, the question of the reconstruction of the set of intervals will be dealt with.

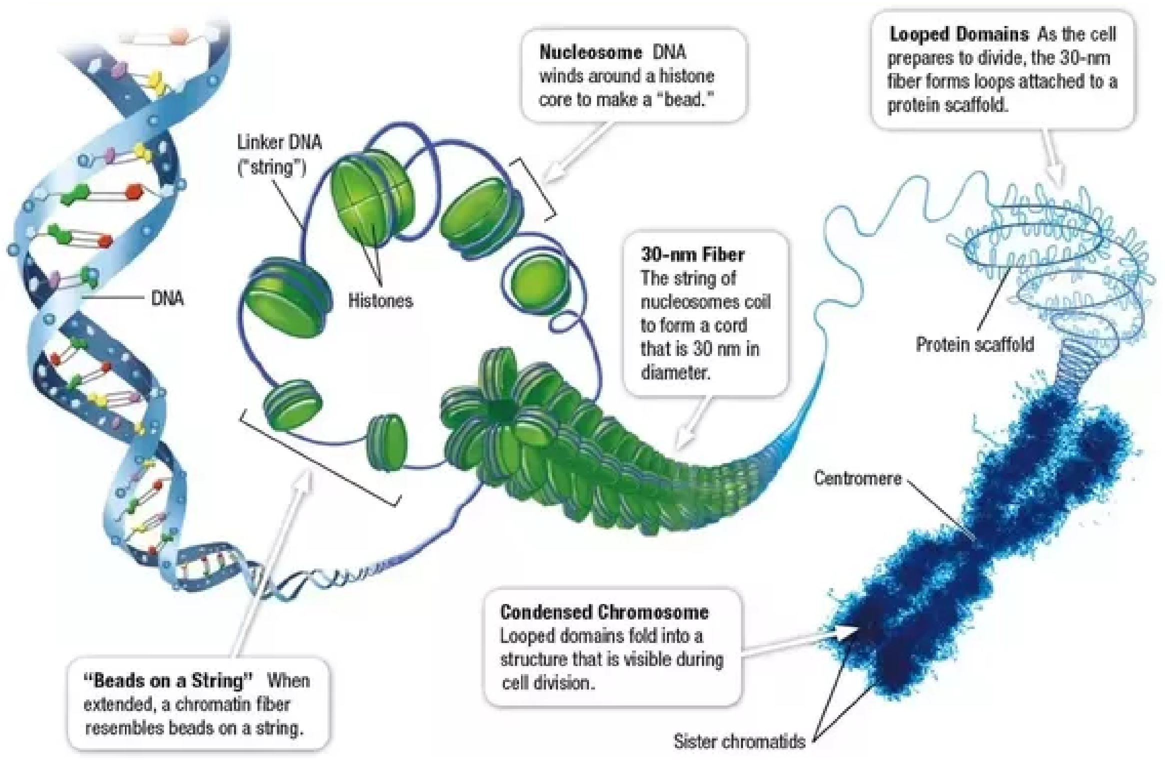

Nucleosome DNA winds around a histone core to make a "bead."
Looped Domains As the cell prepares to divide, the 30-nm fiber forms loops attached to a protein scaffold.
Linker DNA ("string")
DNA
Histones
30-nm Fiber The string of nucleosomes coil to form a cord that is 30 nm in diameter.
Protein scaffold
Centromere
Condensed Chromosome Looped domains fold into a structure that is visible during cell division.
"Beads on a String" When extended, a chromatin fiber resembles beads on a string.
Sister chromatids

Figure 1.6: A figure from Quora.com (`http://goo.gl/iM89uE`). An illustration of the structure of a chromosome down to the molecular level of DNA.

now known as Chargaff's rule. Watson and Crick also used an image from the X-ray crystallography work of Franklin and Wilkins, without knowledge of Franklin [266]. The double-helix model of Watson and Crick was not widely known or accepted till the end of the 1950's, when their conclusions were experimentally confirmed. Together with Wilkins, they were awarded the Nobel Prize in Physiology or Medicine in 1962. Figure 1.6 depicts the current understanding of the structure of chromosomes.

Benzer's Study. See [176, 199, 161, 354] for details about Benzer's life and influence in biology. He originally studied physics and his early work was related to germanium crystals which were later used in semiconductors. After reading Schrödinger's What is Life? [319], he got fascinated in the emerging area of molecular genetics. His unique background in physics allowed him to construct a revolutionary approach in the field. Also, he was one of the pioneers of neurogenetics and behavioral genetics.

In 1952, Benzer noticed a surprising property of *Enterobacteria phage T4*. Bacteriophages are viruses that infect and replicate in bacteria, and T4 infect bacteria *Escherichia coli*. One type of mutation of T4 called r (for rapid) can be easily identified because it produced large plaque compared to the wild type virus. Specific regions of T4 chromosomes were known, called rI, rII, and rIII. Benzer found that rII mutants failed to grow on the strain K of bacteria. He gave the following description of this discovery: "Well, at first I thought I made a mistake. I thought I had forgotten to put the phage on there. Dummkopf, do it again! I did it again and saw the same phenomenon. So I immediately realized—a Eureka moment—that this was a system in which I could do very fine genetic mapping!" [176].

In his experiment, he would infect the strain K placed in a Petri dish with two different rII mutants. Themselves, they would produce no plaque. But if there would be a recombination which created the wild type, plaque would be visible. Such recombination would be only possible if these two rII mutations would not overlap, which could be detected. Benzer [23] published his findings in 1955 and used them to map the region rII. Nowadays, this is called T4 rII system.

From Genes to Topological Spaces. In 1959, Benzer [24] published his work on the topology of fine genetic structure, which we describe below. Genes are the molecular unit of heredity and nowadays it is known that they are regions of DNA. But in early 1950's, not much was known and they were considered to be discrete entities of chromosomes, indivisible by recombination and arranged like beads on a string. Morgan [283] showed that chromosomes are linear arrangements of genes. Benzer writes: "At this finer level, within the 'gene' the question arises again: what is the arrangement of the *sub*-elements? Specifically, are *they* linked together in a linear order analogous to the higher level of integration of the genes in the chromosome?"

Benzer pointed out that previous research was dealing with distances between close changes in the structure. However, the distances sometimes behaved in a complicated manner and they should not be taken as the criterion for linear arrangement. Benzer argued that the question should be studied from the point of topology since the connections between parts matter, not distances. In his experiment, he therefore asked qualitative questions instead of quantitative ones. He studied the topology of the region rII and assumed that every mutation changes a connected part of the re-

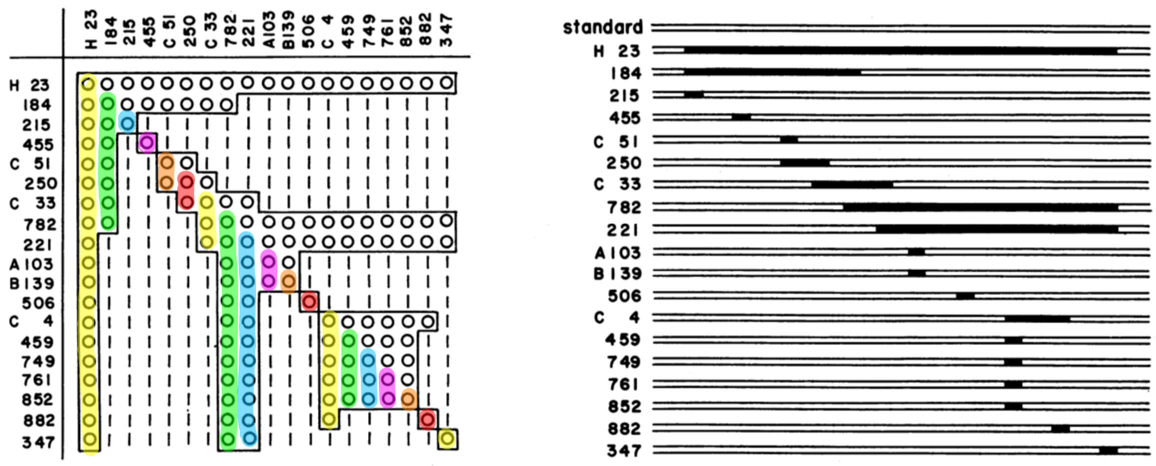


Figure 1.7: The figure from [24, Fig. 5 and 6]. On the left, we have added highlighting of consecutive zeroes on and below the diagonal.

gion. From his experiment in [23], he was able to determine which pairs of mutations intersect and which pairs do not.

Benzer described the data by matrices: for n mutations, he constructed an n-by-n matrix A having $a_{i,j} = 1$ if and only if the mutations i and j did not intersect (so they could produce the wild type by recombination), and $a_{i,j} = 0$ otherwise; see Fig. 1.7 on the left. The mathematical question was to find which topology is determined by these intersections, as Benzer writes: “It would be an interesting mathematical problem to derive the characteristic feature of the matrix for each kind of topological space.”

Benzer computed the data for 145 rII mutants of T4. He wanted to determine whether the region rII has linear topology and he nicely sums up the main idea:

> “Consider now a connected standard structure of linear topology without branches or loops: a perfect tape recording of a piece of music. Such a structure can be rendered unacceptable by a blemish—one false note, perhaps, or a blank interval (due to a jump of the tape, say). Given two independent ‘mutant’ versions, it may be possible to fabricate a standard one, but only if the two blemishes do not overlap. [...] Now if various mutant versions are tried two-by-two, in each case noting only whether or not successful recombination is possible, a new sort of result may be found. A blemish in the recording can involve a segment of the structure [...]. It may therefore occur that one mutation intersects two others that do not themselves intersect. Given enough defective versions, the yes-or-no results of recombination experiments would enable one to construct a linear map showing the various defects in their relative positions within the standard structure.”

The data have linear topology if each mutation can be represented by an interval and these intervals intersect exactly as described in the matrix A. Benzer states that A has this property if and only if there exists a symmetric reordering PAP^{T} for some permutation matrix P such that in each column the zeroes in the diagonal and below

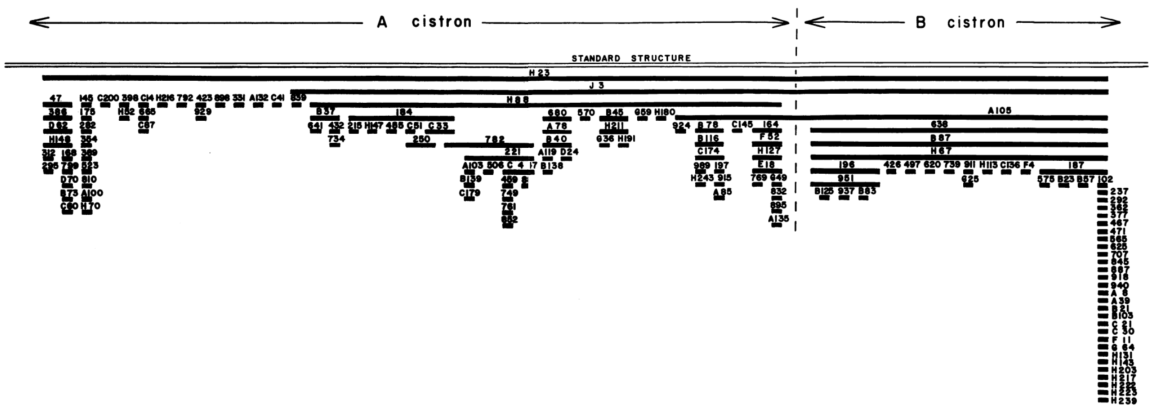


Figure 1.8: The figure from [24, Fig. 8]. Interval representation of the interval graph formed by 145 rII mutants of T4.

it appear consecutively (see the highlighting in Fig. 1.7 on the left). This reordering of rows/columns determines the left-to-right ordering of left endpoints in an interval representation which can be easily derived from the reordered matrix; see Fig. 1.7 on the right.

For his data, Benzer was able to determine a representation of 145 mutations by intervals, as depicted in Fig. 1.8. In the conclusions, Benzer comments his findings based on Occam's razor: "It is in the nature of the present analysis that the existence of complex situations cannot be disproved. However, the fact of the matter is that a simple linear model suffices to account for the data." Further, Benzer described two simple matrices which cannot be reordered as described above, so their appearance in the data would imply non-linear topology, see Fig. 1.9.

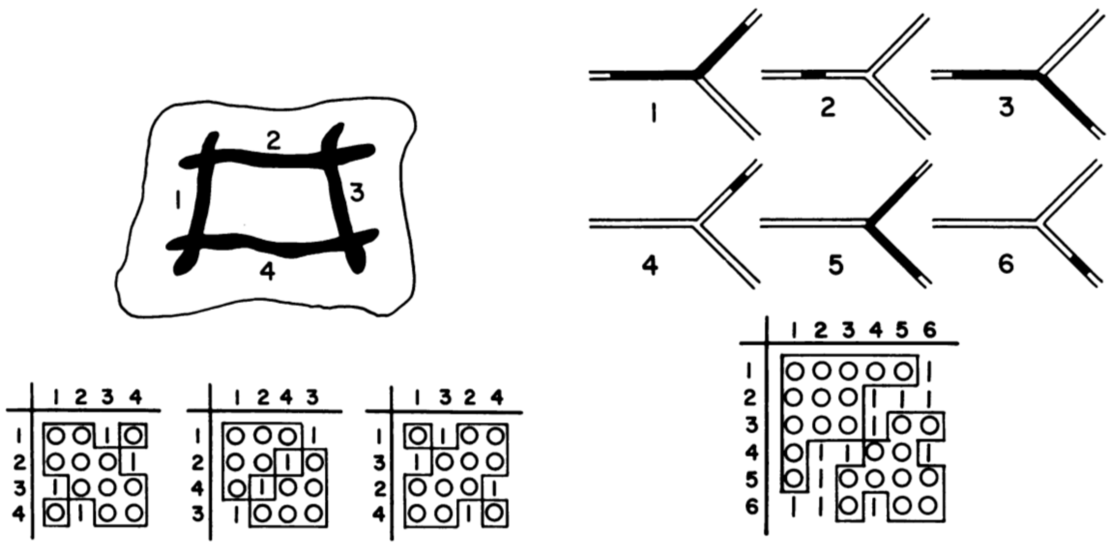


Figure 1.9: The figure from [24, Fig. 2 and 3]. Two simple structures which do not have linear topology and cannot be realized by intervals. On the left, a 2-dimensional structure, on the right, a branched structure.

Benzer does not mention any graphs in his paper [24]. Instead of a matrix, we may represent the data by a graph G. The vertices $\boldsymbol{V}(G)$ correspond one-to-one to mutations and we have an edge between two mutations if and only if they intersect. An *interval representation* of G is a collection of intervals $\bigl\{\langle u\rangle : u \in \boldsymbol{V}(G)\bigr\}$ such that $uv \in \boldsymbol{E}(G)$ if and only if $\langle u\rangle \cap \langle v\rangle \neq \emptyset$. A graph is called an *interval graph* if it can be represented by intervals in this manner. Benzer's result [24] states that the graph constructed from the data for 145 rII mutants of T4 is an interval graph.

Benzer's surprising biological applications stimulated research in intersection graphs and in particular interval graphs. Figure 1.9 gives two forbidden induced subgraphs for interval graphs: C_4 and 1-net; the full characterization of minimal forbidden induced subgraphs was given by Lekkerkerker and Boland [249] already in 1962. In the early 1960's, two other papers [133, 149] gave different characterizations of interval graphs. In 1976, Booth and Lueker [39] solved a long-standing open problem of recognizing interval graphs in linear time. Also, intersection graphs of branching structures (Fig. 1.9 on the right) were characterized as chordal graphs [45, 145].

Benzer's huge influence in this area of graph theory is best evidenced by citations of his works. His 1955 paper [23] with T4 rII system has over 600 citations in Google Scholar, mostly in biology. On the other hand, his 1959 paper [24] with interval graphs has over 400 citations, with a large fraction of them related to graph theory and interval graphs.

We note that despite the initial motivation in biology, interval graphs never found much application there. For some time, interval graphs played an important role in DNA hybridization [204], in which short pieces of DNA are studied independently. One of the reasons is that the methods for constructing interval representations were never able to cope with errors in the data. In many situations, all pairs of pieces of DNA cannot be compared, so we would have to construct an interval representation only for the partial data which lead to *interval probe graphs* [364, 276]. Also, other more efficient experimental methods were developed. Nevertheless, Benzer's application of interval graphs in his study of fine genetic structure [24] is an important and beautiful historical milestone in genetics.

1.1.3 Riemann Mapping Theorem and Circle Packings

The Riemann Mapping Theorem is a surprising result of global flexibility of conformal mappings in complex analysis. For instance, it implies that the unit square can be bijectively mapped to the unit disk while locally preserving angles and ratios of lengths. We describe its connection to circle packing which are representations of planar graphs by touching disks. To reveal the importance of conformal mappings, we first introduce complex numbers and complex analysis. Our approach is inspired by the excellent visual treatment of complex analysis in [287].[1]

Geometry of Complex Numbers. We first sketch the historical development, see [287, 277] for more details. In 1545, complex numbers first appeared in the context

[1]I would like to thank Petr Kratochvíl for introducing me to this book several years back and to the students of my complex analysis class based on this book in Spring 2016.

of roots of negative numbers, when Italian mathematician Cardano described formulas for solving cubic equations. In 1572, Bombelli gave the first substantial calculations using them.

For more than 250 years, there was not much interest in studying such strange numbers. For instance, Cardano used the description "subtle as they are useless". Descartes coined the derogatory term imaginary for them: "For any equation one can imagine as many roots [as its degree would suggest], but in many cases no quantity exists which corresponds to what one imagines." [32]. Even in 1770, Euler called imaginary numbers "impossible".

This attitude completely changed at the beginning of the 19th century when independently Wessel, Argand and Gauss recognized the geometry of complex numbers and thus completely revealed their meaning. Shortly after, in works of Cauchy, Riemann and others, the main body of complex analysis was constructed.

The geometry works as follows. The complex numbers correspond to points (or vectors) in a plane called the *complex plane* $\mathbb{C}$. Each complex number can be represented as $a+bi$ in the Cartesian coordinates or $re^{\varphi i} = r(\cos\varphi + i \sin\varphi)$ in the polar coordinates. Three basic operations have the following meaning as transformations of $\mathbb{C}$:

- The *addition* $z \mapsto z + A$ translates $\mathbb{C}$ by the vector A.
- The *multiplication* $z \mapsto z \cdot A$ for $A = re^{\varphi i}$ scales $\mathbb{C}$ by the factor r and rotates it by the angle φ. In [287], the excellent term *amplitwist* is coined since the multiplication is a combination of amplification (scaling) and twisting (rotation).
- The *conjugation* $z \mapsto \bar{z}$ reflects $\mathbb{C}$ along the real axis.

By combining these operations, we can describe all similarity transformations of the Euclidean plane. Suddenly, complex numbers are not imaginary numbers anymore, but they give a powerful arithmetic for these transformations [287, Section 1.IV]. Alternatively, complex numbers can be represented by 2-by-2 real matrices:

$$a + bi = re^{\varphi i} \qquad \text{is represented by} \qquad \begin{pmatrix} a & -b \\ b & a \end{pmatrix} = \begin{pmatrix} r\cos\varphi & -r\sin\varphi \\ r\sin\varphi & r\cos\varphi \end{pmatrix}. \tag{1.4}$$

The three basic operations correspond to matrix addition, matrix multiplication and matrix transposition, respectively.

But in many situations we still work with complex numbers and complex analysis almost magically, as going against our intuition. Consider the following two statements:

(i) There exist complex roots for every negative real number, e.g., $\sqrt{-1} = \pm i$.
(ii) Every real polynomial $P(x) = a_n x^n + \cdots + a_1 x + a_0$ has n complex roots.

These results are surprising only if we try to interpret them with the algebraic understanding of complex numbers as $a + bi$ from the 16th century. If we instead use the geometric view, then everything becomes immediately clear.

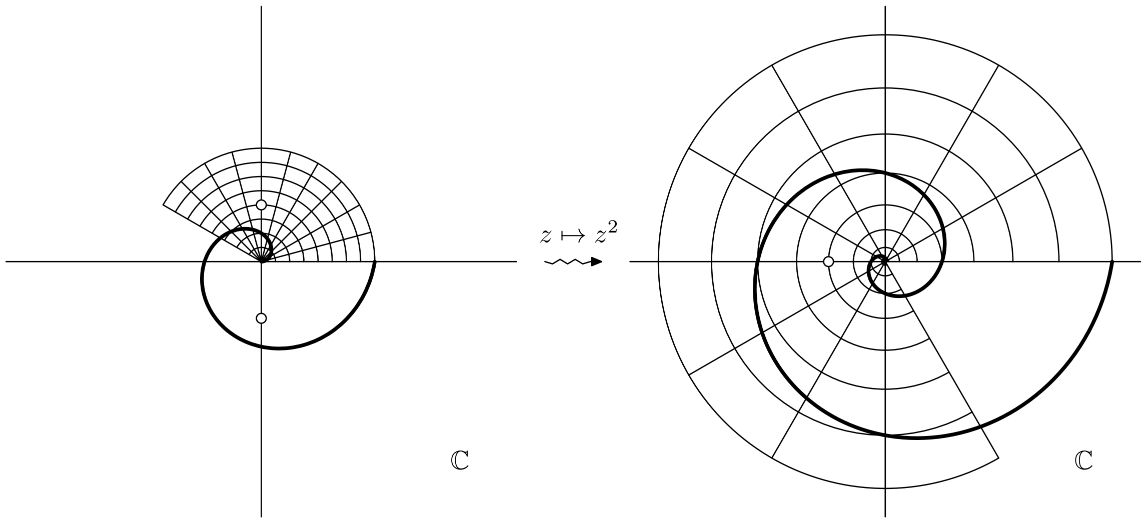


Figure 1.10: The mapping $z \mapsto z^2$ as a geometric transformation of the complex plane $\mathbb{C}$. On the left, a grid (in the polar coordinates), a spiral and two points i and $-i$. On the right, their images are shown, which gives a good insight into the geometric nature of the transformation. Notice that the spiral on the left winds once around the origin, but its image on the right winds twice around it.

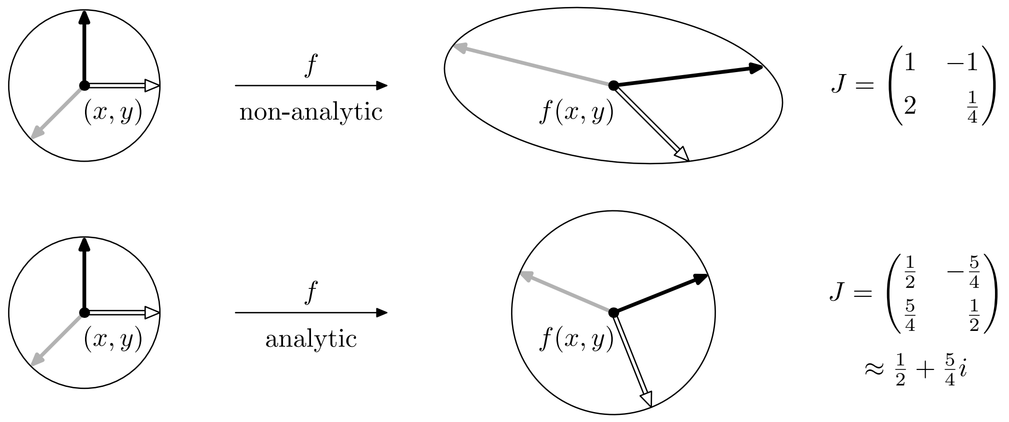
Concerning (i), the main point is that the mappings $x \mapsto x^2$, for $x \in \mathbb{R}$, and $z \mapsto z^2$, for $z \in \mathbb{C}$, are completely different, and just our notation makes them seem similar. Geometrically, x^2 is the area of a square with sides x, so it is non-negative, and the mapping has as the graph the well-known parabola. On the other hand, for $z = re^{\varphi i}$, we have $z^2 = r^2 e^{2\varphi i}$, so the length is squared but the angle is doubled [287, Section 2.II]. Geometrically, the transformation $z \mapsto z^2$ winds the complex plane $\mathbb{C}$ twice around the origin, as depicted in Fig. 1.10. Then, it is not surprising that $\sqrt{-1} = \pm i$, since the transformation maps the both complex numbers i and $-i$ to -1.

Concerning (ii), we can generalize the above for the mapping $z \mapsto z^n$. It is a transformation which n-times winds the complex plane $\mathbb{C}$ around the origin. A complex polynomial $P(z)$ is some combination of these transformations: a_0 for a translation, $a_1 z$ for an amplitwist, and $a_2 z^2$ till $a_n z^n$ winds $\mathbb{C}$ around the origin twice to n-times. Therefore, the mapping covers almost every point of $\mathbb{C}$ by n layers of $\mathbb{C}$, and the number of roots is at most n. (To understand it in more details, we would need to deal with multiple roots, which are related to branch points of multifunctions; see [287, Section 2.VI].) Certainly, this is not a proof of the Fundamental Theorem of Algebra, but it at least explains why it makes sense that every complex polynomial has n complex roots.

Analytic Functions and Conformal Mappings. We first describe derivatives of complex functions $f : \mathbb{C} \to \mathbb{C}$ [287, Chapter 4]. As in real analysis, the derivative is the best local "linear" approximation:

$$f(z+h) \approx f(z) + f'(z)h, \qquad \text{for } h \text{ small.}$$

Suppose that we view f as a mapping $f : \mathbb{R}^2 \to \mathbb{R}^2$, so $f(x,y) = (u(x,y), v(x,y))$ for some mappings $u, v : \mathbb{R}^2 \to \mathbb{R}$. Then the best local linear approximation is given



f
non-analytic
(x, y)
f(x, y)
$J = \begin{pmatrix} 1 & -1 \\ 2 & \frac{1}{4} \end{pmatrix}$
f
analytic
(x, y)
f(x, y)
$J = \begin{pmatrix} \frac{1}{2} & -\frac{5}{4} \\ \frac{5}{4} & \frac{1}{2} \end{pmatrix}$
$\approx \frac{1}{2} + \frac{5}{4}i$

Figure 1.11: On the top, the Jacobian J of a general mapping f maps an infinitesimal circle around (x, y) to an infinitesimal ellipse around $f(x, y)$. On the bottom, the Jacobian J of the complex derivative $f'(z)$, $z = x + yi$, is an amplitwist, mapping an infinitesimal circle around (x, y) to an infinitesimal circle around $f(x, y)$.

by the Jacobian matrix

$$J = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}$$

and we have

$$f(x + h, y + h') \approx f(x, y) + J \begin{pmatrix} h \\ h' \end{pmatrix}, \qquad \text{for } h,\ h' \text{ small.}$$

In general, an infinitesimal circle around (x, y) is mapped to an infinitesimal ellipse around $f(x, y)$; see [287, Section 4.IV] and SVD in [334].

When can the Jacobian J be represented by a complex number $f'(z)$, for $z = x + yi$? Exactly when J corresponds to an amplitwist, i.e., it is of the form (1.4). Therefore, the complex derivative $f'(z)$ exists if and only if the well-known Cauchy-Riemann equations are satisfied:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \qquad \text{and} \qquad \frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}.$$

Geometrically, existence of a derivative means that an infinitesimal circle around (x, y) is mapped to an infinitesimal circle around $f(x, y)$, so the mapping f locally preserves the geometry: angles and ratios of lengths [287, Section 4.IV]. For comparison, see Fig. 1.11.

A mapping preserving angles in some region is called *conformal* and this property is implied by non-zero derivatives in this region. For instance, $z \mapsto z^2$ of Fig. 1.10 has the derivative $2z$ and it is conformal everywhere except in the origin. Analytic functions, which are nice functions $f : \mathbb{C} \to \mathbb{C}$, are in the center of complex analysis. There are several equivalent definitions when what it means f to be analytic, for instance by the existence of derivatives or that f is conformal in some region. Therefore, the study of conformal mappings is important.

Riemann Mapping Theorem. We can think of the property of analytic functions that infinitesimal circles are mapped to infinitesimal circles as "local rigidity" [330]. In 1851, Riemann [302] described the following theorem which proves their "global flexibility": for any two simply connected regions U and V in the plane (which are non-empty and different from the plane), there exists a conformal bijective mapping from U to V. Since composition of conformal bijective mappings is a conformal bijective mapping, it is sufficient to prove that there exists a conformal bijective mapping f from U to the unit disk D.

We sketch the main idea of Riemann's original proof [302]. We note that his treatment is not completely rigorous [160] and there are other methods to prove it, e.g. [49]. But crucially, Riemann's idea of what he called the *Dirichlet principle* profoundly influenced mathematics.

We want to construct a conformal bijective mapping f from U to D such that $f(z_0) = 0$. The trick is to guess that

$$f(z) = (z - z_0)e^{g(z)}$$

for some analytic function $g(z) : \mathbb{C} \to \mathbb{C}$. We have $g(z) = u(z) + v(z) \cdot i$, where $u, v : \mathbb{C} \to \mathbb{R}$ are the real part of g and the imaginary part of g, respectively.

First, we deal with the real part $u(z)$ only. For every $z \in \partial U$, we have $|f(z)| = 1$, which determines that $u(z) = -\log|z - z_0|$; see [287, Chapter 2] for properties of complex exponentials and logarithms. Recall the Laplace operator from Section 1.1.1. Since g is analytic, the real part u is a harmonic function and satisfies the Laplace equation $\Delta u = 0$ on the interior $\mathring{U}$; see [287, Section 11.III]. The Dirichlet principle states that u prescribed on the boundary ∂U is uniquely determined on $\mathring{U}$ by $\Delta u = 0$ as the function minimizing the Dirichlet energy.

So the real part $u(z)$ of $g(z)$ is constructed. From the assumption that the region U is simply connected, it follows that $v(z)$ is the so-called harmonic dual of $u(z)$ and is uniquely determined by the local rigidity of analytic functions [287, Chapter 12].

We cannot leave this topic without pointing out a connection with Section 1.1.1 where the Laplace operator Δ and its discretization the Laplacian matrix L_G were discussed. Solving the Laplace equation $\Delta u = 0$ with the boundary conditions corresponds to solving the discretized linear system $\widehat{L_G}\boldsymbol{u} = \boldsymbol{b}$ as in (1.2). The discrete version of the Dirichlet principle is that solving this system is equivalent with energy minimization in (1.3). This discretization can be used to compute approximate conformal bijections [35] and has many applications in applied mathematics.

Instead of existence, we can ask what are all possible conformal bijective maps from U to D. Notice that if f and g are two such maps, then the composition $f \circ g^{-1}$ is a conformal automorphism of D, and vice versa. So it is sufficient to identify all conformal automorphisms of D, forming the automorphism group $\mathrm{Aut}(D)$.

Möbius transformations are transformations of the form

$$z \mapsto \frac{az + b}{cz + d}$$

for some $a, b, c, d \in \mathbb{C}$. They generalize rotations, translations, scalings and other geometric transformations, and they are related to circular inversions [287, Chapter

3]. For instance, there exists a unique Möbius transformation sending any three points to any other three points. In [287, Sections 3.IX and 7.VII], it is proved that Aut(D) consists of a 3-parametric family of Möbius transformations

$$z \mapsto e^{\varphi i} \frac{z-a}{\bar{a}z-1},$$

where a is an arbitrary point of $\mathring{D}$ which is mapped to 0, and φ is an arbitrary angle of the rotation.

Approximation by Circle Packings. A circle packing is a configuration of disks in the plane with pairwise disjoint interiors. It is a contact representation of the following graph G. The vertices $\boldsymbol{V}(G)$ correspond one-to-one to the disks of the packing. We have $uv \in \boldsymbol{E}(G)$ if and only if the disks corresponding to u and v are touching, i.e., they have a common point on their boundaries. For an example, see Fig. 1.12.

We can easily observe that every such graph G represented by a circle packing is necessarily planar. (This is true for every contact representation of connected sets in the plane such that no three intersect in one point.) The celebrated Koebe-Andreev-Thurston Theorem [232, 339] states the converse: every planar graph has a contact representation by touching disks in the plane. Further, when G is a triangulation, this representation is unique up to Möbius transformations and reflections.

The Riemann Mapping Theorem proves that there always exists a conformal bijective mapping from U to D. But construction of such mappings is difficult, and no explicit formulas are given. For instance, when U is a square, a conformal bijective mapping cannot be described by elementary formulas [101].

In 1985, Thurston suggested a beautiful iterative approach to computing approximate conformal bijective mappings; see Figure 1.13 for details. Its correctness was proved by Rodin and Sullivan [309]. Compared to other methods, the speed of this method is limited because computing circle packings is difficult. But this method has further applications because it can be generalized to other situations; see [330] for a survey.

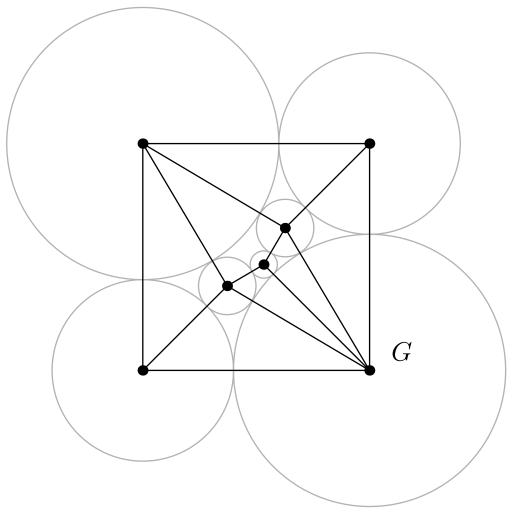


Figure 1.12: A circle packing representing the depicted planar graph G.

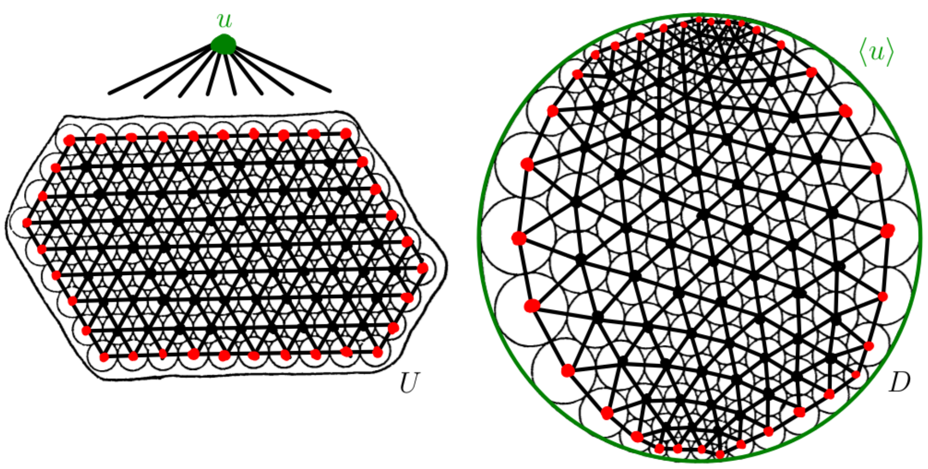


Figure 1.13: This is a modification of [309, Fig 1.1]. By the Riemann Mapping Theorem, there exists a conformal bijection from the region U to the disk D. Thurston's method computes its approximation by the depicted regular hexagonal circle packing. This packing is a contact representation of the planar graph drawn in bold. A triangulation is created from it by adding another vertex u (depicted in green) adjacent to all red vertices corresponding to the circles on the boundary. On the right, we have a circle packing representation of this triangulation where the outer circle $\langle u \rangle$ corresponding to the disk D represents the added vertex u. The circles on the left are mapped to circles on the right which defines an approximative conformal bijection from U to D. The key property proved in [309] is that equilateral triangles are mapped closer to equilateral triangles the further they are from the boundary.

1.2 Definitions

In this section, we give the key definitions used in the rest of this thesis. Many other definitions are given in the remaining chapters, and their location can be found using the Index.

For a graph G, we denote by $\boldsymbol{V}(G)$ its vertices and by $\boldsymbol{E}(G) \subsetneq \binom{\boldsymbol{V}(G)}{2}$ its edges. By n and m, we denote the number of vertices and edges of G, respectively. If we work with multiple graphs, we also use $\boldsymbol{v}(G) = |\boldsymbol{V}(G)|$ and $\boldsymbol{e}(G) = |\boldsymbol{E}(G)|$. We note that the notion of extended graphs is defined in Section 7.1 and used in most of Part II. For $A \subseteq \boldsymbol{V}(G)$, we denote by $G[A]$ the subgraph induced by A. We denote the *closed neighborhood* of x by $N[x]$, i.e., $N[x] = \{y : xy \in \boldsymbol{E}(G)\} \cup \{x\}$.

For a graph G, we denote the complement by $\overline{G}$ which is the graph such that $\boldsymbol{V}(\overline{G}) = \boldsymbol{V}(G)$ and $uv \in \boldsymbol{E}(\overline{G}) \iff uv \notin \boldsymbol{E}(G)$. For a class of graphs $\mathcal{C}$, we denote by co-$\mathcal{C}$ the class of all complements, i.e., co-$\mathcal{C} = \{G : \overline{G} \in \mathcal{C}\}$.

Geometric Representations. Let $\mathcal{C}$ be a class of graphs together with a class of geometric representations $\mathfrak{Rep}$. For a broader approach to graph representations, see the monograph [328]. For instance, both the adjacency matrix A_G and the Laplacean matrix L_G from Fig. 1.2 represent the graph G, meaning that the graph G can be derived from them. In the context of this thesis, $\mathfrak{Rep}$ either consists of planar embeddings or embeddings in surfaces of higher genus (defined at the beginning of Section 1.4), or

$\mathfrak{Rep}$ consists of intersection representations (defined at the beginning of Section 1.3).

The correspondence between representations and graphs is described by a multimapping $\rho : \mathfrak{Rep} \to \mathcal{C}$ such that each representation $\mathcal{R} \in \mathfrak{Rep}$ represents (encodes) the graphs $\rho(\mathcal{R}) \subseteq \mathcal{C}$. (If $\mathcal{C}$ contains exactly one graph for each isomorphism class, then ρ is a mapping and each $\rho(\mathcal{R})$ is a single graph.) For a graph $G \in \mathcal{C}$, we denote by $\mathfrak{Rep}(G)$ the class of all geometric representations of G from $\mathfrak{Rep}$, i.e.,

$$\mathfrak{Rep}(G) = \Bigl\{\mathcal{R} : \mathcal{R} \in \mathfrak{Rep}, G \in \rho(\mathcal{R})\Bigr\}.$$

We often require the property that every graph in $\mathcal{C}$ can be represented by a representation in $\mathfrak{Rep}$. Then the mapping ρ is surjective.

Recognition. For each graph class $\mathcal{C}$, there is a natural well-studied decision problem called *recognition*: does an input graph G belong to $\mathcal{C}$? In the context of geometric representations $\mathfrak{Rep}$, the recognition problem is the following decision problem:

Problem:	Recognition – $\textsc{Recog}(\mathcal{C}, \mathfrak{Rep})$
Input:	A graph G.
Question:	Is there a representation $\mathcal{R} \in \mathfrak{Rep}(G)$?

When a class of geometric representations $\mathfrak{Rep}$ considered for the class of graphs $\mathcal{C}$ is clear from the context, we may just write $\textsc{Recog}(\mathcal{C})$.

In this thesis, we are interested in structural properties of graph classes, coming from their geometric representations. We are mainly concerned with graph classes which can be recognized efficiently, meaning that there exists a polynomial-time algorithm for their recognition problem. The reason is that non-existence of such an algorithm (for instance, when the recognition problem is NP-hard, assuming P $\neq$ NP) implies that the structure of $\mathcal{C}$ is not very strong.

Moreover, efficient recognition algorithms are almost always able to certify "yes" answers by constructing geometric representations, which is the starting point of many other results and applications. In some situations, it is desirable to have a *certifying recognition algorithm* which also certifies "no" answers by finding some simple obstruction in the input proving non-existence of geometric representations for graphs which do not belong to the class. A precise definition of obstruction depends on the type of geometric representation. For instance, certifying recognition algorithms for planar graphs find subdivisions of K_5 or $K_{3,3}$ as obstructions [245, 351], while certifying recognition algorithms for interval graphs find one of the minimal forbidden induced subgraphs of Lekkerkerker and Boland [249, 255].

1.3 Intersection Representations

An intersection representation $\mathcal{R}$ of a graph G is a collection of sets $\Bigl\{\langle x \rangle : x \in \boldsymbol{V}(G)\Bigr\}$ such that $\langle x \rangle \cap \langle y \rangle \neq \emptyset$ if and only if $xy \in \boldsymbol{E}(G)$. Let $\mathfrak{Rep}$ be the class of all such representations. Since every graph has an intersection representation [269], interesting classes $\mathfrak{Rep}$ of intersection representations are obtained by restricting the representing sets to some nice classes of, say, geometric objects, e.g., continuous curves in plane, chords of a circle, convex sets, etc. For an overview of these classes, see the books [156, 275, 328].

For understanding most of Part I, interval graphs and their subclasses are essential, and they are described in Sections 1.3.1 and 1.3.2. Other classes of intersection graphs, described in the remainder of this section, mostly appear in Chapter 2 and the reader can refer to their description here only when needed.

1.3.1 Interval Graphs

The most famous class of intersection graphs are interval graphs. An *interval representation* of G is an intersection representation such that each $\langle u\rangle$ is a closed interval of the real line. A graph is called an *interval graph* if it has an interval representation, and we denote the class of interval graphs by INT; see Fig. 1.14a for an example. It is one of the oldest and most understood classes of graphs. As evidence of its popularity, Web of Knowledge lists more than 370 papers with the words "interval graph(s)" in the title.

Applications and Basic Properties. As stated in Section 1.1.2, they were introduced independently by Hajós [172] in 1957 and by Benzer [24] in 1959. Further applications include scheduling, psychology, archaeology, etc. [332, 305, 208]. They play an important role in models of time reasoning [155, 159] where relations between pairs of intervals are modeled by Allen algebras [3]; see Section 2.9.5 for more details.

Interval graphs also have nice theoretical properties. They are perfect (see Section 1.3.3) and closely related to path-width decompositions [38]. Feder and Hell [115] proved the following dichotomy for the complexity of the H-list homomorphism problem, where H has a loop at every vertex: the problem can be solved in polynomial time if H is an interval graph, and it is NP-complete otherwise; see also [83]. Interval graphs can be recognized in linear time [39, 235, 168, 75]. Many hard combinatorial problems including minimum vertex coloring, maximum clique, maximum independent set, and minimum dominating set become polynomial-time tractable for interval graphs.

Characterizations. In 1965, Fulkerson and Gross [133] characterized which rectangular 0-1 matrices have the *consecutive ones property*, which means that their rows can be reordered in such a way that 1's in each column appear consecutively. Using this, they described a fundamental characterization of interval graphs as exactly those

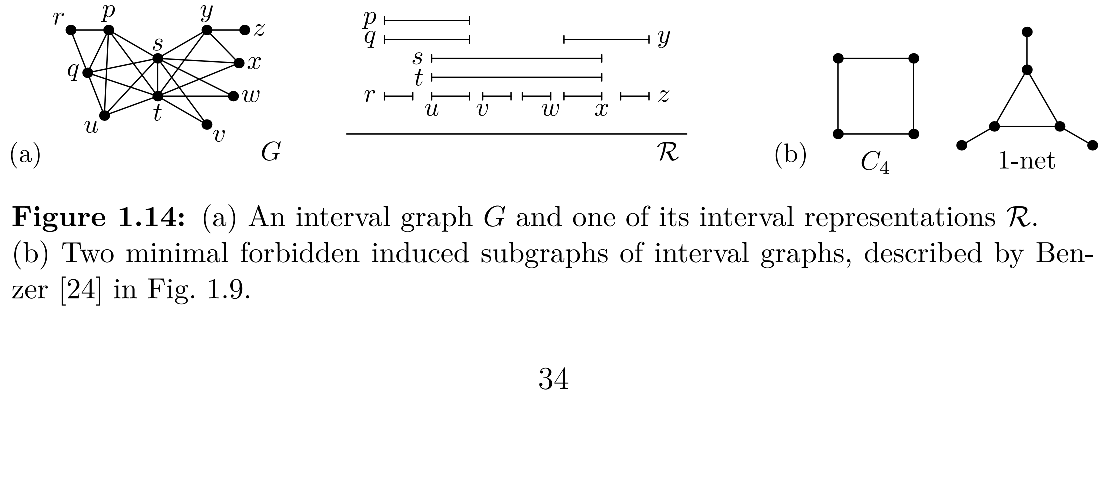


Figure 1.14: (a) An interval graph G and one of its interval representations $\mathcal{R}$. (b) Two minimal forbidden induced subgraphs of interval graphs, described by Benzer [24] in Fig. 1.9.

graphs whose incidence matrices between vertices and maximal cliques have consecutive ones property, which led to an $\mathcal{O}(n^3)$ recognition algorithm for interval graphs. In 1976, Booth and Lueker [39] described an efficient data structure called PQ-trees which gave the first linear-time algorithms for testing the consecutive ones property and recognizing interval graphs, and has many other applications, including linear-time recognition of planar graphs. We describe these results in detail in Section 3.1.

Another fundamental characterization of interval graphs was proved by Lekkerkerker and Boland [249] in 1962. They described the minimal forbidden induced subgraphs of interval graphs and characterized interval graphs as the intersection of chordal graphs and asteroidal triple-free graphs. We note that two of these forbidden induced subgraphs depicted in Fig. 1.14b were already identified by Benzer [24]. Computationally, these forbidden subgraphs can be found in linear time [255] (using PQ-trees), which gives a linear-time certifying algorithm for recognizing interval graphs: if "yes", a representation is constructed, if "no", a minimal forbidden induced subgraph is found. See Section 2.2 for more detail.

In 1964, Gilmore and Hoffman [149] characterized interval graphs by their relation to comparability graphs, which is linked to the reordering approach of Benzer [24], and we describe it in Section 1.3.4.

Notation. For a closed interval $\langle x \rangle$, we denote its left endpoint by $\ell(x)$ and its right endpoint by $r(x)$. If $r(x) < \ell(y)$, we say that $\langle x \rangle$ is *on the left* of $\langle y \rangle$ and $\langle y \rangle$ is *on the right* of $\langle x \rangle$. We say that $\langle y \rangle$ is *between* $\langle x \rangle$ and $\langle z \rangle$ if $\langle x \rangle$ is on the left of $\langle y \rangle$ and $\langle z \rangle$ is on the right of $\langle y \rangle$, or vice versa. In Chapters 3 and 4, we also work with open intervals, for which the inequalities in these definitions are non-strict.

Proper and Unit Interval Graphs. In 1969, due to a motivation in psychology, Roberts [304] introduced two subclasses of interval graphs. An interval representation is called *proper* if $\langle u \rangle \subseteq \langle v \rangle$ implies $\langle u \rangle = \langle v \rangle$, and *unit* if the length of all intervals $\langle u \rangle$ is one: $r(u) - \ell(u) = 1$. The classes of *proper* and *unit interval graphs* (denoted $\mathsf{PROPER\ INT}$ and $\mathsf{UNIT\ INT}$) consist of all interval graphs which have proper and unit interval representations, respectively. Roberts [304] proved that $\mathsf{PROPER\ INT} = \mathsf{UNIT\ INT}$ and characterized them as interval graphs without the claw $K_{1,3}$ as an induced subgraph. They can be recognized in linear time [257, 73], and we sketch more detail in Section 2.3.

Interval Orders. An interval representation also represents an associated partial order $<$ on the intervals called an *interval order*, introduced by Fishburn [123]. For intervals $\langle x \rangle$ and $\langle y \rangle$, we have $x < y$ if and only if $\langle x \rangle$ is on the left of $\langle y \rangle$. So both interval graphs and interval orders are represented by collections of intervals, and indeed they are closely related [127, 126].

The study of interval orders has the following motivation. Suppose that each interval describes when one event can happen in the timeline. If $a < b$, we know for sure that the event a happened before the event b. If two intervals intersect, we do not have any information about the order of the corresponding events. For more information, see the survey [341].

We note that already in 1956, Luce [261] introduced *semiorders* which are *unit*

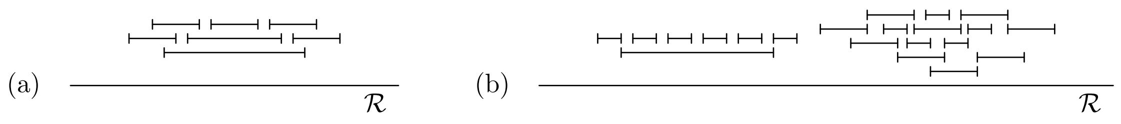


Figure 1.15: (a) An interval representation with nesting three. (b) The disjoint union of two components with minimum nesting two requiring three different lengths of intervals. On the left, the shorter intervals are shorter than $\frac{1}{4}$ of the longer ones. On the right, they are longer than $\frac{1}{3}$.

interval orders, i.e., interval orders representable by unit intervals. The motivation was in psychology. If the left endpoints $\ell(x)$ of the intervals $\langle x\rangle$ correspond to some evaluation, we are indifferent of values which are close to each other, but we can distinguish values which are far from each other: $x < y$ if and only if $\ell(x) + 1 < \ell(y)$. This is also the reason why proper/unit interval graphs are sometimes called indifference graphs. See an excellent book [297] for more detail.

1.3.2 k-nested and k-length Interval Graphs

We consider two hierarchies of subclasses of interval graphs which generalize proper and unit interval graphs. The class k-NestedINT consists of all interval graphs which have representations with no $k+1$ intervals $\langle u_0\rangle, \dots, \langle u_k\rangle$ such that $\langle u_0\rangle \subsetneq \langle u_1\rangle \subsetneq \cdots \subsetneq \langle u_k\rangle$; see Fig. 1.15a. The class k-LengthINT consists of all interval graphs which have representations having at most k different lengths of intervals; see Fig. 1.15b. We know by [304] that 1-NestedINT = PROPER INT = UNIT INT = 1-LengthINT.

For an interval graph G, we denote the minimum nesting over all interval representations by $\nu(G)$, and the minimum number of interval lengths by $\lambda(G)$. Since nested intervals have different lengths, we know that $\nu(G) \le \lambda(G)$ and this inequality may be strict (as in Fig. 1.15b). For each $k \ge 2$,

$$(k-1)\text{-LengthINT} \subsetneq k\text{-LengthINT} \subsetneq k\text{-NestedINT} \subsetneq (k+1)\text{-NestedINT}.$$

Fishburn [127, Theorem 5, p. 177] shows that graphs G in 2-NestedINT have unbounded $\lambda(G)$. Therefore, 2-NestedINT $\not\subseteq$ k-LengthINT for each k. Figure 1.16 depicts inclusions of considered classes.

Known Results and Motivation. The classes k-LengthINT were introduced by Graham as a natural hierarchy between unit interval graphs and interval graphs; see Fig. 1.16. Even after decades of research, the only results known are curiosities that

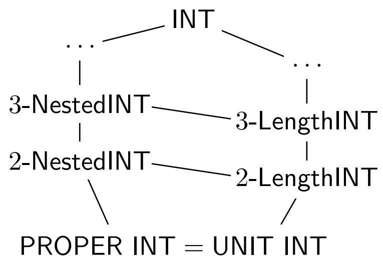


Figure 1.16: The Hasse diagram of proper inclusions of the considered classes.

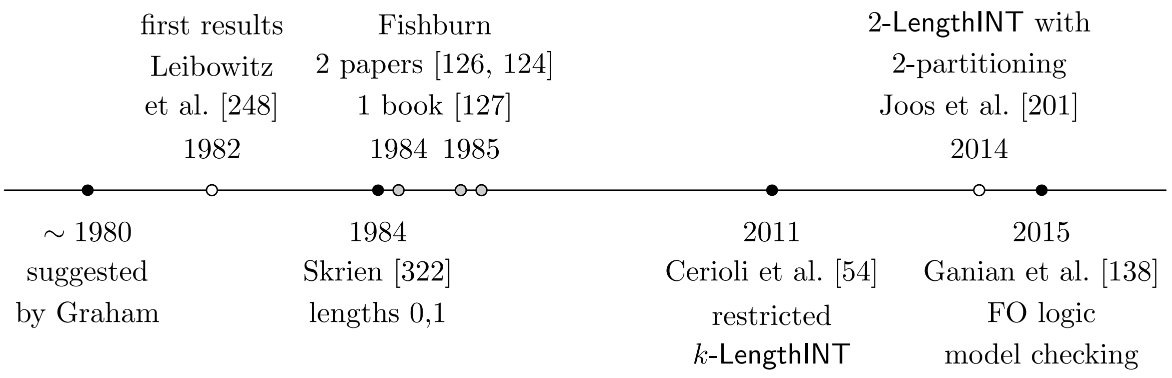


Figure 1.17: Timeline of results for k-LengthINT. Notice the big gap between 1985 and 2011.

illustrate the incredibly complex structure of k-LengthINT, very different from the case of unit interval graphs. For instance, k-LengthINT is not closed under disjoint unions; see Fig. 1.15b. Timeline of results is depicted in Fig. 1.17.

Leibowitz et al. [248] show that the class 2-LengthINT contains caterpillars, threshold graphs, and unit interval graphs with one additional vertex. Further, interval graphs G with $\lambda(G) > 2$ such that $\lambda(G \setminus x) \le \lambda(G) - 2$ for some $x \in V(G)$ are constructed in [248]. Fishburn [126] shows that there are infinitely many forbidden interval induced subgraphs for 2-LengthINT, while 1-LengthINT are interval graphs just without $K_{1,3}$ [304]. It is also known [124] that there are interval graphs in 2-LengthINT such that, when the shorter length is fixed to 1, the longer one can be one of the real numbers belonging to arbitrarily many distinct intervals of the real line, arbitrarily far from each other.

Not much is known about the computational complexity of problems involving k-LengthINT, even recognition is open for $k = 2$. In [54], a polynomial-time algorithm is given for computing $\lambda(G)$ for interval graphs G which are extended bull-free or almost threshold (which highly restricts them). Skrien [322] characterized 2-LengthINT which can be realized by lengths zero (points) and one (unit intervals), leading to a linear-time recognition algorithm. As most of the efficient algorithms for intersection graph classes require representations, there are almost no algorithmic applications of the fact that a given interval graph can be represented by k lengths.

All these difficulties lead us to introduce the other hierarchy of k-NestedINT which generalizes proper interval graphs; see Fig. 1.16. To the best of our knowledge, the only reference is Fishburn's book [127] in which the parameter $\nu(G)$ called *depth* is considered and linked to k-LengthINT. There are some different generalizations of proper interval graphs [300], which are less rich and not linked to k-LengthINT. We describe all results in Section 2.4 and Chapter 5.

1.3.3 Chordal Graphs

Chordal graphs were originally introduced in the context of perfect graphs by Berge [25, 26] in 1961 as graphs with no induced cycle of length four or more, and they also appeared in [171, 99, 133]; see Fig. 1.18. We denote the class of chordal graphs by

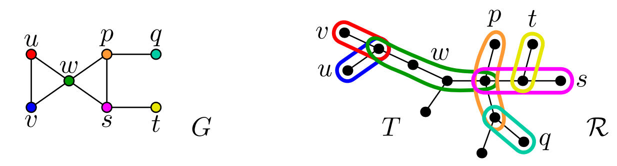


Figure 1.18: A chordal graph G on the left and one of its intersection representations $\mathcal{R}$ by subtrees of a tree T on the right.

CHOR. They are also known as *triangulated graphs* and *rigid circuit graphs*, since every induced cycle is triangulated. Rose et al. [311] describe a linear-time recognition algorithm for chordal graphs, based on an elegant application of lexicographic breadth-first search (LBFS).

Perfect Graphs. Let $\chi(G)$ denote the chromatic number of G, and let $\omega(G)$ denote the clique-number of G. Trivially we have $\omega(G) \le \chi(G)$ and the graphs for which every induced subgraph satisfies equality are *perfect graphs* [25, 26]. This graph class has very strong properties. The Weak Perfect Graph Theorem [259, 258] states that they are closed under complementation and the Strong Perfect Graph Theorem [65] asserts that a graph is perfect if and only if it contains no odd cycle and no complement of an odd cycle as an induced subgraph.

Perfect graphs are a superclass of many geometrically represented graph classes, and they are strongly related to several others, as evidenced by the title of Golumbic's book [156]. One example of subclass of perfect graphs are *trivially perfect graphs* which are interval graphs having a representations such that each pair of interval $\langle u \rangle$ and $\langle v \rangle$ is either disjoint, or one is subset of the other. Berge [25] observed that chordal graphs are perfect.

Perfect Elimination Schemes. An alternative definition is that chordal graphs are precisely graphs G which have a so-called *perfect elimination scheme* [310]. This is a linear ordering $v_1, \dots, v_n$ of $\boldsymbol{V}(G)$ such that for each v_i, the set $N[v_i] \cap \{v_i, v_{i+1}, \dots, v_n\}$ induces a complete graph. In other words, G is chordal if there exists a *simplicial vertex* $v \in \boldsymbol{V}(G)$ such that $N[v]$ induces a complete graph and $G \setminus v$ is a chordal graph. The recognition algorithm of [311] finds a perfect elimination scheme if it exists.

The name perfect elimination scheme was introduced in [310] because of a motivation in solving sparse symmetric positive definite linear systems $A\boldsymbol{x} = \boldsymbol{b}$. Such systems can be solved by Gaussian elimination without pivoting [334], but the elimination process may introduce additional non-zeroes making the matrix non-sparse [93, Section 2.7.4]. To preserve symmetricity of A, we may reorder the rows and the columns by a permutation matrix P and solve the linear system $PAP^{\mathsf{T}}\boldsymbol{y} = P\boldsymbol{b}$ instead; further, this reordering does not influence numerical properties of the matrix [349]. It is proved in [141] that it is NP-complete to find an ordering minimizing the number of introduced non-zeroes by the elimination. Rose [310] studied the existence of a perfect reordering for which the Gaussian elimination introduces no additional non-zeroes. He proves that it exists if and only if the matrix created from A by replacing non-zero coefficients with ones is the adjacency matrix A_G of a chordal graph G, and this reordering is a perfect elimination scheme of G.

Subtree in Tree Representations. In 1974, independently Buneman [45] and Gavril [145] described geometric representations of chordal graphs, linking them to Benzer's branching structures (Fig. 1.9 on the right). They proved that a graph G is a chordal graph if and only if there exists a tree T such that G has an intersection representation $\mathcal{R}$ of subtrees of T; see Fig. 1.18.

From these geometric representations, it immediately follows that $\mathsf{INT} \subsetneq \mathsf{CHOR}$. Also, treewidth decompositions are closely related to these subtree-in-tree representations [38].

Path Graphs. Path graphs were originally considered by Gavril [146]. A graph is a *path graph* if and only if there exists a tree T such that there exists an intersection representation $\mathcal{R}$ by subpaths of T; for instance, the graph G in Fig. 1.18 is a path graph. We denote this class as PATH and we get that

$$\mathsf{INT} \subsetneq \mathsf{PATH} \subsetneq \mathsf{CHOR}.$$

The fastest recognition algorithm for path graphs [316] runs in time $\mathcal{O}(mn)$.

1.3.4 Comparability Graphs and Related Geometric Graph Classes

In the early 1960's, Gillmore and Hoffman [149] introduced comparability graphs, and they were also independently studied in [361, 148]. Let $\rightarrow$ be an orientation of a graph G where $x \rightarrow y$ denotes that $xy \in \boldsymbol{E}(G)$ and it is oriented from x to y. An orientation is called *transitive* if $x \rightarrow y$ and $y \rightarrow z$ implies that $xz \in \boldsymbol{E}(G)$ and $x \rightarrow z$. A graph is a *comparability graph* if and only if it has a transitive orientation; see Fig. 1.19a for an example. In other words, every comparability graph is created from the directed graph of a poset (not a Hasse diagram) by removing the orientation of edges. We denote the class of comparability graphs by COMP.

Polynomial-time recognition algorithms for comparability graphs follow implicitly from [149, 148]. The first explicit algorithm described by Golumbic [154] runs in time $\mathcal{O}((n+m)\Delta)$ where Δ is the maximum degree. More insightful algorithms are based on modular decomposition, described in Section 2.7.1, which captures all transitive orientations of a comparability graph and has many other applications. It can be computed in linear time [77, 272, 337] and a transitive orientation can be constructed in linear time as well [272], if it exists. But it is non-trivial to test whether the constructed orientation is transitive, and the best current approach is to use matrix multiplication with complexity $\mathcal{O}(n^{\omega})$ (currently, $\omega \approx 2.3729$ [247]), so this is the complexity of current fastest recognition algorithm for comparability graphs.

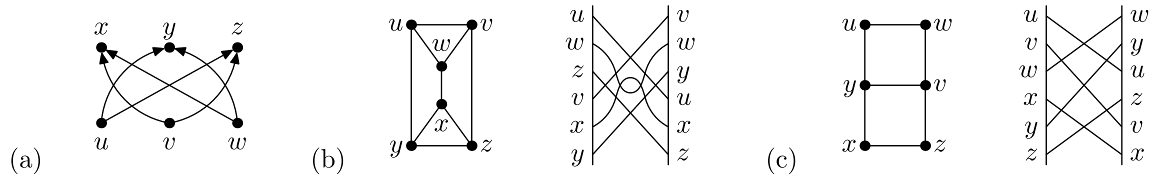


Figure 1.19: (a) A comparability graph with a transitive orientation. (b) A function graph and one of its representations. (c) A permutation graph and one of its representations.

Characterization of Interval Graphs. Gilmore and Hoffman [149] used comparability graphs to prove the following characterization of interval graphs. Let G be an interval graphs and let $\mathcal{R}$ be one of its interval representations. As explained in Section 1.3.1, $\mathcal{R}$ has an associated poset called an interval order in which non-intersecting pairs of intervals are ordered from left to right. In other words, the complement $\overline{G}$ is a comparability graph, and this was also independently proved in [148]. Gilmore and Hoffman proved that

$$\mathsf{INT} = \mathsf{co\text{-}COMP} \cap \mathsf{CHOR},$$

where $\mathsf{co\text{-}COMP}$ are the complements of comparability graphs, called *co-comparability graphs.*

Order Dimension. An important structural parameter of a poset P is its *Dushnik-Miller dimension* $\dim(P)$ [102], defined as the least number of linear orderings $L_1, \dots, L_k$ such that $P = L_1 \cap \cdots \cap L_k$. For a finite poset P, its dimension is always finite since P is the intersection of all its linear extensions. Similarly, we define the *dimension* of a comparability graph G, denoted by $\dim(G)$, as the dimension of any transitive orientation of G; every transitive orientation has the same dimension by [137]. By k-DIM, we denote the subclass consisting of all comparability graphs X with $\dim(X) \le k$. We get the following infinite hierarchy of graph classes:

$$1\text{-}\mathsf{DIM} \subsetneq 2\text{-}\mathsf{DIM} \subsetneq 3\text{-}\mathsf{DIM} \subsetneq 4\text{-}\mathsf{DIM} \subsetneq \cdots \subsetneq \mathsf{COMP}.$$

For instance, [317] proves that a graph G is planar if and only if the bipartite graph of the incidence between $\boldsymbol{V}(G)$ and $\boldsymbol{E}(G)$ belongs to 3-DIM. Yannakakis [362] proved that recognizing k-DIM, for $k \ge 3$, is NP-complete. Also, see the survey [357].

Function Graphs. Golumbic et al. [158] introduced *function graphs* as intersection graphs of graphs of continuous real-valued function on the interval $[0, 1]$. For each $u \in \boldsymbol{V}(G)$, we have a continuous function $f_u : [0, 1] \to \mathbb{R}$ and

$$\langle u \rangle = \Big\{(x, f_u(x)) : x \in [0, 1]\Big\}.$$

Alternatively, we have two parallel vertical lines, and each $\langle u \rangle$ is an x-monotone continuous curve from the left line to the right line. Figure 1.19b given an example. We denote the class of function graphs by FUN.[2]

Function graphs were introduced in [158] because of the following relation to comparability graphs:

$$\mathsf{FUN} = \mathsf{co\text{-}COMP}.$$

It is easy to see that every function graph is a co-comparability graph: we can orient the edges between non-intersection functions from the bottom function to the top one, and this orientation is clearly transitive.

For the other implication, let G be a comparability graph with $\boldsymbol{V}(G) = \{1, \dots, n\}$. The graph G has a transitive orientation, giving a poset P of a finite dimension k such that $P = L_1 \cap \cdots \cap L_k$ for some linear orderings $L_1, \dots, L_k$. We add $k - 2$ vertical

[2]Sometimes, it is more convenient to rotate these figures by 90 degrees, to have y-monotone curves between two horizontal lines; for instance, in Figs. 1.20 and 1.21.

lines in between, so we have vertical lines $\ell_1, \dots, \ell_k$. On each of these lines, we place n points corresponding one-to-one to the functions in some ordering. On ℓ_i, we place them as in the linear ordering L_i. By connecting the corresponding points by piecewise linear functions f_u, it is easy to prove that we get a function representation of $\overline{G}$.

Permutation Graphs. *Permutation graphs* is a subclass of function graphs which can be represented by linear functions [16]; see Fig. 1.19c for an example. We denote this class by PERM. Permutation graphs can be recognized in linear time [272]. Even, Pnueli, and Lempel [111] proved that

$$\mathsf{PERM} = \mathsf{COMP} \cap \mathsf{co\text{-}COMP} = 2\text{-}\mathsf{DIM}.$$

So G is a permutation graph if and only if both G and $\overline{G}$ can be transitively oriented. Below, we argue this characterization of permutation graphs.

Let G be a permutation graph. Each permutation representation defines two linear orderings L_1 and L_2 as the orderings of endpoints of $\langle u \rangle$ on two vertical lines. For instance, in Fig. 1.19c, these two linear orders are $L_1 = uvwxyz$ and $L_2 = wyuzvx$. Two vertices are adjacent if and only if their order differs in L_1 and L_2. Therefore, every permutation graph is $\mathsf{co}\text{-}2\text{-}\mathsf{DIM}$. Suppose that we reverse the ordering L_2 of the right line and get $L_2^{\leftrightarrow}$. Then we get a permutation representation of $\overline{G}$, so permutation graphs are closed under complementation: $\mathsf{PERM} = \mathsf{co\text{-}PERM}$ and $\mathsf{PERM} = 2\text{-}\mathsf{DIM}$. We get that $L_1 \cap L_2$ is a transitive orientation of $\overline{G}$ and $L_1 \cap L_2^{\leftrightarrow}$ is a transitive orientation of G. So $\mathsf{PERM} \subsetneq \mathsf{COMP} \cap \mathsf{co\text{-}COMP}$. The other direction is proved similarly.

Interval Dimension. Similarly to the Dushnik-Miller dimension, let the interval dimension $\mathrm{idim}(P)$ of a poset P be the least number of interval orderings $I_1, \dots, I_k$ such that $P = I_1 \cap \dots \cap I_k$. Since every linear ordering is also an interval ordering, we have $\mathrm{idim}(P) \le \dim(P)$. Similarly, for a comparability graph G, let $\mathrm{idim}(G)$ be the interval dimension of any transitive orientation; the value is the same for every transitive orientation [264]. Let k-IDIM denote the class of all comparability graphs of interval dimension at most k. We have k-$\mathsf{DIM} \subsetneq k$-$\mathsf{IDIM}$ and $\mathsf{INT} = \mathsf{co}\text{-}1\text{-}\mathsf{IDIM}$. For more information, see [357] and the references therein.

Trapezoid Graphs. In [81, 72], *trapezoid graphs* as intersection graphs of trapezoids with bases on two vertical parallel lines were introduced; see Fig. 1.20 for an example. We denote this class by $\mathsf{TRAPEZOID}$. It follows from [81] that

$$\mathsf{TRAPEZOID} = \mathsf{co}\text{-}2\text{-}\mathsf{IDIM},$$

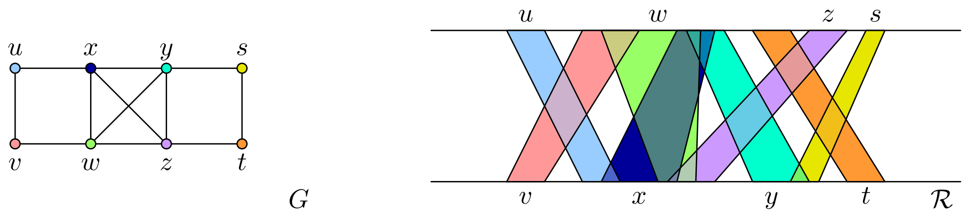


Figure 1.20: A graph G and one of its trapezoid representations $\mathcal{R}$. Observe that G is neither an interval graph, nor a permutation graph.

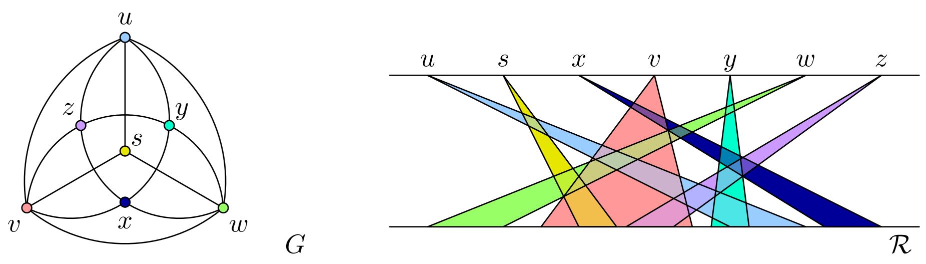


Figure 1.21: A graph G and one of its triangle representations $\mathcal{R}$. Again, G is neither an interval graph, nor a permutation graph.

so $\mathsf{PERM} \cup \mathsf{INT} \subsetneq \mathsf{TRAPEZOID} \subsetneq \mathsf{FUN}$. Trapezoid graphs can be recognized in time $\mathcal{O}(n^2)$ [264] (see also [63]), while recognizing k-IDIM, for $k \geq 3$, is NP-complete [362].

Triangle Graphs. In [72], a subclass of trapezoid graphs called *triangle graphs* was introduced, and we denote it by $\mathsf{TRIANGLE}$. We have two horizontal lines. Each triangle graph has an intersection representation, in which each set is represented by a triangle with one side on the bottom line and the third vertex on the top line; see Fig. 1.21 for an example. An alternative name for the class is *point-interval intersection graphs*. We have

$$\mathsf{INT} \cup \mathsf{PERM} \subsetneq \mathsf{TRIANGLE} \subsetneq \mathsf{TRAPEZOID}.$$

The complement of each triangle graph is a comparability graph whose poset can be represented as an intersection of a linear ordering with an interval ordering, so this graph class uses the concept of linear-interval poset dimension [53]. Very recently, the first polynomial-time recognition algorithms for triangle graphs were described [278, 336].

1.3.5 Circle Graphs

A *circle representation* $\mathcal{R}$ is an intersection representation in a circle such that each $\langle u \rangle$ is a chord of this circle. A graph is called *circle graph* if it has a circle representation; see Fig. 1.22a. We denote the class of circle graphs by CIRCLE. They were first considered by Even and Itai [109] in the early 1970s in the study of stack sorting techniques. Other motivations are due to their relations to Gauss words [89] (see Fig. 1.22b) and matroid representations [88, 42]. Circle graphs are important regarding rank-width [291]. Also, $\mathsf{PERM} \subsetneq \mathsf{CIRCLE}$.

Interval Overlap Graphs. Suppose that we pick an arbitrary point of the circle that is not an endpoint of a chord. We cut the circle at this point and straighten it into a segment; see Fig.1.23. From this straightening of the circle, each chord can now be seen as an arc above the resulting segment. Notice that two chords $\langle u \rangle$ and $\langle v \rangle$ cross if and only if their endpoints appear in the order *uvuv* or *vuvu* from left to right. Alternatively, circle graphs are called *interval overlap graphs*. Their vertices can be represented by intervals and two vertices are adjacent if and only if their intervals overlap which means they intersect and one is not a subset of the other.

Each circle representation $\mathcal{R}$ of G is associated with a unique circular word τ. The word τ is obtained by the circular order of the endpoints of the chords in $\mathcal{R}$ as

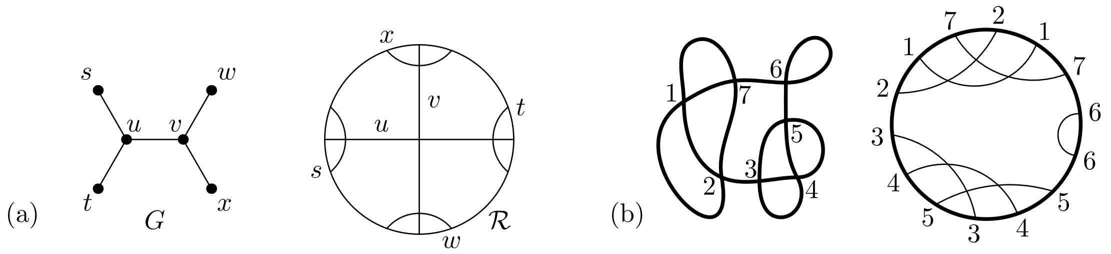


Figure 1.22: (a) A circle graph G with one of its representations $\mathcal{R}$. The chords are depicted as arcs to make the figure more readable. (b) A self-intersecting closed curve with n intersections numbered $1, \dots, n$ corresponds to a representation of circle graph with the vertices $1, \dots, n$ where the endpoints of the chords are placed according to the order of the intersections along the curve.

they appear along the circle when traversed clockwise, so each vertex appears twice in τ. The occurrences of u and v alternate in τ, if $uvuv$ is a circular subsequence of τ, which happens if and only if $uv \in \boldsymbol{E}(G)$. For example $\mathcal{R}$ in Fig. 1.22 corresponds to the circular word $\tau = susxvxtutwvw$.

Properties. In general, the difference between $\omega(G)$ and $\chi(G)$ can be arbitrarily high, e.g., there is a triangle-free graph with an arbitrary high chromatic number [284]. Circle graphs are known to be *almost perfect* which means that $\chi(G) \le f(\omega(G))$ for some function f. The best known result for circle graphs [236] states that $f(k)$ is $\Omega(k \log k)$ and $\mathcal{O}(2^k)$.

Some NP-hard problems, such as maximum weighted clique and independent set [147], become tractable on circle graphs. On the other hand, problems such as vertex colorability [139] and Hamiltonicity [84] remain NP-complete even for circle graphs.

Recognition. The complexity of recognition of circle graphs was a long standing open problem; see [328] for an overview. The first results, e.g., [109], gave existential characterizations which did not give polynomial-time algorithms. The mystery whether circle graphs can be recognized in polynomial time frustrated mathematicians for some years. It was resolved in the mid-1980s and several polynomial-time algorithms were discovered [41, 135, 285] (in time $\mathcal{O}(n^7)$ and similar). Later, a more efficient algorithm [327] based on *split decomposition* was given, and the current state-of-the-art recognition algorithm [150] runs in a quasi-linear time in the number of vertices and

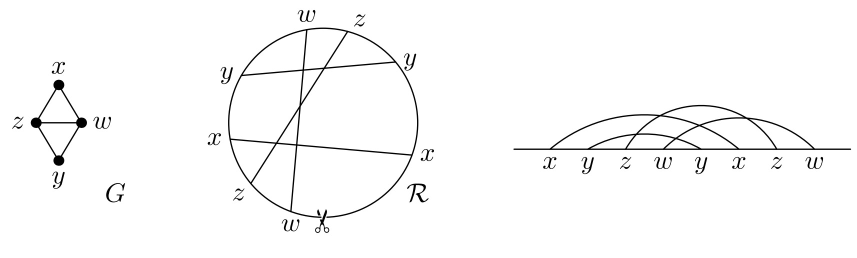


Figure 1.23: An example of a circle graph with a circle graph representation on the left; an interval overlap representation of the same graph on the right.

the number of edges of the graph. For more details about recognition, see Section 2.6.

1.3.6 Circular-Arc Graphs

In 1969, Klee [226] (see also [170]) introduced a generalization of interval graphs called *circular-arc graphs* which are intersection graphs of arcs of a circle; see Fig. 1.24. He asked what is the complexity of recognition. We denote the class of circular-arc graphs as CIRCULAR-ARC. Tucker [342] described an involved $\mathcal{O}(n^3)$ recognition algorithm. Currently, circular-arc graphs can be recognized in linear time [271, 203].

Circular-arc graphs were considered as a natural generalization of interval graphs, but they are much more involved. Fulkerson and Gross [133] characterized interval graphs by the existence of a certain ordering of maximal cliques (see Section 3.1), but circular-arc graphs may have exponentially many maximal cliques, and further arcs of maximal cliques do not satisfy the Helly property: a common intersection does not have to exist. Circular-arc graphs are very tricky, prone to making mistakes. For instance, Hsu [195] described a polynomial-time algorithm for graph isomorphism of circular-arc graphs, but a mistake was later found [80] and the complexity of graph isomorphism is still open on circular-arc graphs.

Subclasses. We quickly sketch several subclasses of circular-arc graphs, the reader may refer to the survey [253] and the references therein. Helly circular-arc graphs, denoted HELLY CIRCULAR-ARC, consist of all circular-arc graphs having circular-arc representations such that the arcs of every maximal clique satisfy Helly property. Therefore, their behaviour is much closer to interval graphs. Proper and unit circular-arc graphs, denoted PROPER CIRCULAR-ARC and UNIT CIRCULAR-ARC, generalize proper and unit interval graphs, respectively. Proper circular-arc graphs consist of all circular-arc graphs having circular-arc representations such that no arc is a proper subset of another arc. Unit circular-arc graphs consist of all circular-arc graphs having circular-arc representations in which all arcs have the same length. Unlike for interval graphs, UNIT CIRCULAR-ARC $\subsetneq$ PROPER CIRCULAR-ARC. Figure 1.25 depicts examples.

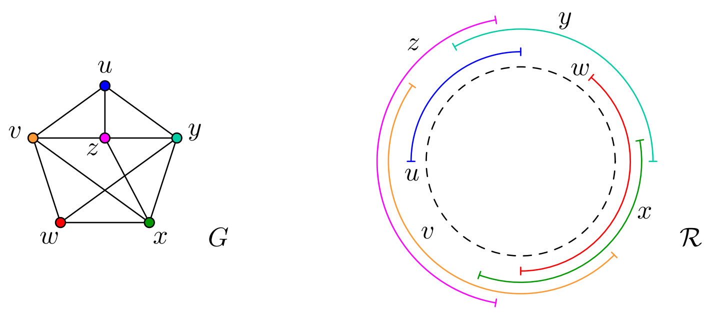


Figure 1.24: A circular-arc graph G and one of its circular-arc representations $\mathcal{R}$. The circle is depicted in dashed.

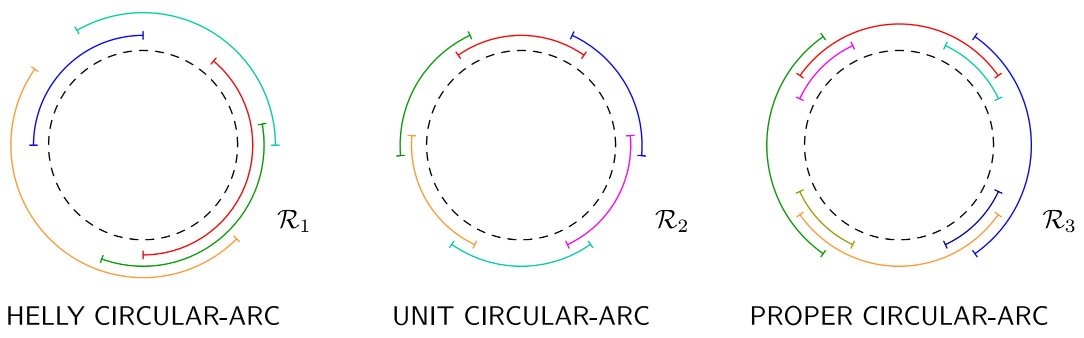


Figure 1.25: Three example representations of subclasses of circular-arc graph. Observe that the graph represented by $\mathcal{R}_3$ is not a unit circular-arc graph.

1.3.7 String Graphs

The class of string graphs consist of intersection graphs of curves, called strings, in the plane. We denote this graph class as STRING. It was first introduced in 1976 by Ehrlich, Even and Tarjan [104] who proved that 3-coloring is NP-complete for string graphs. The complexity of recognition was a mystery for quite some time. In 1991, Kratochvíl [237] showed that it is NP-hard. But it was unclear how to certify a string representation in polynomial space since a string graph may require exponentially many intersections in every string representation [238]. This question was resolved in 2002 when string graph recognition was proved to be in NP [314].

String graphs generalize most intersection classes of graphs. More precisely, it contains every intersection graph of arc-connected sets in the plane. Therefore, every result for string graphs also applies to all classes of intersection graphs mentioned above. For instance, Gavenčiak et al. [144, 142, 143] study the game of cops and robbers on intersection graphs. They prove that 15 cops are sufficient to catch the robber on every connected string graph, so this number is bounded on all connected intersection graphs of arc-connected sets in the plane. (Better bounds are proved for some of subclasses of string graphs; see [144, 142, 143] and the references therein.)

Interval Filament Graphs. In 2000, Gavril [147] introduced interval filament graphs as generalizations of interval graphs and circle graphs. An interval filament graph G is an intersection graph of graphs of the following continuous functions. For each $u \in \boldsymbol{V}(G)$, we have an interval $[a,b]$ and a function $f_u : [a,b] \to \mathbb{R}$ such that $f_u(a) = f_u(b) = 0$ and $f_u(x) > 0$ for $x \in (a,b)$. Then $\langle u \rangle = \Big\{(x, f(x)) : x \in [a,b]\Big\}$. We denote the class of interval filament graphs by IFA. We have that

$$\mathsf{CHOR} \cup \mathsf{FUN} \cup \mathsf{CIRCLE} \subsetneq \mathsf{IFA} \subsetneq \mathsf{STRING}.$$

Pergel [294] proved that recognition of interval filament graphs is NP-complete.

1.4 Planar Embeddings

An *embedding* $\mathcal{R}$ of a graph G is defined as follows. To each vertex $v \in \boldsymbol{V}(G)$, it assigns one point $\langle v \rangle$ in the plane $\mathbb{R}^2$. To each edge $uv \in \boldsymbol{E}(G)$, it assigns a curve

$\langle uv\rangle$ in the plane starting at $\langle u\rangle$ and ending at $\langle v\rangle$. Figure 1.1 gives three examples of embeddings. An embedding is called *planar* if the curves representing edges do not cross, i.e., they have pairwise disjoint interiors. We denote the class of all planar embeddings by CURVES. A graph is called *planar* if it has a planar embedding. The embedding in Fig. 1.1 on the right is planar, but the remaining two embeddings are not. *Faces* of the embedding are the regions of $\mathbb{R}^2 \setminus \mathcal{R}$. Combinatorially, a planar embedding is described by circular orderings of edges around each vertex. Another combinatorial description is by spherical maps; see Section 8.1 for more details. We denote the class of planar graphs by PLANAR.

The class of planar graphs is the oldest and most studied in graph theory, see the books [98, 282] for more details. The four color problem, asking whether each planar graph can be colored by four colors, was first asked in 1852 and greatly influenced graph theory. Around 1980, the first computer-assisted proof was announced by Appel and Haken [7], and a different proof was described in [306]. The famous results of Kuratowski [245] and Wagner [351] characterize planar graphs as those graphs which do not contain K_5 and $K_{3,3}$ as subdivisions or as minors. Several linear-time algorithms for recognition of planar graphs are known [193, 112, 39, 43, 92], Further, the algorithm of [43] is certifying, finding a subdivision of K_5 or $K_{3,3}$ in non-planar inputs. In this thesis, we are concerned with structural and algebraic properties of planar graphs.

Riemann showed that $\mathbb{C}$ extended by ∞ can be mapped by a conformal bijection into the sphere called *stereographic projection* [287, Section 3.IV]. Therefore, every planar embedding of a graph can be mapped into a spherical embedding (without crossing of edges) and vice versa; see Fig. 1.26. So planar graphs are exactly graphs which can be drawn onto the sphere without crossing edges.

A straight-line embedding is a planar embedding such that each edge is represented by a segment. By STRAIGHTLINE, we denote the class of straight-line embeddings. The well-known Fáry Theorem [114] states that every planar graph has a straight-line embedding: STRAIGHTLINE(G) is non-empty for each planar graph G.

1.4.1 Polyhedral Graphs

In 1922, Steinitz [329] proved that (vertex) 3-connected planar graphs correspond to polyhedrons. This means that that for every polyhedron P, there exists a planar embedding of a 3-connected planar graph G such that $\boldsymbol{V}(G)$ correspond to the vertices of P, $\boldsymbol{E}(G)$ correspond to the edges of P, and the faces of the planar embeddings

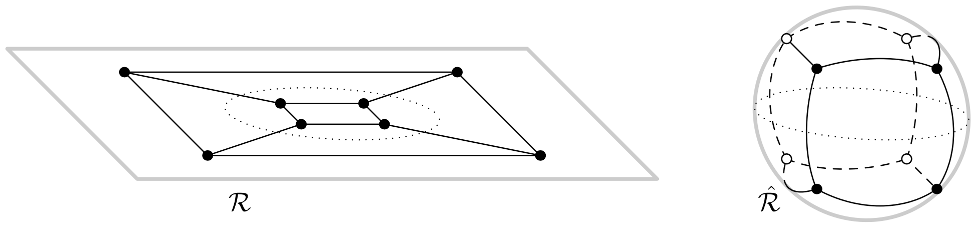


Figure 1.26: A planar graph G with a planar embedding $\mathcal{R}$ on the left and the corresponding spherical embedding $\hat{\mathcal{R}}$ on the right. We depict the unit circle and its image on the sphere by dots.

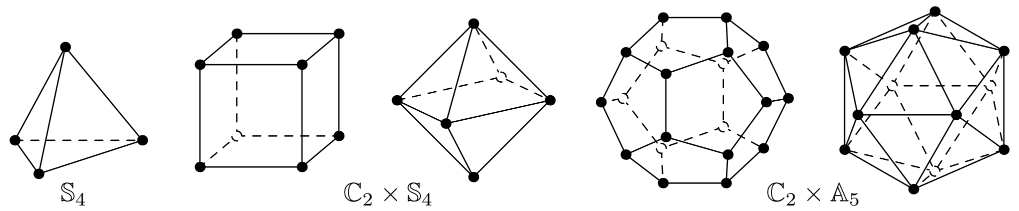

$\mathbb{S}_4$
$\mathbb{C}_2 \times \mathbb{S}_4$
$\mathbb{C}_2 \times \mathbb{A}_5$

Figure 1.27: The five platonic solids which capture the automorphism groups of the corresponding planar graphs by their groups of isometries.

correspond to the faces of P. And vice versa, every 3-connected planar graph G can be represented by some polyhedron. Whitney [359] further proved that 3-connected planar graphs have unique combinatorial embeddings in the sphere, where uniqueness is up to reversing the circular order of the edges around all vertices.

Mani [268] strengthened this result by describing the geometry of the automorphism group $\mathrm{Aut}(G)$ of a 3-connected planar graph G. He showed that there exists a polyhedron P such that $\mathrm{Aut}(G)$ coincides with the group of isometries of P. Figure 1.27 gives examples of such polyhedra associated to the graphs of platonic solids. Also, the polyhedron P can be placed inside the sphere and projected onto it, so that each isometry of P corresponds to some isometry of the sphere. Therefore, every automorphism in $\mathrm{Aut}(G)$ can be geometrically viewed as an isometry of the sphere with G drawn onto it. See [44] for further generalizations. We describe this in more detail in Section 8.1.

1.4.2 Contact Representations

A contact representation $\mathcal{R}$ is a special type of intersection representation, in which vertices are represented by geometric sets $\langle u \rangle$ with pairwise disjoint interiors, so $uv \in \boldsymbol{E}(G)$ if and only if $\partial \langle u \rangle \cap \partial \langle v \rangle \neq \emptyset$. We note that when each $\langle u \rangle$ is a segment, these segments are further not allowed to cross.

Notice that if each $\langle u \rangle$ is an arc-connected set in the plane, then only planar graphs can have geometric representations. On the other hand, not all planar graphs may be representable. As discussed in Section 1.1.3, by Koebe, Andreev and Thurston [232, 339], every planar graph has a contact representation by disks in the plane. Further, every planar graph has a contact representation by touching isosceles triangles with horizontal bases [91]. Every bipartite planar graph has a so-called *grid intersection representations* [177, 90], which are contact representations of segments such that the parts are represented by horizontal and vertical segments, respectively. Figure 1.28 shows examples.

1.4.3 Subclasses of Planar Graphs

Certainly the most famous subclass of planar graphs is the class of trees, denoted TREE. We also mention the class of caterpillar graphs, denoted CATERPILLAR, containing each tree which consists of a path with leaves arbitrarily attached. We note that CATERPILLAR = INT ∩ TREE. A pseudoforest is a graph such that each con-

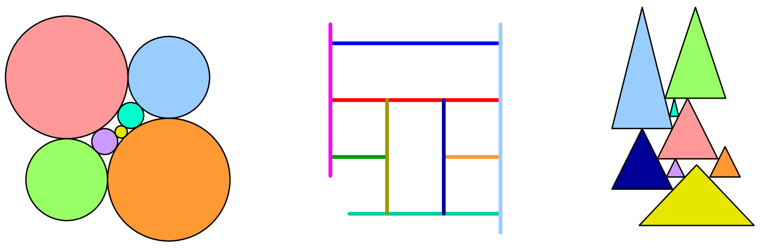


Figure 1.28: Three examples of contact representations of planar graphs.

nected component contains at most one cycle. We denote the class of pseudoforests by PSEUDOFOREST.

The class of outerplanar graphs, denoted OUTERPLANAR, consists of all planar graphs which have a planar embedding having all vertices in one face. We have TREE $\subsetneq$ OUTERPLANAR $\subsetneq$ CIRCLE.

Series-parallel graphs are defined inductively as follows. Each series-parallel graph contains a pair of terminal vertices (s,t). The graph K_2 with the vertices (s,t) is series-parallel. For two series-parallel graphs $G_1(s_1,t_1)$ and $G_2(s_2,t_2)$, we can construct a series-parallel graph on the vertices $\boldsymbol{V}(G_1) \cup \boldsymbol{V}(G_2)$ by using two operations. The *parallel operation* identifies $s_1 = s_2$ and $t_1 = t_2$ and has the terminal vertices (s_1,t_1). The *series operation* identifies $t_1 = s_2$ and has the terminal vertices (s_1,t_2). The class of all *generalized series-parallel* graphs (or just series-parallel graphs) consists of all graphs having each 2-connected block a series-parallel graph, and we denote this class by SERIES-PARALLEL. Clearly, OUTERPLANAR $\subsetneq$ SERIES-PARALLEL.

1.4.4 Bounded Genus Graphs and Other Graph Classes

For the purpose of Chapter 9, we quickly sketch possible generalizations of planar graphs; for more information, see e.g. [282, 98]. Instead of embeddings in the plane, we may consider embeddings in topological surfaces of higher genus, for instance projective planar and toroidal graphs. Further generalizations are graph classes with forbidden minors [307] and with forbidden topological subgraphs [163].

1.5 Results of This Thesis

The title of this thesis is "Extension Properties of Graphs and Structures". What is meant by extension properties? Suppose that we have partial information about some graph property. In this thesis, we study the question whether this partial information can be used to deduce the full information about the property. Such questions may be studied from the computational point of view or from the structural one. We concentrate on geometrically represented graphs and properties related to them.

In this section, we give an overview of the results described in this thesis. It is structured in two parts, which are mostly independent of each other. (The only major overlap is in Section 9.6.) Part I deals with the question of which partial intersection representations can be extended to full representations, and we are mostly concerned with interval graphs. Part II deals with algebraic properties of graphs: symmetries,

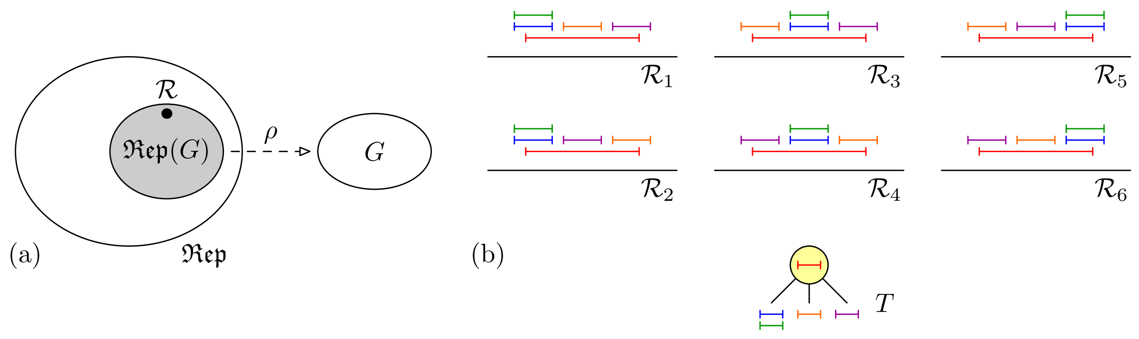


Figure 1.29: (a) We study what information about a graph G can be derived from the structure of all its geometric representations $\mathfrak{Rep}(G)$. (b) All six different interval representations of a given interval graph, which are efficiently described by the PQ-tree T at the bottom.

graph isomorphism and regular graph covers, and it is mostly concerned with planar graphs. For more detailed overviews, see Chapters 2 and 6, respectively.

Structure of All Representations. One of the reasons to study geometric representations of graphs is that the existence of such representations can be used to deduce additional information about graphs. For instance, to compute a minimum vertex coloring of an interval graph G, we first find an interval representation $\mathcal{R}$. Then we color the intervals greedily from left to right, always using the smallest available color. It can be easily proved that such a coloring uses the minimum number of colors.

Figure 1.29 shows the main underlying theme of this thesis. Let $\mathfrak{Rep}$ be a class of geometric representations. Suppose that instead of a representation $\mathcal{R}$ of G, we consider the entire set $\mathfrak{Rep}(G)$ of representations of G. For geometrically represented graph classes which are nicely structured, $\mathfrak{Rep}(G)$ is well understood and can be obtained. For instance, for interval graphs, Booth and Lueker [39] described a tree data structure called PQ-trees which efficiently captures all interval representations of a graph G (see Sections 3.1 and 4.2). The natural question is what additional information can be found about G using $\mathfrak{Rep}(G)$. We show that for the studied problem, a full understanding of $\mathfrak{Rep}(G)$ is very helpful.

1.5.1 Part I: Partial Representation Extension Problems

The partial representation extension problems generalize the recognition problems for geometrically represented graphs. Aside from the graph, the input also gives a partial representation, which prescribes a representation of a part of the graph. We ask whether this partial representation can be extended to a full representation of the input graph without altering the representation of the prescribed parts. For intersection representations, a partial representation is an intersection representation of an induced subgraph, and the sets representing these vertices cannot be altered.

In the context of intersection representations, I introduced this problem in my Bachelor's thesis from 2010 and showed that the problems can be solved in polynomial time for interval graphs, proper interval graphs and permutation graphs.[3] Since then,

[3]We note the partial embedding extension problems for planar graphs [292, 4] were already studied

research on the partial representation extension problems has been very active and many computational and structural results for a variety of intersection graph classes have been proved. In Part I, we describe this development and mostly concentrate on the partial representation extension problem for interval graphs.

Chapter 2. We precisely define the partial representation extension problems and give an overview of the state-of-the-art results. We describe the main ideas and techniques, and also discuss related restricted representation problems.

Chapter 3. We introduce the classical characterization of interval graphs by Fulkerson and Gross [133] in terms of consecutive orderings of maximal cliques, and describe PQ-trees of Booth and Lueker [39], which efficiently store all these consecutive orderings.

We show that a partial interval representation inposes a partial ordering $\lhd$ on the order of maximal cliques. The main structural characterization states that a partial interval representation is extendible if and only if there exists a consecutive ordering of maximal cliques extending $\lhd$. Further, we show that this property can be tested in linear time by reordering PQ-trees, which leads to a linear-time algorithm for partial representation extension of interval graphs. Section 2.2 contains a more detailed overview.

Chapter 4. We generalize the minimal forbidden induced subgraphs of interval graphs to partially represented interval graphs. We study minimal obstructions which are minimal graphs with non-extendible partial representations. If a minimal obstruction is contained in a partially represented interval graph, then the partial representation is clearly non-extendible.

We show that minimal obstructions are involved, consisting of 10 infinite classes, but nevertheless we fully describe their structure. The proof uses the characterization of extendible partial interval representations of Chapter 3 and modified PQ-trees (MPQ-trees) of Korte and Möhring [235]. If a partial representation is non-extendible, some node of the MPQ-tree cannot be reordered, and we divide the argument according to the type of this node. We find a small configuration of maximal cliques obstructing the reordering, derive positions of pre-drawn intervals and locate a minimal obstruction.

As a consequence of our characterization, we prove that partial interval representations satisfy the Helly property: a partial representation is extendible if and only if every quadruple of pre-drawn intervals is extendible by themselves, while ignoring the positions of the remaining pre-drawn intervals. Also, we show that the linear time algorithm of Chapter 3 can be modified to certify non-extendible partial representations by finding a minimal obstruction. See Section 2.2 for a detailed overview.

Chapter 5. We study interval graphs of limited nesting and count of lengths. We show that the partial representation extension problem is NP-hard already for 2-$\mathsf{LengthINT}$. On the other hand, we illustrate the nice structure of k-$\mathsf{NestedINT}$ by describing a relatively simple linear-time recognition algorithm by dynamic programming on MPQ-trees. See Section 2.4 for precise statements of our results.

in 2007. See Section 2.8 for more detail.

1.5.2 Part II: Extending Algebraic Properties of Graphs

We study algebraic properties of graphs, namely their automorphism groups, the graph isomorphism problem and regular graph covering. Automorphism groups describes symmetries of graphs. The graph isomorphism problem (denoted GraphIso) asks whether two graphs are the same up to relabeling the vertices. Its computational complexity is a famous unresolved problem in theoretical computer science and it is closely related to the complexity of computing automorphism groups; see Section 6.4. Regular graph covering is a similarity relation between two graphs which originated in topology, and it implies that many graph properties are shared.

The main structural tool is the 3-connected reduction which decomposes each graph G into *3-connected components* (essentially 3-connected graphs, cycles, and dipoles). We are mostly concerned with planar graphs since 3-connected planar graphs are very restricted. For instance, as described in Section 1.4, the automorphism groups of 3-connected planar graphs are spherical groups. The main question is how to combine these spherical groups of all 3-connected components to construct the automorphism group $\mathrm{Aut}(G)$ of the entire planar graph G. So the main question studied in this part is whether an understanding of algebraic properties for 3-connected components can be extended into an understanding of these properties for the entire planar graph. We note that some of the presented results also apply to general graphs.

Chapter 6. We give a more detailed overview of Part II. We precisely define the studied algeabraic properties, state the main result proved in this thesis and discuss other related results.

Chapter 7. Seminal papers by Mac Lane [265] and Trakhtenbrot [340] introduced the idea that every graph G can be decomposed into 3-connected components. It was further extended in [344, 188, 191, 78, 352, 27]. This decomposition can be represented by a *reduction tree* whose nodes are 3-connected graphs, and this tree is known in the literature mostly under the name SPQR tree [95, 96, 97, 167].

The reduction procedure works by finding certain inclusion minimal subgraphs called *atoms* and by replacing them with edges. We introduce augmented reduction trees which use colored and possibly directed edges, to capture isomorphism classes and symmetry types of atoms. This allows to track changes in the automorphism group, since the reduction defines a natural group epimorphism. We show that the augmented reduction tree captures all automorphisms of a graph. See Section 6.2 for a more detailed overview.

Chapter 8. In 1869, Jordan [202] inductively characterized the automorphism groups of trees as precisely the class of groups closed under the direct product and the wreath product with symmetric groups. We prove a similar Jordan-like inductive characterization of the automorphism groups of planar graphs. The basis is our 3-connected decomposition and the well-known characterization of automorphism groups of 3-connected planar graphs. See Section 6.3 for more details. By applying our results, we also characterize the automorphism groups of 2-connected planar graphs, outerplanar graphs and series parallel graphs.

Chapter 9. Let G and H be graphs. Suppose that for each $u \in \boldsymbol{V}(G)$, we are given a list $\mathfrak{L}(u) \subseteq \boldsymbol{V}(H)$ of possible images of u. The *list restricted graph isomorphism problem* (denoted ListIso) asks whether there exists an isomorphism $\pi : G \to H$ such that $\pi(u) \in \mathfrak{L}(u)$. In 1981, Lubiw [260] proved that ListIso is NP-complete. After 35 years, we revive the study of this problem and consider which results for GraphIso translate to ListIso.

We prove the following:

- When GraphIso is GI-complete for a class of graphs, it usually translates into NP-completeness of ListIso.
- Combinatorial algorithms for GraphIso translate into algorithms for ListIso: for trees, planar graphs, interval graphs, circle graphs, permutation graphs, bounded genus graphs, and bounded treewidth graphs.
- Two basic algorithms based on group theory do not translate: ListIso remains NP-complete for cubic colored graphs with sizes of color classes bounded by 8.

Also, ListIso allows to classify results for the graph isomorphism problem. Some algorithms are robust and translate to ListIso. A fundamental problem is to construct a combinatorial polynomial-time algorithm for cubic graph isomorphism, avoiding group theory. By the last result, ListIso is NP-hard for these graphs, so no robust algorithm for cubic graph isomorphism exists, unless P = NP.

Chapter 10. If G regularly covers H, then H is called a (regular) quotient of G. In 1988, Negami [288] characterized the quotients of planar graphs exactly as projective planar graphs. Inspired by his approach, we give more insight into the behaviour of regular graph covering with respect to the 3-connected decomposition, namely with respect to 1-cuts (easy) and 2-cuts (very complex). For a graph G, we structurally describe all its quotients by composing them from the quotients of the 3-connected components in the decomposition. Further, we apply our results to planar graphs, give a direct proof of the Negami Theorem [288] and describe more insight into the geometry of quotients of planar graphs.

Chapter 11. We use the structural results of Chapter 10 to construct the following algorithm. For an input planar graph G and an arbitrary graph H, it tests the existence of regular covering in FPT time when parameterized with respect to the size of H. Unlike most FPT algorithms in the area, our algorithm is quite involved. Even with the full description of all quotients of G, it is not easy to test whether H is one of them, and the existence of a polynomial-time algorithm remains an open problem. As special cases, when G is 3-connected, H is 2-connected, or the ratio of sizes of $\boldsymbol{v}(G)$ and $\boldsymbol{v}(H)$ is an odd integer, we can solve the problem in polynomial time.

PART

I

The Partial Representation Extension Problems

2 State of The Art for Partial Representation Extension

This chapter contains:

- *2.2 and 2.4.* An overview of results proved in Chapters 3, 4, and 5.
- *2.3, 2.5, 2.6 and 2.7.* We survey results for partial representation extension of proper interval, unit interval, chordal, circle, comparability, permutation, function, trapezoid, and proper-circular arc graphs.
- *2.8 Extending Other Types of Partial Representations.* We survey results for partial embedding extension, partial representation extension of contact representations of planar graphs, and partial visibility extension.
- *2.9 Related Restricted Representation Problems.* We describe chronological orderings, bounded representations, representation sandwich, simultaneous representations, and Allen algebras from time reasoning.
- *2.10 Open Problems.* We conclude with a list of open problems.

http://pavel.klavik.cz/orgpad/repext.html

2.1 Definitions and Motivation

Let $\mathcal{C}$ be a class of graphs and $\mathfrak{Rep}$ be a class of intersection representations. Suppose that G' is an induced subgraph of a graph $G \in \mathcal{C}$. A *partial representation* $\mathcal{R}'$ of G is an intersection representation $\big\{\langle x\rangle' : x \in V(G')\big\} \in \mathfrak{Rep}(G')$. The vertices of G' and the sets of $\mathcal{R}'$ are called *pre-drawn*. A representation $\mathcal{R}$ *extends* $\mathcal{R}'$ if and only if it assigns the same sets to the vertices of G', i.e., $\langle x\rangle = \langle x\rangle'$ for every $x \in V(G')$.

Partial Representation Extension. In Part I, we study the following decision problem generalizing the recognition problem:

Problem:	Partial representation extension – $\textsc{RepExt}(\mathcal{C}, \mathfrak{Rep})$
Input:	A graph $G \in \mathcal{C}$ and a partial representation $\mathcal{R}' \in \mathfrak{Rep}$.
Question:	Is there a representation $\mathcal{R} \in \mathfrak{Rep}$ of G extending $\mathcal{R}'$?

When the class of geometric representations $\mathfrak{Rep}$ is clear from the context, we may denote the problem just by $\textsc{RepExt}(\mathcal{C})$. Figure 2.1 compares the recognition problems and the partial representation extension problems. Figure 2.2 gives an overview the state-of-art complexity results for the partial representation extension problems.

Let $\mathrm{Ext}(\mathcal{R}')$ denote the class of representations in $\mathfrak{Rep}$ extending $\mathcal{R}'$, i.e., $\mathcal{R} \in \mathrm{Ext}(\mathcal{R}')$ if and only if $\mathcal{R} \in \mathfrak{Rep}$ extends $\mathcal{R}'$, and let $\mathrm{Ext}(\mathcal{R}', G) = \mathrm{Ext}(\mathcal{R}') \cap \mathfrak{Rep}(G)$. When the class $\mathfrak{Rep}$ is not clear from the context, we may also write $\mathrm{Ext}_{\mathfrak{Rep}}(\mathcal{R}')$ and $\mathrm{Ext}_{\mathfrak{Rep}}(\mathcal{R}', G)$, respectively. For $A \subseteq \boldsymbol{V}(G')$, we denote by $\mathcal{R}'[A]$ the restriction of the partial representation $\mathcal{R}'$ to only pre-drawn sets in A.

Motivation. To solve the recognition problem, an arbitrary representation can be constructed. Solving partial representation extension is harder, and better understanding of the structure of all possible representations is very helpful. This added difficulty is the well-known phenomenon in architecture in a saying that reconstructing an existing house is much harder than building a new house from scratch. In mathematics, this added difficulty is a desirable property since one is forced to improve the structural understanding of the studied classes to solve this problem; and this structural understanding can be later applied in attacking other problems.

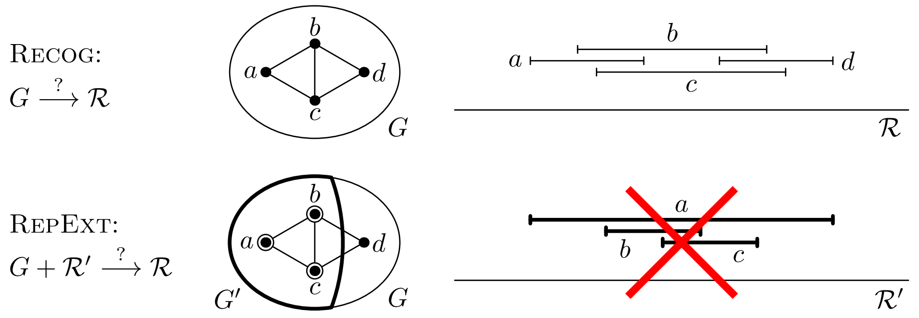


Figure 2.1: The graph G is an interval graph, but the partial representation $\mathcal{R}'$ is not extendible. In all figures, we depict pre-drawn vertices with circles and pre-drawn sets in bold.

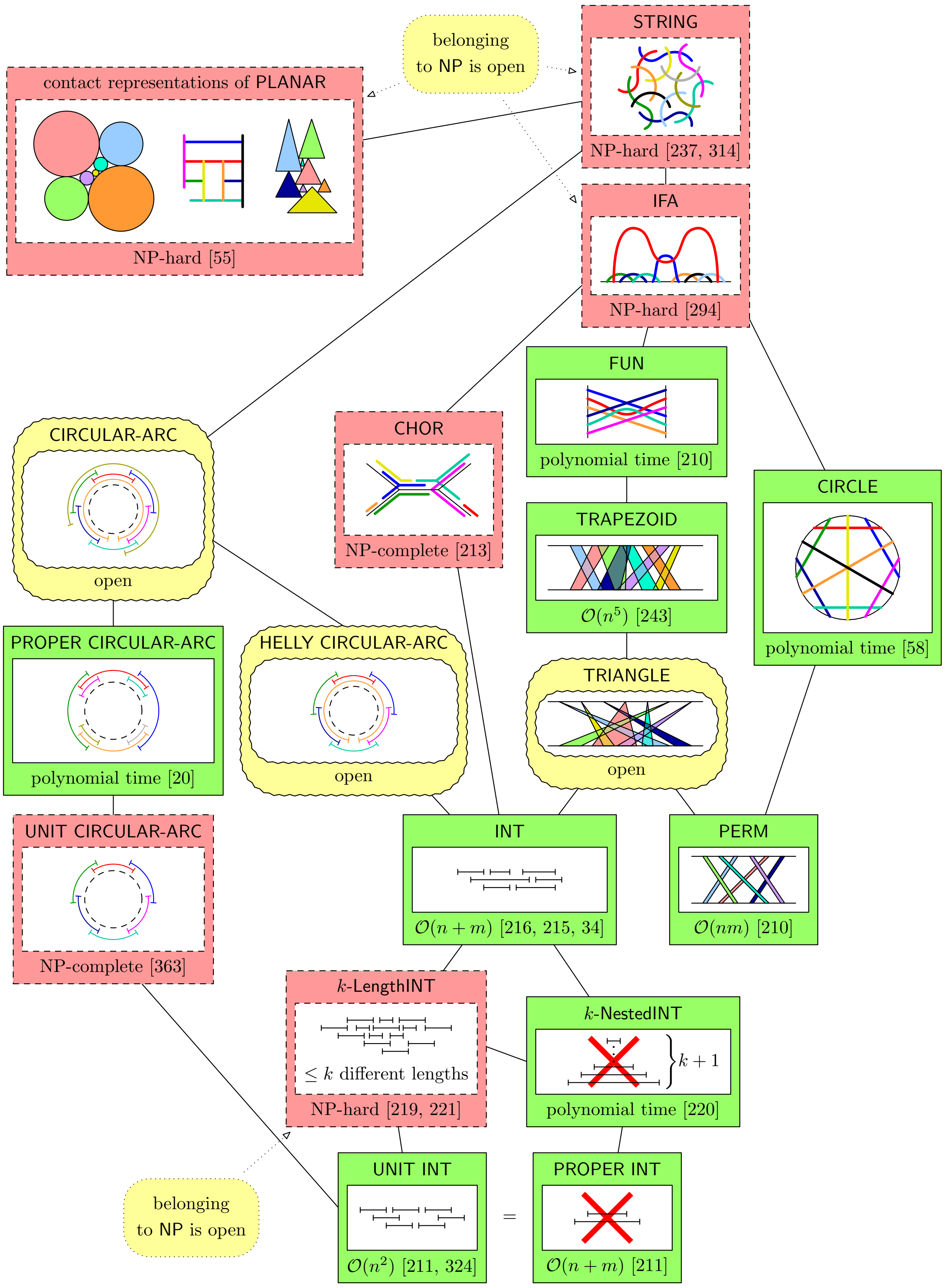
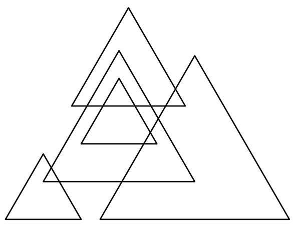
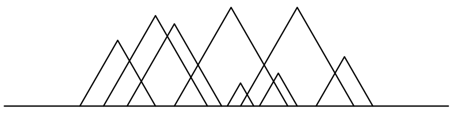


Figure 2.2: Hasse diagram of inclusions of intersection graph classes, together with the known complexity results for the partial representation extension problems. Notice that the complexity does not depend on inclusions of graph classes. For instance, if $\mathcal{C}'$ is a subclass of $\mathcal{C}$ and the problem RepExt($\mathcal{C}, \mathfrak{Rep}$) can be solved in polynomial time, it does not in general imply that the problem RepExt($\mathcal{C}', \mathfrak{Rep}'$) can be solved in polynomial time.

(a) NP-hard [239] (b) linear-time solvable [39, 235, 168, 75]

Figure 2.3: (a) Recognizing intersection graphs of homothetic equilateral triangles is NP-hard. (b) With all bases on one line, we get interval graphs which can be recognized in linear time.

In Chapter 3, 4, and 5, we give a lot of insight into the structure of all representations of interval graphs. The papers [211, 212, 58, 59], described in Sections 2.3 and 2.6 build completely new structural results for unit interval and circle graphs which might be of independent interest. The complexity of the partial representation extension problem for circular-arc graphs is the main open problem; and if it is solved in polynomial time, it will likely lead to new structural results describing all possible representations.

Partial representation extension belongs to a larger group of restricted representation problems, see Section 2.9 for more detail.

Homothetic Equilateral Triangles on Two Lines. We conclude this section with an initial motivation which lead us to introduce the partial representation extension problems in [216]. Kratochvíl and Pergel [239] studied intersection graphs of homothetic equilateral triangles and proved that their recognition is NP-hard; see Fig. 2.3a. On the other hand, when bases of all triangles belong to one line, we can ignore the rest of triangles and we get the well-known interval graphs which can be recognized in linear time [39]; see Fig. 2.3b. We tried to solve the following (still open) problem of Pultr: what is the complexity of recognizing when bases of all triangles belong to two parallel lines; see Fig. 2.4.

Suppose that two lines are horizontal and all equilateral triangles point upwards. Each intersection representation $\mathcal{R}$ partitions $\boldsymbol{V}(G)$ into two subsets T and B such that the vertices in T are represented by triangles on the top line and the vertices in B are represented by triangles on the bottom one. So $\mathcal{R}$ consists of two interval

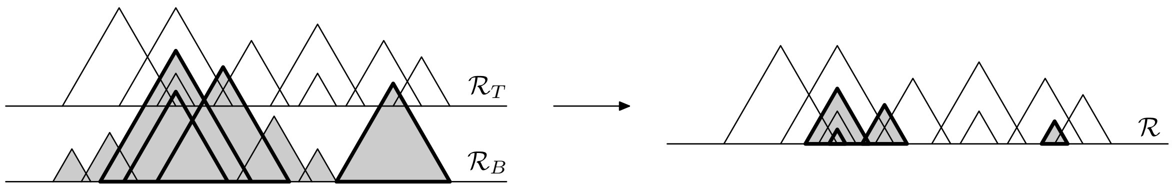


Figure 2.4: On the left, a representation using two lines. The triangles of $\mathcal{R}_T$ are depicted in white, the triangles of $\mathcal{R}_B$ in gray, and those in B' are depicted in bold. On the right, the corresponding interval representation $\mathcal{R}$ extending the partial representation $\mathcal{R}'_{B'}$.

representations $\mathcal{R}_B = \mathcal{R}[B]$ and $\mathcal{R}_T = \mathcal{R}[T]$ together with some intersections in between. More precisely, if $uv \in \boldsymbol{E}(G)$, $u \in B$ and $v \in T$, then $\langle u \rangle$ is high enough to intersect $\langle v \rangle$ on the top line.

To simplify the problem, suppose that we would know the partition T and B. To simplify even further, let a representation $\mathcal{R}_B$ of $G[B]$ be known. Can we complement it by a representation $\mathcal{R}_T$ of $G[T]$? Let

$$B' = \{u \in B : \langle u \rangle \text{ intersects the top line}\}.$$

Positions of triangles corresponding to the vertices in $B \setminus B'$ can be ignored in construction of $\mathcal{R}_T$. On the other hand, the triangles of B' restrict $\mathcal{R}_T$. For each $u \in B'$, we can cut the triangle $\langle u \rangle$ by the top line and obtain a fixed triangle $\langle u \rangle'$ based at the top line. We get an interval graph $G[T \cup B']$ with a partial representation $\mathcal{R}'_{B'}$; see Fig. 2.4. The complementing representation $\mathcal{R}_T$ can be constructed if and only if the partial representation $\mathcal{R}'_{B'}$ is extendible.

2.2 Interval Graphs

In 2011, we have initiated the study of partial representation extension problems by the class of interval graphs [216]. There have been two natural motivations to start with interval graphs. As mentioned in Section 1.3.1, this class is the oldest and most understood class of intersection graphs. Also, there are many structural results and techniques known for interval graphs. Namely, we can use PQ-trees [39] (described in Section 3.1) which combinatorially determine all interval representations of an interval graph. This way we have discovered the most important properties of partial representation extension, applicable to other more complex graph classes, without dealing with technical details.

Therefore, the problem RepExt(INT) is well-understood from the structural and algorithmic point of view. In particular, we show that a good understanding of the structure of all representations is essential to solve the problem; unlike recognition for which an arbitrary representation has to be found. We present the results of [216, 215, 222, 223] in Chapters 3 and 4.

2.2.1 Structural Results

There are two essential structural papers for interval graphs: the characterization of Fulkerson and Gross [133] and the list of minimal forbidden induced subgraphs by Lekkerkerker and Boland [249]. We generalize both of these results to partially represented interval graphs. We note that we allow the intervals to share the endpoints and to have zero lengths.

Generalizing Fulkerson and Gross. Fulkerson and Gross [133] proved in 1965 the following characterization. A graph is an interval graph if and only if there exists an ordering $<$ of its maximal cliques such that for each vertex the cliques containing this vertex appear consecutively in $<$. Intervals of the real line have the Helly property, so all intervals representing one maximal clique have a common intersection. In

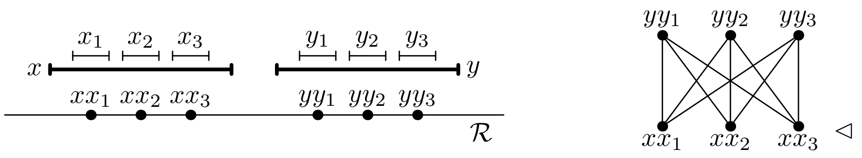


Figure 2.5: An interval graph consisting of two stars with pre-drawn central vertices. One of the extending representations is on the left. Any extending representation places all maximal cliques containing x on the left of all maximal cliques containing y. Thus, the ordering of the maximal cliques has to extend the partial ordering $\lhd$, given by the Hasse diagram on the right.

this intersection, we choose one point which we call a clique-point. The ordering is the left-to-right ordering of the chosen clique-points; see Fig. 2.5. We describe this characterization in detail in Section 3.1.

In Section 3.2, we generalize this result and characterize extendible partial representations. A partial representation gives a partial ordering $\lhd$ which has to be extended by the ordering $<$ of the maximal cliques of any extending representation; see Figure 2.5 for an example. In Theorem 3.2.2, we prove that the constraints posed by $\lhd$ are not only necessary, but also sufficient.

Generalizing Lekkerkerker and Boland. Recall that chordal graphs are graphs with no induced cycle of length four or more. Alternatively, they are intersection graphs of subtrees of a tree. Three vertices form an *asteroidal triple* if there exists a path between every pair of them avoiding the neighborhood of the third vertex. *Asteroidal triple-free graphs* (AT-FREE) are graphs containing no asteroidal triples. Lekkerkerker and Boland [249] characterized interval graphs as INT = CHOR ∩ AT-FREE. They described this characterization by the minimal forbidden induced subgraphs given in Fig. 2.6 which we call *Lekkerkerker-Boland obstructions* (LB).

We generalize the characterization of Lekkerkerker and Boland [249] to describe minimal obstruction which make partial representations non-extendible. Each obstruction consists of a small graph and its non-extendible partial representation. Aside LB obstructions, we have two trivial obstructions, called SE, and ten infinite classes of minimal obstructions. The main class, called k-FAT obstructions, has three wrongly ordered disjoint pre-drawn intervals $\langle x_k\rangle'$, $\langle y_k\rangle'$, and $\langle z_k\rangle'$. The obstruction consists of a zig-zag structure with k levels where the last level cannot be placed. See Fig. 2.7a and

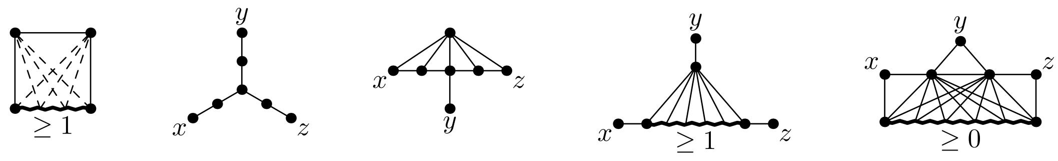


Figure 2.6: Five types of LB obstructions which are the minimal forbidden induced subgraphs of INT. The bold curly lines correspond to induced paths with denoted minimal lengths. The leftmost obstructions are induced cycles of length four or more. The remaining four types of obstructions are minimal asteroidal triples (x, y, z) which are chordal graphs. In all figures, we always depict all edges with normal lines and some highlighted non-edges with dashed lines.

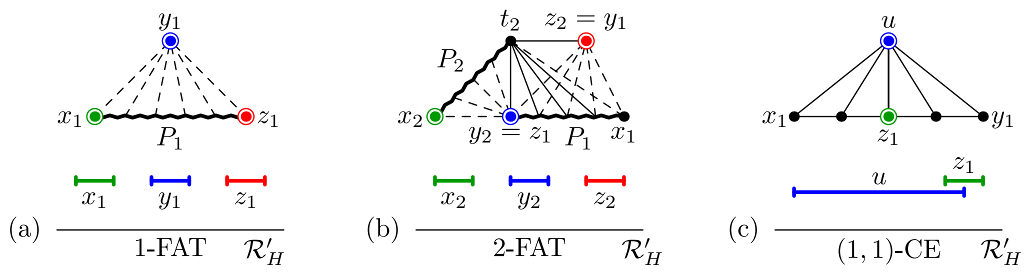


Figure 2.7: Three examples of minimal obstructions, each consisting of a graph H and a non-extendible partial representation $\mathcal{R}'_H$. Curly lines denote induced paths and dashed edges are non-edges. The obstructions (a) and (b) are the first two k-FAT obstructions, and (c) is the simplest (k,ℓ)-CE obstruction.

b for 1-FAT and 2-FAT obstructions. There are eight other infinite classes derived from k-FAT obstructions by adding a few vertices and having different vertices pre-drawn. The last infinite class of (k,ℓ)-CE obstructions consists of a k-FAT obstruction glued with an ℓ-FAT obstruction and contains only two pre-drawn vertices; see Fig. 2.7c for a $(1,1)$-CE obstruction. We formally define these minimal obstructions in Section 4.1.

Theorem 2.2.1. *A partial representation $\mathcal{R}'$ of G is extendible if and only if G and $\mathcal{R}'$ contain no LB, SE, k-FAT, k-BI, k-FS, k-EFS, k-FB, k-EFB, k-FDS, k-EFDS, k-FNS and (k,ℓ)-CE obstructions.*

Every minimal obstruction contains at most four pre-drawn intervals. Thus, the following surprising Helly-type result saying that a partial interval representation is extendible if and only if every quadruple of pre-drawn intervals is extendible by itself follows as a straightforward corollary:

Corollary 2.2.2. *A partial interval representation $\mathcal{R}'$ is extendible if and only if for every $A \subseteq \boldsymbol{V}(G')$, $|A| \le 4$, the restricted partial representation $\mathcal{R}'[A]$ is extendible.*

2.2.2 Algorithmic Results

Our structural results translate into a linear-time certifying algorithm for RepExt(INT). All other algorithms for the partial representation extension problems [58, 210, 34, 211, 212, 213, 214] are able to certify solvable instances by outputting an extending representation. Using the minimal obstructions, we construct the first algorithm for partial representation extension certifying also non-extendible partial representations.[1]

Theorem 2.2.3. *Assume that the input gives the endpoints in a partial representation $\mathcal{R}'$ sorted from left to right. Then there exists an $\mathcal{O}(n+m)$ certifying algorithm for the partial representation extension problem, where n is the number of vertices and m is the number of edges of the input graph. If the answer is "yes", it outputs an extending representation. If the answer is "no", it detects one of the minimal obstructions.*

[1]Formally speaking, a polynomial-time algorithm certifies unsolvable instances by outputting "no" and by a proof of its correctness. Our algorithm outputs a simple proof that a given partial representation is non-extendible in terms of a minimal obstruction. This proof can be independently verified which is desirable.

To test whether a partial representation $\mathcal{R}'$ is extendible, we use Theorem 3.2.2. We work with a PQ-tree (Section 3.1) which combinatorially describes all orderings of the maximal cliques yielding interval representations. We test whether this tree can be reordered according to $\lhd$. By applying several tricks, we can test this for a specific type of partial orderings called interval orders (defined in Section 1.3.1) in linear time. If the answer is "yes", by following the proof of Theorem 3.2.2, we find a representation $\mathcal{R}$ extending $\mathcal{R}'$. If the answer is "no", we follow the proof of Theorem 2.2.1 to construct one of the minimal obstructions.

To obtain the linear-time algorithm, we need some reasonable assumption on a partial representation which is given by the input. Similarly, most of the graph algorithms cannot achieve better running time than $\mathcal{O}(n^2)$ if the input graph is given by an adjacency matrix instead of a list of neighbors for each vertex. We say that a partial representation is *sorted* if it gives all (left and right) endpoints of the pre-drawn intervals sorted from left to right. We assume that the input partial representation is given sorted. If this assumption is not satisfied, the algorithm needs additional time $\mathcal{O}(k \log k)$ to sort the partial representation where k is the number of pre-drawn intervals.

Another approach. Bläsius and Rutter [34] describe an alternative linear-time algorithm for RepExt(INT), by solving a more general problem described in Section 2.9.4 using simultaneous PQ-trees. This framework applies to other problems such as simultaneous planar embeddings. Consequently their algorithm is quite involved and gives no understanding of partial representation extension. We note that Bläsius and Rutter [34] need the same assumption of a sorted partial representation for their linear-time algorithm.

2.3 Proper and Unit Interval Graphs

These two subclasses of interval graphs are described in Section 1.3.1. We give an overview of the results for the partial representation extension problems of [211, 212, 323, 324].

2.3.1 Proper Interval Graphs

We sketch the algorithm of [211, 212] showing the following result:

Theorem 2.3.1 (Klavík et al. [211, 212])**.** *The problem* RepExt(PROPER INT) *can be solved in time $\mathcal{O}(n+m)$, where n is the number of vertices and m is the number of edges.*

Two vertices u and v are called *twins* if $N[u] = N[v]$. This is an equivalence relation and it defines *twin classes.* In every proper interval representation $\mathcal{R}$ of G, the left-to-right ordering of left endpoints is the same as the left-to-right ordering of right ones. So we get a left-to-right ordering $<$ of $\boldsymbol{V}(G)$. Further, the vertices of each twin class appear consecutively in $<$. The orderings $<$ are characterized as follows.

Lemma 2.3.2 (Roberts [304], Deng et al. [94])**.** *For a connected proper interval graph, the ordering $<$ is uniquely determined up to local reordering of twin classes and complete reversal.*

A component is called *located* if it has at least one interval pre-drawn, and *unlocated* otherwise. The unlocated components can always be placed far from all pre-drawn intervals, so we can deal with them using the standard linear-time recognition algorithm [73].

The located components are ordered from left-to-right by an ordering $\blacktriangleleft$, otherwise the partial representation is not extendible. Using Lemma 2.3.2, we test for each located component whether the left-to-right order $<'$ of pre-drawn intervals is compatible with $<$ or its reversal (up to local reordering of twin classes). This can be decided in linear time, concluding our sketch of the proof of Theorem 2.3.1. (We note that touching pre-drawn intervals $\langle u\rangle'$ and $\langle v\rangle'$, having $r(u) = \ell(v)$, pose additional contraints which can be easily deal with; see [212, Lemma 2.3] for details.)

Another approach of Bang-Jensen et al. [20] solving RepExt(PROPER INT) in polynomial time using acyclic local tournaments [94] is described in Section 2.7.6.

2.3.2 Unit Interval Graphs

By [304], PROPER INT = UNIT INT. But the problem RepExt(UNIT INT) is harder to solve than RepExt(PROPER INT), as illustrated by an example in Fig. 2.8a.

Let PROPER INT $\mathfrak{Rep}$ be the class of proper interval representations and let UNIT INT $\mathfrak{Rep}$ be the class of unit interval ones. By definition, UNIT INT $\mathfrak{Rep} \subsetneq$ PROPER INT $\mathfrak{Rep}$. The relation is depicted in Fig. 2.8b. So while PROPER INT = UNIT INT, their partial representation extension problems are different:

$$\begin{aligned}\textsc{RepExt}(\textsf{PROPER INT}) &= \textsc{RepExt}(\textsf{PROPER INT} = \textsf{UNIT INT}, \textsf{PROPER INT}\ \mathfrak{Rep})\\ &\neq \textsc{RepExt}(\textsf{PROPER INT} = \textsf{UNIT INT}, \textsf{UNIT INT}\ \mathfrak{Rep}) = \textsc{RepExt}(\textsf{UNIT INT}).\end{aligned}$$

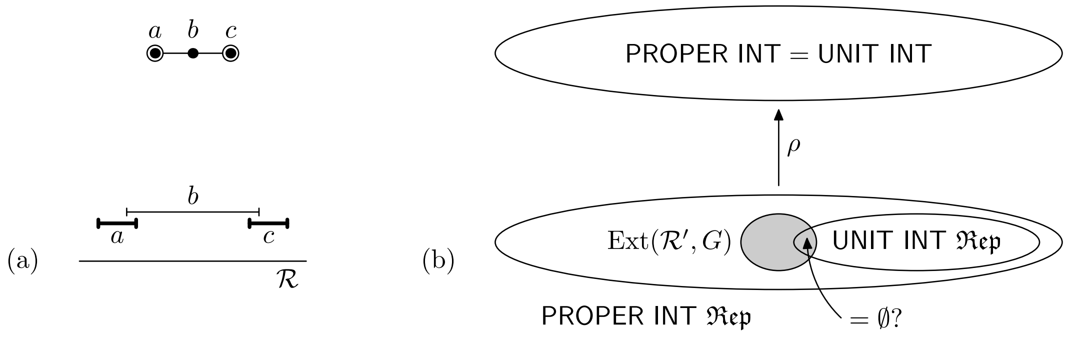


Figure 2.8: (a) A partial representation which is extendible by the depicted proper interval representation, but non-extendible by a unit interval representation.
(b) The three classes studied in [211, 212]: the class of proper/unit interval graphs PROPER INT = UNIT INT, the class of proper interval representations PROPER INT $\mathfrak{Rep}$ and its subclass of unit interval representations UNIT INT $\mathfrak{Rep}$. The denoted mapping ρ assigns to a representation the graph it represents. Roberts' Theorem [304] states that $\rho|_{\textsf{UNIT INT}\ \mathfrak{Rep}}$ is surjective.

Let $\mathrm{Ext}(\mathcal{R}', G)$ denote all proper interval representations of G extending $\mathcal{R}'$. The problem $\textsc{RepExt}(\mathsf{PROPER\ INT})$ is solvable if and only if $\mathrm{Ext}(\mathcal{R}', G) \neq \emptyset$, but the problem $\textsc{RepExt}(\mathsf{UNIT\ INT})$ is solvable if and only if

$$\mathrm{Ext}(\mathcal{R}', G) \cap \mathsf{UNIT\ INT}\ \mathfrak{Rep} \neq \emptyset,$$

In particular, the problem $\textsc{RepExt}(\mathsf{UNIT\ INT})$ is solvable only when the problem $\textsc{RepExt}(\mathsf{PROPER\ INT})$ is solvable.

Required Resolution. Some difference between the problems $\textsc{RepExt}(\mathsf{PROPER\ INT})$ and $\textsc{RepExt}(\mathsf{UNIT\ INT})$ is expected. Unit interval representations deal with precise positions of endpoints, but only the left-to-right ordering of endpoints matters for (proper) interval graphs; we call this difference as *geometry versus topology.*

For instance, recognition algorithms for unit interval graphs need extra work to certify positive answers by building unit interval representations. Also, the required resolution for their drawing needs to be considered. In [73], it is proved constructively that every n-vertex unit interval graph has a representation in the grid of resolution $\frac{1}{n}$: for each $\langle u \rangle$, we have $\ell(u) = \frac{k}{n}$ for some $k \in \mathbb{Z}$.

For $\textsc{RepExt}(\mathsf{UNIT\ INT})$, we want to show that an extending representation, if it exists, can be described with polynomially-large resolution with respect to the size of input. Clearly, the grid has to contain endpoints of all predrawn intervals. Let the endpoints of pre-drawn interval be expressed as irreducible fractions $\frac{p_1}{q_1}, \frac{p_2}{q_2}, \cdots, \frac{p_b}{q_b}$. Then we define:

$$\varepsilon' := \frac{1}{\mathrm{lcm}(q_1, q_2, \ldots, q_b)}, \qquad \text{and} \qquad \varepsilon := \frac{\varepsilon'}{n}, \tag{2.1}$$

where $\mathrm{lcm}(q_1, q_2, \ldots, q_b)$ denotes the *least common multiple* of $q_1, \ldots, q_b$.

Lemma 2.3.3 (Klavík et al. [211, 212])**.** *If there exists a unit interval representation $\mathcal{R}$ extending $\mathcal{R}'$, then there exists a unit interval representation $\hat{\mathcal{R}}$ in which all intervals have endpoints on the ε-grid, where ε is defined by (2.1).*

Sketch of Proof. See Figure 2.9 for an overview. We transform $\mathcal{R}$ into $\hat{\mathcal{R}}$ by shifting each interval twice. First, we apply a *left shift* $\mathrm{LS}(v)$ for each interval $\langle v \rangle$, then we apply a *right shift* $\mathrm{RS}(v)$ to $\langle v \rangle$.

We choose $\mathrm{LS}(v)$ equal to the distance of $\ell(v)$ from the closest ε'-grid point on the left, so $0 \leq \mathrm{LS}(v) < \varepsilon'$. Notice that original intersections are kept by left shifting,

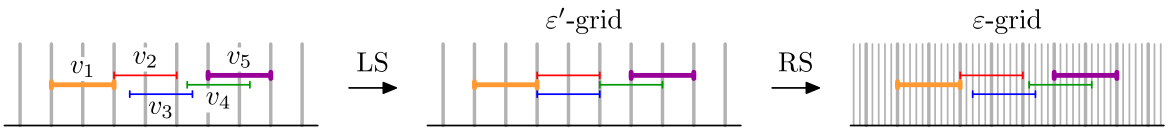


Figure 2.9: First, we shift intervals to the left to the ε'-grid. The left shifts of $v_1, \ldots, v_5$ are $(0, 0, \frac{1}{2}\varepsilon', \frac{1}{3}\varepsilon', 0)$. Then, we shift to the right in the refined ε-grid. Right shifts have the same relative order as left shifts: $(0, 0, 2\varepsilon, \varepsilon, 0)$.

but we might introduce new touching pairs of intervals. We remove them by suitable right shifting in the ε-grid

$$\mathrm{RS} : \boldsymbol{V}(G) \to \Big\{0, \varepsilon, 2\varepsilon, \dots, (n-1)\varepsilon\Big\}$$

having the *right-shift property*: for all $u, v \in \boldsymbol{V}(G)$ with $r(u) = \ell(v)$, $\mathrm{RS}(u) \geq \mathrm{RS}(v)$ if and only if $v_iv_j \in \boldsymbol{E}(G)$. So the right-shift property ensures that RS fixes wrongly introduced touching pairs created by LS while keeping correct intersections.

Construction of such a mapping RS in the key trick in the proof. Notice that if we relax the image of RS to $[0, \varepsilon')$, then the reversal of LS has the right-shift property, since it produces the original correct representation $\mathcal{R}$. But the right-shift property depends only on the relative sizes of the shifts and not on the precise values. Therefore, we can construct RS from the reversal of LS by keeping the shifts in the same relative order. If $\mathrm{LS}(v_i)$ is one of the kth smallest shifts, we set $\mathrm{RS}(v_i) = (k-1)\varepsilon$. Since pre-drawn intervals are never moved, $\hat{\mathcal{R}}$ extends $\mathcal{R}'$. It is easy to check that it is a correct unit interval representation. □

If the partial representation $\mathcal{R}'$ is empty, then the above lemma non-constructively proves that every unit interval graph has a representation in the grid of resolution $\frac{1}{n}$.

Linear Programming. We can deal with unlocated components separately as before, using the algorithm of [73]. On the other hand, located components, ordered by $\blacktriangleleft$ from left-to-right, restrict each other as depicted in Fig. 2.10. Therefore, we process components $C_1 \blacktriangleleft C_2 \blacktriangleleft \cdots \blacktriangleleft C_c$ from left to right and try to push each component as far to the left as possible, to leave maximum space for remaining components. Suppose that we process the component C_t and let E_{t-1} be the rightmost endpoint of previously placed C_{t-1}, with $E_0 = -\infty$.

It is proved in [212] that only two left-to-right orderings of intervals in C_t have to be considered. Let $v_1 < \cdots < v_k$ be one of them. We solve the following linear program, where the variable ℓ_i determines $\ell(v_i)$:

$$\begin{aligned}
&\text{Minimize:} & E_t &:= \ell_k + 1, & & \\
&\text{subject to:} & \ell_i &\leq \ell_{i+1}, & &\forall i = 1, \dots, k-1, & (2.2)\\
& & \ell_i &= \ell(v_i), & &\forall v_i \in \boldsymbol{V}(G'), & (2.3)\\
& & \ell_i &\geq \ell_j - 1, & &\forall v_iv_j \in \boldsymbol{E}(G), v_i < v_j, & (2.4)\\
& & \ell_i + \varepsilon &\leq \ell_j - 1, & &\forall v_iv_j \notin \boldsymbol{E}(G), v_i < v_j. & (2.5)
\end{aligned}$$

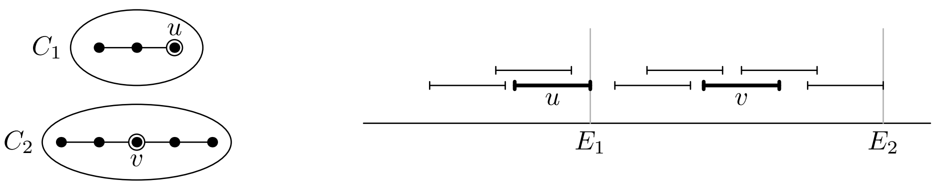


Figure 2.10: The intervals $\langle u \rangle$ and $\langle v \rangle$ are pre-drawn. The component C_1 can only be represented with $\langle u \rangle$ being the right-most interval, since otherwise C_1 would block space for the component C_2.

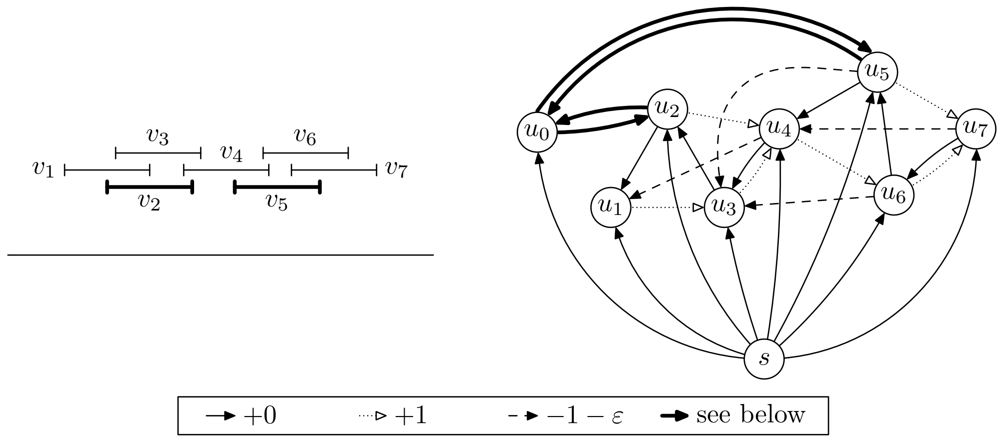


Figure 2.11: On the left, a unit interval graph with two pre-drawn intervals. On the right, the corresponding digraph D with the weights encoded as in the box. The weights of the bold edges are as follows: $w(u_0, u_2) = \ell(v_2)$, $w(u_0, u_5) = \ell(v_5)$, $w(u_2, u_0) = -\ell(v_2)$, and $w(u_5, u_0) = -\ell(v_5)$.

The constraint (2.2) corresponds to the left-to-right ordering $<$, (2.3) ensures that $\mathcal{R}$ extends $\mathcal{R}'$, and (2.4) and (2.5) ensure correctness of constructed unit interval representation. We solve two linear programs for each C_t and we use the smaller value of E_t (if both are solvable) in subsequent linear programs for C_{t+1}.

The above linear programs can be rewritten into a *system of difference constraints*. In optimization, it is well-known that such a linear program can be solved by applying Bellman-Ford algorithm on a weighted digraph D, see Fig. 2.11 for an example, and [212] and [71, Chapter 24.4] for details.

Proposition 2.3.4 (Klavík et al. [211, 212]). *The problem* RepExt(UNIT INT) *can be solved in time $\mathcal{O}(n^2r + nD(r))$ where n the number of vertices, r is the size of input and $D(r)$ is the complexity of dividing numbers of length r.*

Left-shifting Algorithm. A faster algorithm solving the above linear program is described in [211, 212] which leads to solving RepExt(UNIT INT) in time $\mathcal{O}(n^2 + nD(r))$. We relax the condition (2.3) to $\ell_i \geq \ell(v_i)$. We construct the *left-most ε-grid representation* $\mathcal{R}$ of C_t which simultaneously minimizes all $\ell_1, \dots, \ell_k$; it is proved in [212, Corollary 4.6] that it always exists. Therefore, the partial representation $\mathcal{R}'$ is extendible if and only if the left-most representation $\mathcal{R}$ satisfies $\ell_i = \ell(v_i)$, i.e., it extends $\mathcal{R}'$.

We start with some unit interval representation far enough to the right; it can be constructed using [73]. We transform it by a series of *left shifts*, each decreasing some ℓ_i by ε while preserving the correctness of the unit interval representation; see Fig. 2.12a for an example. In [212, Proposition 4.8], it is proved that, assuming ε is sufficiently small, a representation is the left-most representation if and only if no interval can be left-shifted. This is proved by showing that an *obstruction digraph* H, having $(v_i, v_j) \in \boldsymbol{E}(H)$ if and only if v_j has to be left-shifted before v_i, is always acyclic; see Fig. 2.12b.

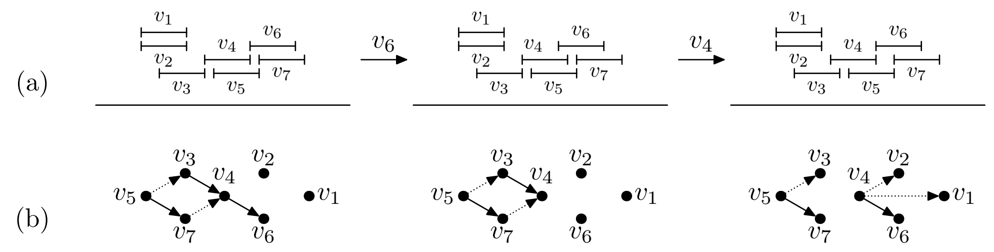


Figure 2.12: (a) An ε-grid representation modified by left-shifting of v_6 and v_4. (b) The corresponding obstruction digraphs H for each of the representations. Only sinks of the obstruction digraphs can be left-shifted. There are two types of edges called *left edges* (depicted dotted) and *right edges* (depicted normally).

To reach the running time $\mathcal{O}(k^2 + kD(r))$ for each component, several tricks has to be employed. Instead of shifting by ε, we use left shifts as long as possible. We also work in the ε-grid for $\varepsilon = \frac{\varepsilon'}{n^2}$. Geometrically, the algorithm computes a Manhattan walk inside the polytope given by the linear program described before, consisting of $\mathcal{O}(k^2)$ steps. For details, see [212].

Synthetic Graphs. In the language of semiorders, which are unit interval orders, similar linear programs without pre-drawn intervals were studied by Pirlot [295, 296]. For a unit interval representation, its synthetic graph S is obtained from the digraph depicted in Fig. 2.11 by removing u_0 and s. By using weight as in Fig. 2.11, lengths of shortest paths in S encode informations about relative positions of endpoints between intervals in every ε-grid representation. For every $i < j$, the difference $\ell(u_i) - \ell(u_j)$ is less or equal the length of a shortest path from u_j to u_i in S. For instance, a shortest path in Fig. 2.11 from u_6 to u_1 is $u_6 \to u_3 \to u_4 \to u_1$, so in every unit interval representation with this ordering of intervals, we have $\ell(u_1) \le \ell(u_6) - 1 - 2\varepsilon$. Also, every obstruction digraph H is a subgraph of the synthetic graph S.

Recently, Soulignac [323, 324] extended synthetic graphs to unit circular-arc representations, building a unified framework for many different problems involving unit circular-arc representations. He derived a faster algorithm solving RepExt(UNIT INT) in time $\mathcal{O}(n^2{+}r)$. More generally, he solved the problem RepExt(UNIT CIRCULAR-ARC) in time $\mathcal{O}(n^2 + r)$, assuming that the circular order of arcs is specified by the input. This assumption cannot be lifted since Zeman [363] proved that the problem RepExt(UNIT CIRCULAR-ARC) is NP-complete. (The reduction is similar to the one used in [211, 212, 213, 214] and in the proof of Theorem 2.4.1.)

2.4 k-nested and k-length Interval Graphs

In Chapter 5, we prove the following results for interval graphs of limited nesting and count of lengths, described in Section 1.3.2.

Hardness of Extending Partial Representations with Two Lengths. In [201], a polynomial-time algorithm is given for recognizing 2-LengthINT when intervals are partitioned into two subsets A and B, each of one length, and both $G[A]$ and $G[B]$ are connected. This approach might be generalized for partial representation extension, but we show that removing the connectedness condition makes it hard:

Theorem 2.4.1. *The problem* RepExt(2-LengthINT) *is* NP*-hard when every pre-drawn interval is of one length a. It remains* NP*-hard even when*

(i) the input prescribes two lengths $a = 1$ and b, and
(ii) for every interval, the input assigns one of the lengths a or b.

Also, the problem is W[1]*-hard when parameterized by the number of pre-drawn intervals.*

Computing Minimal Nesting. We describe a dynamic programming algorithm for recognizing k-NestedINT, based on MPQ-trees (see 4.2). We show that we can optimize nesting greedily from the bottom to the top. We compute a so-called minimal representation for each subtree and we show how to combine them.

Theorem 2.4.2. *The minimum nesting number $\nu(G)$ can be computed in time $\mathcal{O}(n+m)$ where n is the number of vertices and m is the number of edges. Therefore, the problem* Recog(k-NestedINT) *can be solved in linear time.*

This result has the following application in the computational complexity of deciding logic formulas over graphs called *FO property testing.* Let φ be the length of a first-order logic formula for graphs. By the locality, this formula can be decided in G in time $n^{\mathcal{O}(\varphi)}$. Since it is W[2]-hard to decide it for general graphs when parameterized by φ, it is natural to ask for which graph classes there exists an FPT algorithm running in time $\mathcal{O}(n^c \cdot f(\varphi))$ for some computable function f.

In [138], it is shown that FO property testing is W[2]-hard even for interval graphs. On the other hand, if an interval graph is given together with a k-length interval representation, [138] gives an FPT algorithm with respect to the parameters φ and the particular lengths of the intervals. It was not clear whether such an algorithm is inherently geometric. Recently, Gajarský et al. [136] give a different FPT algorithm for FO property testing for interval graphs parameterized by φ and the nesting k, assuming that a k-nested interval representation is given by the input. By our result, this assumption can be removed since we can compute an interval representation of the optimal nesting in linear time.

Related Results and Research Directions. Since k-NestedINT seem to share many properties with proper interval graphs, several future directions of research are immediately offered. Using our results, it is possible to describe minimal forbidden induced subgraphs [179]. For the computational problems which are tractable for proper interval graphs and hard for interval graphs, the complexity of the intermediate problems for k-NestedINT can be studied. In Lemma 5.2.2, we show that k-NestedINT can be efficiently encoded, similarly to proper interval graphs. See Section 5.4 for more discussion.

Partial Representation Extension. The problem RepExt(k-NestedINT) is more involved since a straightforward greedy optimization from the bottom to the top does not work. The described recognition algorithm can be generalized to solve RepExt(k-NestedINT) in polynomial time [220]. It contrasts with Theorem 2.4.1. The partial representation extension problems for k-NestedINT and k-LengthINT are problems for which the geometric version (at most k lengths) is much harder than the corresponding topological problem (the left-to-right ordering of endpoints of intervals).

2.5 Chordal graphs

In [213, 214], the partial representation extension problems are studied for chordal graphs and their three subclasses path graphs, interval graphs and proper interval graphs, in the setting of subtree-in-tree representations; recall Section 1.3.3.

Tree Modifications. It is not completely clear how partial representations should be defined. A partial representation $\mathcal{R}'$ prescribes subtrees of $\boldsymbol{V}(G')$ and also specifies some tree T' in which these subtrees are placed. A representation $\mathcal{R}$ uses a tree T which is created by some modification of T'. In [213, 214], four possible modifications are considered, leading to different partial representation extension problems:

- Fixed – the tree cannot be modified at all, i.e, $T = T'$.
- Sub – the tree can only be subdivided, i.e., T is a subdivision of T'.
- Add – we can add branches to the tree, i.e., T' is a subgraph of T.
- Both – we can both add branches and subdivide, i.e, a subgraph of T is a subdivision of T'. In other words T' is a topological minor of T.

We denote the problems by RepExt($\mathcal{C}, \mathfrak{T}$), where $\mathfrak{T}$ denotes the type. See Fig. 2.13. For the classes PROPER INT and INT, we require both T and T' to be paths. For PATH (and implicitly for PROPER INT and INT), we only consider subpaths of T and T' instead of subtrees.

Constructing a representation in a specified tree T' is interesting even if no subtree is pre-drawn, i.e., G' is empty; this problem is denoted by Recog*($\mathcal{C}, \mathfrak{T}$). Clearly, the hardness of the Recog* problem implies the hardness of the corresponding RepExt problem.

Complexity Results. The complexity of the Recog* and RepExt problems for all four classes and all four types is studied in [213, 214] and the results are displayed in Table 2.1.

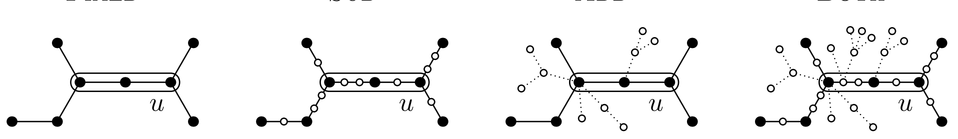


Figure 2.13: The four possible modifications of T' with a single pre-drawn vertex u. The added branches in T are denoted by dots and new vertices of T are denoted by small circles.

		PROPER INT	INT	PATH	CHOR
FIXED	RECOG*	$\mathcal{O}(n+m)$ [214]	$\mathcal{O}(n+m)$ [214]	NP-complete [214]	NP-complete [214]
	REPEXT	NP-complete [214]	NP-complete [214]	NP-complete [214]	NP-complete [214]
SUB	RECOG*	$\mathcal{O}(n+m)$ [257, 73]	$\mathcal{O}(n+m)$ [39, 75]	NP-complete [214]	NP-complete [214]
	REPEXT	$\mathcal{O}(n+m)$ [214]	$\mathcal{O}(n+m)$ [214]	NP-complete [214]	NP-complete [214]
ADD	RECOG*	$\mathcal{O}(n+m)$ [257, 73]	$\mathcal{O}(n+m)$ [39, 75]	$\mathcal{O}(nm)$ [146, 316]	$\mathcal{O}(n+m)$ [311]
	REPEXT	$\mathcal{O}(n+m)$ [214]	NP-complete [214]	NP-complete [214]	NP-complete [214]
BOTH	RECOG*	$\mathcal{O}(n+m)$ [257, 73]	$\mathcal{O}(n+m)$ [39, 75]	$\mathcal{O}(nm)$ [146, 316]	$\mathcal{O}(n+m)$ [311]
	REPEXT	$\mathcal{O}(n+m)$ [211, 212]	$\mathcal{O}(n+m)$ [34, 215]	**Open**	NP-complete [214]

Table 2.1: The complexity of different problems for the four considered graph classes.

- All NP-completeness results are reduced from the 3-PARTITION problem. The reductions are very similar to [211, 212] and the reduction of the proof of Theorem 2.4.1.
- The polynomial cases for INT and PROPER INT are based on the algorithms for recognition and extension, described in Sections 2.2 and 2.3. Since the space in T is limited, these algorithms are adapted for the specific problems.

Also some basic parameterized results are described for three parameters: the number of connected components, the number of pre-drawn subtrees and the size of the tree T'; see [213, 214] for details.

Every interval graph has a real-line representation in which all endpoints are at integer positions. But the result that REPEXT(INT, ADD) is NP-complete can be interpreted in the way that extending such representations is NP-complete. (Here, we require that also the non-pre-drawn intervals have endpoints placed at integer positions.) On the other hand, the linear-time algorithm for REPEXT(PROPER INT, ADD) shows that integer-position proper interval representations can be extended in linear time.

Topological *H*-graphs. The above results are related to the following graph classes, introduced in [31]. Let H be some graph. The class of all topological H-graphs, denoted H-GRAPH, consists of all intersection graphs of connected subgraphs in some subdivision of H. We get that INT $= K_2$-GRAPH, CIRCULAR-ARC $= K_3$-GRAPH, and when T is a tree, T-GRAPH $\subsetneq$ CHOR.

It was asked in [31] whether, for each fixed H, the class H-GRAPH can be recognized in polynomial time. If H is part of the input, it follows from the results in [213, 214] (Table 2.1, the problem RECOG*(CHOR, SUB)) that the recognition problem of H-GRAPH is NP-complete even when H are trees. The computational complexity of various problems for H-GRAPH was recently considered in [61], including the following results for the recognition problem:

- For stars $S_d = K_{1,d}$, the problem RECOG(S_d-GRAPH) can be solved in time $\mathcal{O}(n^4)$, even when d is a part of the input.

- For each fixed tree T, the problem $\textsc{Recog}(T\text{-}\mathsf{GRAPH})$ can be solved in polynomial time.
- When H contains the diamond (K_4 without an edge) as a minor, then the problem $\textsc{Recog}(H\text{-}\mathsf{GRAPH})$ is NP-complete. This negatively answers the question of [31].

We note that the second results gives an XP algorithm for $\textsc{Recog}^*(\mathsf{CHOR}, \textsc{Sub})$ when parameterized by the size of the tree T, which was an open problem in [213, 214]. The computational complexity of the partial representation extension problem for H-GRAPH remains open.

Problem 2.5.1. *What is the complexity of* $\textsc{RepExt}(\mathsf{CHOR}, \textsc{Sub})$ *when parameterized by the size of the tree T?*

2.6 Circle Graphs

The problem $\textsc{RepExt}(\mathsf{CIRCLE})$ was solved in polynomial time by Chaplick et al. [58, 59]. We first sketch the current state-of-art recognition algorithm based on split decomposition. Then, we explain how it captures all circle representations which can be used to solve the partial representation extension problem.

Split Decomposition. Let G be a connected graph. A *split* of G is a partition $\boldsymbol{V}(G)$ into four parts A, B, $\mathfrak{s}(A)$ and $\mathfrak{s}(B)$, such that:

- For every $a \in A$ and $b \in B$, we have $ab \in \boldsymbol{E}(G)$.
- There is no edge between $\mathfrak{s}(A)$ and $B \cup \mathfrak{s}(B)$, and between $\mathfrak{s}(B)$ and $A \cup \mathfrak{s}(A)$.
- Split is *non-trivial*, meaning that both sides of the split have at least two vertices: $|A \cup \mathfrak{s}(A)| \geq 2$ and $|B \cup \mathfrak{s}(B)| \geq 2$.

Fig. 2.14 shows two possible representations of a split. In other words, between A and B, we have a cut in G which is a complete bipartite graph. Notice that a split is uniquely determined just by the sets A and B, since $\mathfrak{s}(A)$ consists of connected components of $G \setminus (A \cup B)$ attached to A, and similarly for $\mathfrak{s}(B)$ and B. We refer to this split as the split *between* A and B.

Split decompositions are used in the current state-of-the-art algorithms for recognizing circle graphs. If a circle graph contains no split, it is called a *prime graph.* The circular word of a circle representation of a prime graph is unique up to reversal and it can be constructed in polynomial time [135]. There is an algorithm which finds a split in a graph in linear time [82].

Split decomposition can be used to recognize circle graphs as follows. We define two graphs G_A and G_B where G_A is a subgraph of G induced by the vertices corresponding to $A \cup \mathfrak{s}(A) \cup \{m_A\}$ where the newly introduced *marker vertex* m_A is adjacent to all the vertices in A and non-adjacent to all the vertices in $\mathfrak{s}(A)$, and G_B is defined similarly for B, $\mathfrak{s}(B)$, and m_B. We get that G is a circle graph if and only if both G_A and G_B are circle graphs.

A representation $\mathcal{R}$ of G can be constructed as follows. We apply the algorithm recursively on G_A and G_B and construct their representations $\mathcal{R}_A$ and $\mathcal{R}_B$; see Fig. 2.15. Then we join these representations to construct $\mathcal{R}$. To this end we take $\mathcal{R}_A$ and replace $\langle m_A \rangle$ by the representation of $B \cup \mathfrak{s}(B)$ in $\mathcal{R}_B$. More precisely, let $m_A \tau_A m_A \hat{\tau}_A$ and $m_B \tau_B m_B \hat{\tau}_B$ be the circular orderings of $\mathcal{R}_A$ and $\mathcal{R}_B$, respectively. The constructed $\mathcal{R}$ has the corresponding circular ordering $\tau_A \tau_B \hat{\tau}_A \hat{\tau}_B$. It is easy to see that $\mathcal{R}$ is a correct circle representation of G.

Structure of Representations of Maximal Splits. On the other hand, a representation like the one in Fig. 2.14 on the right cannot be constructed by the algorithm using the split between A and B. The following structural results are derived in [58, 59]. A split of G between A and B is *maximal* if there exists no split of G between A' and B' such that $A \subseteq A'$, $B \subseteq B'$ and $|A \cup B| < |A' \cup B'|$. We start with an arbitrary split and we move vertices for $\mathfrak{s}(A)$ to B and from $\mathfrak{s}(B)$ to A until a maximal split is reached. We ignore the third condition, so trivial maximal splits having, say, $A = \{u\}$ and $\mathfrak{s}(A) = \emptyset$ are allowed; then u is an articulation in G.

Let $\mathcal{R}$ be a circle representation of a maximal split between A and B. It corresponds to a circular word $\tau = \tau_1 \tau_2 \dots \tau_{2k}$ where τ_i, for i odd, consists only of endpoints of chords in $A \cup \mathfrak{s}(A)$, and τ_i, for i even, consists only of endpoints of chords in $B \cup \mathfrak{s}(B)$. Suppose that we cyclically work with the indexes, the following properties are proved in [58, 59] for A (and hold symmetrically for B):

- For each $u \in A$, the endpoints of u appears once in τ_i and once in τ_{i+k}. Therefore, u is called a *long vertex.*
- For each $v \in \mathfrak{s}(A)$, both endpoints of v appear in some τ_i. Thus, v is called a *short vertex.*
- Let $u, v \in A$. If $uv \notin \boldsymbol{E}(G)$, or there exists a non-trivial path (different from the edge uv) from u to v with all internal vertices in $\mathfrak{s}(A)$, then the endpoints of both u and v appear in the same τ_i and τ_{i+k}.

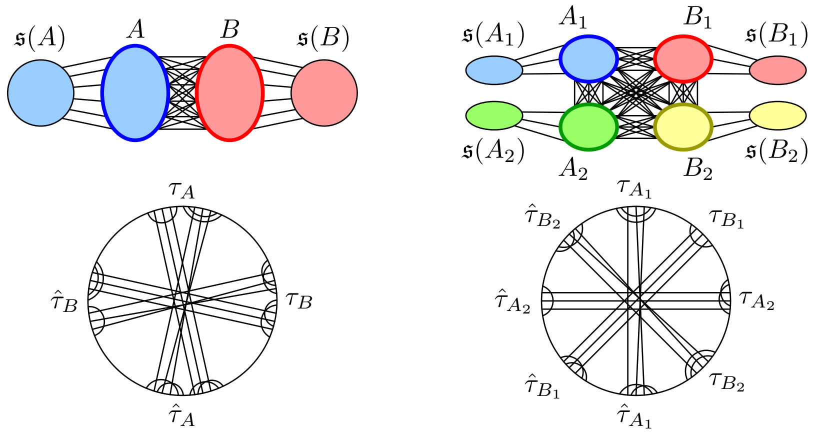


Figure 2.14: Two different representations of G with the split between A and B. The circular subword $\tau_A \hat{\tau}_A$ is induced by $A \cup \mathfrak{s}(A)$, the circular subword $\tau_B \hat{\tau}_B$ by $B \cup \mathfrak{s}(B)$, and similarly on the right.

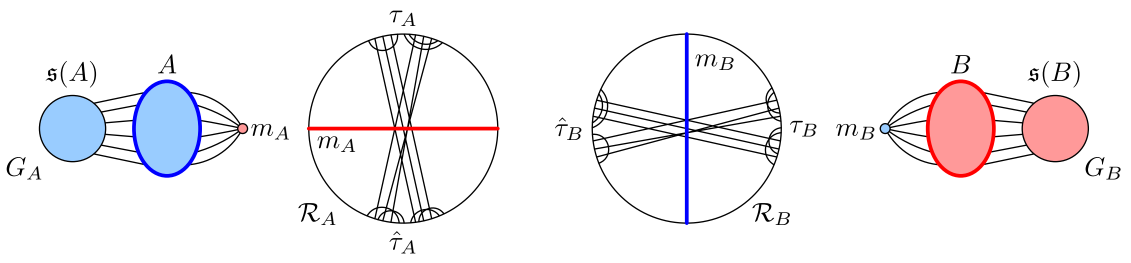


Figure 2.15: The graphs G_A and G_B together with some constructed representations $\mathcal{R}_A$ and $\mathcal{R}_B$. By joining these representations, we get the representation shown in Fig. 2.14 on the left.

Next, it is studied what are all possible representations of a maximal split between A and B. Inspired by Naji [285, Section IV.4], the following relation $\sim$ is defined on $A \cup B$ where $x \sim y$ means that x and y has to be placed in the same subword τ_i of τ. This relation follows from the last property proved in [58, 59]:

- If $xy \notin E(G)$, then $x \sim y$. In particular, $x \sim x$.
- If x and y are connected by a non-trivial path with all the inner vertices in $\mathfrak{s}(A) \cup \mathfrak{s}(B)$, then $x \sim y$.

Let $\sim$ be the transitive closure of the above. We obtain an equivalence relation $\sim$ on $A \cup B$. Notice that every equivalence class of $\sim$ is either fully contained in A or in B. For instance in Fig. 2.14 on the right, the relation $\sim$ has four equivalence classes A_1, A_2, B_1 and B_2.

We choose an arbitrary circular ordering $\Phi_1, \dots, \Phi_\ell$ of the classes of $\sim$. Let G_i be a graph constructed from G by contracting the vertices $V(G) \setminus \bigl(\Phi_i \cup \mathfrak{s}(\Phi_i)\bigr)$ into the marker vertex m_i; i.e., G_i is defined similarly to G_A and G_B above. Let $\mathcal{R}_1, \dots, \mathcal{R}_\ell$ be arbitrary representations of $G_1, \dots, G_\ell$. We join these representations as follows. Let $m_i \tau_i m_i \hat\tau_i$ be the circular ordering of $\mathcal{R}_i$. We construct $\mathcal{R}$ as the circular ordering

$$\tau_1 \tau_2 \dots \tau_{\ell-1} \tau_\ell \hat\tau_1 \hat\tau_2 \dots \hat\tau_{\ell-1} \hat\tau_\ell. \tag{2.6}$$

In Fig. 2.14, we obtain the representation on the left by the circular ordering $A_1 A_2 B_1 B_2$ of the classes of $\sim$ and the representation on the right by $A_1 B_1 A_2 B_2$. Every such circular word defines a correct circle representation. In [59, Proposition 1], it is proved that every circle representation can be constructed as in (2.6) by choosing some representations $\mathcal{R}_1, \dots, \mathcal{R}_\ell$ and some circular ordering of the classes of $\sim$.

Partial Representation Extension. We sketch the following polynomial-time algorithm for RepExt(CIRCLE) described in [59, Chapter 4], with many details omitted:

Theorem 2.6.1 (Chaplick et al. [58, 59]). *The problem* RepExt(CIRCLE) *can be solved in polynomial time.*

It is easy to deal with disconnected graphs, so we can assume that G is connected. If G is a prime graph, then it has at most two different representations $\mathcal{R}$ and $\hat{\mathcal{R}}$ [135] where one is reversal of the other. We just need to test whether one of them extends $\mathcal{R}'$.

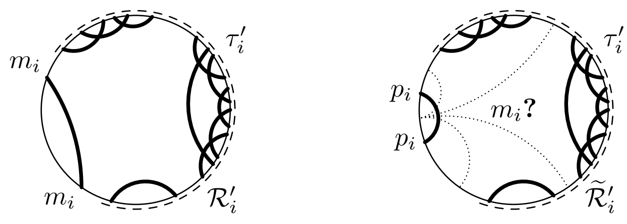


Figure 2.16: The partial representation $\widetilde{\mathcal{R}}_i'$ is less restrictive with respect to the position of m_i. Therefore it might be extendible even when $\mathcal{R}_i'$ is not.

Otherwise, we find a maximal split between A and B in polynomial time using [82]. If it is trivial, we deal with the components of the non-trivial side using a special subroutine. Otherwise, we compute the relation $\sim$ and the equivalent classes. We use the partial representation $\mathcal{R}'$ to derive their circular ordering and recurse on graphs G_i with $\mathcal{R}_i'$. Let Φ be a class of $\sim$ and let $\Psi = \Phi \cup \mathfrak{s}(\Phi)$ be the *extended class* where $\mathfrak{s}(\Phi)$ are all components of $\mathfrak{s}(A) \cup \mathfrak{s}(B)$ attached to Φ. If no chord of Ψ is pre-drawn, we can ignore it.

Let τ' be the circular word of $\mathcal{R}'$. Let $\tau' = \tau_1' \dots \tau_k'$ where each τ_i' is a maximal subwords containing only symbols of one extended class Ψ. By (2.6), each extended class Ψ corresponds to at most two different maximal subwords, otherwise we reject the input. Also, if two extended classes Ψ and $\hat{\Psi}$ each correspond to two different maximal subwords, then occurrences of these subwords alternate in τ'. Otherwise we again reject the input. We distinguish two cases.

Case 1: Some extended class corresponds to two maximal subwords. We can use it to derive the circular ordering of the classes. We construct the graph G_i as above. As the partial representation $\mathcal{R}_i'$ of G_i, we put the word $m_i \tau_i' m_i \tau_j'$ where Ψ_i corresponds to τ_i' and τ_j' (possibly one of them is empty). We test recursively, whether each representation $\mathcal{R}_i'$ of G_i is extendible to a representation of $\mathcal{R}_i$. If yes, we join $\mathcal{R}_1, \dots, \mathcal{R}_\ell$ as in (2.6) Otherwise, the algorithm outputs "no".

Case 2: No extended class corresponds to two maximal subwords. More circular orderings are possible and the situation is more involved. Suppose that τ_i' corresponds to an extended class Ψ_i with the graph G_i. The problem is that τ_i' might be extended in $\mathcal{R}$ either to one of τ_j or $\hat{\tau}_j$ in (2.6), or to both of them; and the latter option restricts the remainder of extended classes as in Case 1. To simulate this choice, we construct two partial representations of G_i depicted in Fig. 2.16, where the second one is for the graph $G_i \cup \{p_i\}$ where p_i is a new neighbor of m_i which pins one endpoint of $\langle m_i \rangle$ to be outside of τ_i'. Clearly, $\widehat{\mathcal{R}}_i'$ is less restrictive. It is proved in [58, 59] that $\mathcal{R}'$ is extendible if and only if $\widehat{\mathcal{R}}_i'$ is extendible for some extended class Ψ_i and $\mathcal{R}_j'$ is extendible for every other extended class Ψ_j.

Split Trees. A split decomposition of G works as follows. Consider a split between A and B. We replace G by the graphs G_A and G_B defined above. Then we apply the decomposition recursively on G_A and G_B, and we stop on prime graphs containing no splits. We note that by different orders of splits, different decompositions of G may be constructed. A split decomposition can be computed in linear time [150].

A split decomposition is called *minimal* if it is constructed by the least number of

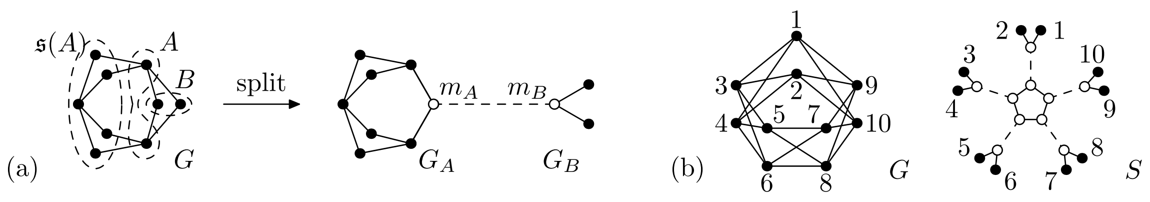


Figure 2.17: (a) An example of a split of the graph G. The marker vertices are depicted in white. The tree edge is depicted by a dashed line. (b) The split tree S of the graph G.

splits. Suppose that we also stop on *degenerate graphs* which are complete graphs K_n and stars $S_n = K_{1,n}$. Cunningham [79] proved that the minimal split decomposition of a connected graph stopping on prime and degenerate graphs is unique.

The unique *split tree* S representing a graph G encodes the minimal split decomposition. A split tree is a graph with two types of vertices (normal and marker vertices) and two types of edges (normal and tree edges). We initially put $S = G$ and modify it according to the minimal split decomposition. If the minimal decomposition contains a split between A and B in G, then we replace G in S by the graphs G_A and G_B, and connect the marker vertices m_A and m_B by a *tree edge* (see Fig. 2.17a). We repeat this recursively on G_A and G_B; see Fig. 2.17b. Each prime and degenerate graph is a *node* of the split tree. A node that is incident with exactly one tree edge is called a *leaf node*.

We note that it is not clear how fast the minimal split decompositions and the split trees can be computed. In [150], this issue is not discussed at all. At this moment, it is only clear that it can be computed in polynomial time using results of Cunningham [79]. Also, it should be possible to derive every circle representation of a connected graph G from the split tree S, but the precise statement is unclear. Therefore, the following natural problem remains open:

Problem 2.6.2. *Is it possible to use split trees S to solve* RepExt(CIRCLE)*?*

2.7 Partial Orientation Extension Problems

In Section 1.3.4, we have described comparability graphs which are graphs whose edges can be transitively oriented, and their relation to several classes of intersection graphs: function graphs, permutation graphs, trapezoid graphs, and others. Also, proper interval and proper circular-arc graphs can be described in the language of orientations which are local tournaments [94].

In this section, we deal with partial orientation extension problems, first considered in general by Bang-Jensen et al. [20]. Let $\mathfrak{orient}$ be a class of admissible orientations. A *partial orientation* $\to'$ is an orientation of a subset of edges. An orientation $\to$ of all edges extends the partial orientation $\to'$ if the orientation in $\to'$ is preserved: $x \to' y$ implies $x \to y$. We study the following computational problem:

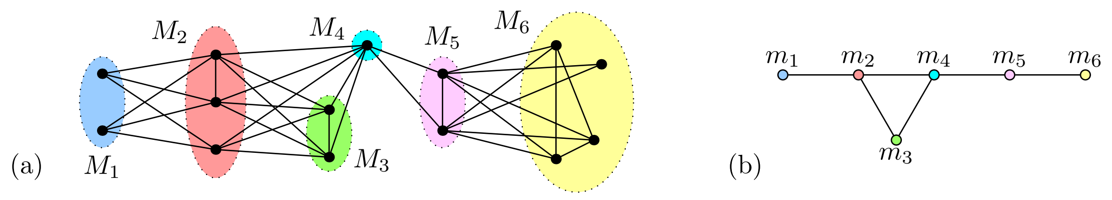


Figure 2.18: (a) A graph G with a modular partition $\mathcal{P}$. (b) The quotient graph $G/\mathcal{P}$ is prime.

Problem:	Partial orientation extension – $\textsc{OrientExt}(\mathcal{C}, \mathfrak{orient})$
Input:	A graph $G \in \mathcal{C}$ and a partial orientation $\to' \in \mathfrak{orient}$ of G.
Question:	Is there an orientation $\to \in \mathfrak{orient}$ of G extending $\to'$?

For instance, let TRANSITIVE be the class of all transitive orientations. It is proved in [210] that the problem $\textsc{OrientExt}(\mathsf{COMP}, \mathsf{TRANSITIVE})$ can be solved in polynomial time. We also describe its implications for the complexity of the partial representation extension problems of permutation and function graphs [210] and trapezoid graphs [243]. The results concerning partial representations extension of proper interval and proper circular-arc graphs of [20] are also discussed.

2.7.1 Modular Decomposition and Modular Trees

A *module* M of a graph G is a set of vertices such that each $x \in \boldsymbol{V}(G) \setminus M$ is either adjacent to all vertices in M, or to none of them. See Fig. 2.18a for examples. A module M is called *trivial* if $M = \boldsymbol{V}(G)$ or $|M| = 1$, and *non-trivial* otherwise. If M and M' are two disjoint modules, then either the edges between M and M' form the complete bipartite graph, or there are no edges at all; see Fig. 2.18a. In the former case, M and M' are called *adjacent*, otherwise they are *non-adjacent*.

Let $\mathcal{P} = \{M_1, \dots, M_k\}$ be a *modular partition* of $\boldsymbol{V}(G)$, i.e., each M_i is a module of G, $M_i \cap M_j = \emptyset$ for every $i \neq j$, and $M_1 \cup \cdots \cup M_k = \boldsymbol{V}(G)$. We define the *quotient graph* $G/\mathcal{P}$ with the vertices $m_1, \dots, m_k$ corresponding to $M_1, \dots, M_k$ where $m_i m_j \in \boldsymbol{E}(G/\mathcal{P})$ if and only if M_i and M_j are adjacent. In other words, the quotient graph is obtained by contracting each module M_i into the single vertex m_i; see Fig. 2.18b.

Modular Decomposition. To decompose G, we find some modular partition $\mathcal{P} = \{M_1, \dots, M_k\}$, compute $G/\mathcal{P}$ and recursively decompose $G/\mathcal{P}$ and each $G[M_i]$. The recursive process terminates on *prime graphs* which are graphs containing only trivial modules. There might be many such decompositions for different choices of $\mathcal{P}$ in each step. In 1960s, Gallai [137] described the *modular decomposition* in which special modular partitions are chosen and which encodes all other decompositions.

The key is the following observation. Let M be a module of G and let $M' \subseteq M$. Then M' is a module of G if and only if it is a module of $G[M]$. A graph G is called *degenerate* if it is K_n or $\overline{K}_n$, and if the distinction is needed, it is called *complete* and *independent*, respectively. We construct the modular decomposition of a graph G in the following way, see Fig. 2.19a for an example:

- If G is a prime or a degenerate graph, then we terminate the modular decomposition on G. We stop on degenerate graphs since every subset of vertices forms a module, so it is not useful to further decompose them.

- Let G and $\overline{G}$ be connected graphs. Gallai [137] shows that the inclusion maximal proper subsets of $\boldsymbol{V}(G)$ which are modules form a modular partition $\mathcal{P}$ of $\boldsymbol{V}(G)$, and the quotient graph $G/\mathcal{P}$ is a prime graph; see Fig. 2.18. We recursively decompose $G[M]$ for each $M \in \mathcal{P}$.

- If G is disconnected and $\overline{G}$ is connected, then every union of connected components is a module. Therefore the connected components form a modular partition $\mathcal{P}$ of $\boldsymbol{V}(G)$, and the quotient graph $G/\mathcal{P}$ is independent. We recursively decompose $G[M]$ for each $M \in \mathcal{P}$.

- If $\overline{G}$ is disconnected and G is connected, then the modular decomposition is defined in the same way on the connected components of $\overline{G}$. They form a modular partition $\mathcal{P}$ and the quotient graph $G/\mathcal{P}$ is complete. We recursively decompose $G[M]$ for each $M \in \mathcal{P}$.

An alternative description is that a module M is called *strong* if for every other module M', either $M \cap M' = \emptyset$, or $M \subseteq M'$ or $M' \subseteq M$. For every graph G, its inclusion maximal proper subsets of $\boldsymbol{V}(G)$ which are strong modules form the modular partition $\mathcal{P}$ of the modular decomposition.

Modular Tree. We encode the modular decomposition by the *modular tree* T. The modular tree T is a rooted tree consisting of *nodes* connected by *directed tree edges*. Nodes are prime and degenerate graphs encountered in the modular decomposition as quotients and terminal graphs, and we also use the names *prime*, *degenerate*, *complete*, and *independent* for them. *Leaf nodes* correspond to the terminal graphs in the modular decomposition, and *inner nodes* are the quotients in the modular decomposition. A *subtree* of the modular tree consists of some node and all its descendants. The subtrees correspond one-to-one to the strong modules of G.

We give a recursive definition. Every modular tree has a *root node*. If G is a prime or a degenerate graph, then T consists of a single node G as its root node. Otherwise, let $\mathcal{P} = \{M_1, \ldots, M_k\}$ be the used modular partition of G and let $T_1, \ldots, T_k$ be the

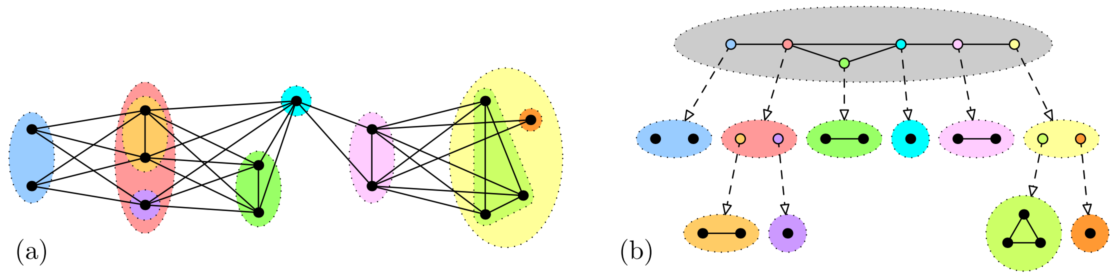


Figure 2.19: (a) The graph G from Fig. 2.18 with the modular partitions used in the modular decomposition. (b) The modular tree T of G, the marker vertices are colored, the tree edges are dashed.

modular trees corresponding to $G[M_1], \dots, G[M_k]$. The modular tree T consists of the disjoint union of $T_1, \dots, T_k$ and of the root node $G/\mathcal{P}$ with the marker vertices $m_1, \dots, m_k$, and we connect each m_i with the root node of T_i by a directed tree edge. For an example, see Fig. 2.19b.

The modular tree T captures adjacencies in G since $xy \in \boldsymbol{E}(G)$ if and only if the corresponding vertices in the common ancestor node of T are adjacent. All vertices of G are in leaf nodes and all inner nodes consist of marker vertices. Each marker vertex corresponds to some strong module in G. The modular tree T of G is unique, and it can be computed in linear time [77, 272, 337].

Recognition of Interval Graphs. As we discuss below, the modular decomposition is a useful tool to work with comparability graphs since it captures all transitive orientations. In Section 1.3.4, we have discussed that $\mathsf{INT} = \mathsf{co\text{-}COMP} \cap \mathsf{CHOR}$. Therefore, modular decomposition can be efficiently applied to interval graphs. Hsu [195] proved that prime interval graphs have exactly two different interval representations, one reversal of the other. Modular decomposition can be used to recognize interval graphs in linear time, see [169] and the references therein. It is an interesting open problem whether the partial representation extension problem for interval graphs can be solved using modular decomposition as well.

2.7.2 Transitive Orientations and Comparability Graphs

We deal with comparability graphs and with the partial orientation extension problem for transitive orientations. First, we discuss that modular trees capture all transitive orientations.

Structure of Transitive Orientations. Let $\rightarrow$ be a transitive orientation of G and let T be the modular tree. For modules M_1 and M_2, we write $M_1 \rightarrow M_2$ if $x_1 \rightarrow x_2$ for all $x_1 \in M_1$ and $x_2 \in M_2$. Gallai [137] shows the following properties. If M_1 and M_2 are adjacent strong modules, then either $M_1 \rightarrow M_2$, or $M_1 \leftarrow M_2$. The graph G is a comparability graph if and only if each node of T is a comparability graph. Every prime comparability graph has exactly two transitive orientations, one being the reversal of the other.

The modular tree T encodes all transitive orientations as follows. For each prime node of T, we arbitrarily choose one of the two possible orientations. For each complete node, we choose one of $n!$ possible orientations. Each independent node has the unique orientation since it contains no edges. A transitive orientation of G is then constructed as follows. We orient the edges of leaf nodes as above. For a node N partitioned in the modular decomposition by $\mathcal{P} = \{M_1, \dots, M_k\}$, we orient $M_i \rightarrow M_j$ if and only if $m_i \rightarrow m_j$ in N. It is easy to check that this gives a valid transitive orientation, and every transitive orientation can be constructed from some transitive orientations of the nodes of T as described above. Figure 2.20 shows an example.

Figure 2.21 shows how to construct a function representation of the complement of a comparability graph by combining function representations of all nodes of the modular tree. By the results of [137, 158], in every function representation, the functions of one strong module always appear consecutively, so they can be replaced by

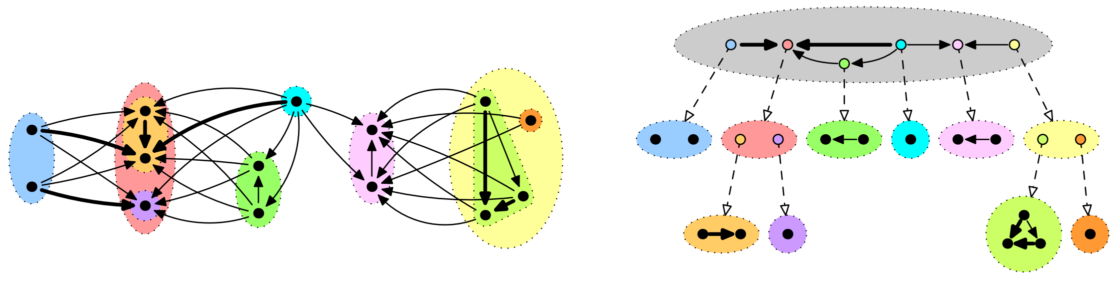


Figure 2.20: A graph G with a partial orientation $\to'$ depicted in bold and its modular tree T. All edges between the blue and red modules correspond to one edge in the root node, so their orientations in $\to'$ has to agree, otherwise $\to'$ is not extendible.

one strip representing the entire module. Therefore, this description allows to construct every function representation $\mathcal{R}$ of $\overline{G}$ by choosing a transitive orientation of G, choosing a function representation for each node of the modular tree respecting this orientation, and putting these representations together as in Fig. 2.21.

Extending Partial Orientations. We can use the modular decomposition to find an extending transitive orientation if it exists. If $x \to' y$, $x \in M_x$, $y \in M_y$, and M_x and M_y are adjacent strong modules, then necessarily $M_x \to M_y$ in every transitive orientation extending $\to'$. In the modular tree, we orient the corresponding edge $m_x \to' m_y$ in the common ancestor node of x and y. If some edge of the modular tree T is forced to be oriented in two different ways, no extending transitive orientation exists. So we get a derived partial orientation for each node of T.

It remains to find an extending transitive orientation of all nodes. For indepen-

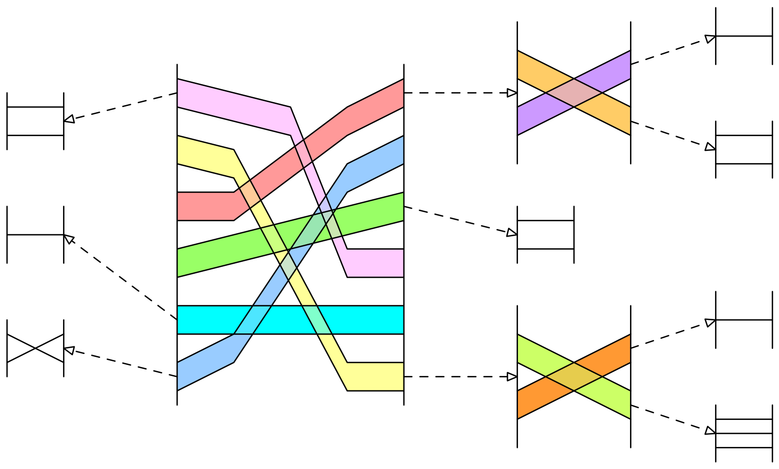


Figure 2.21: Recall from Section 1.3.4 that, by the equality $\mathsf{FUN} = \mathsf{co\text{-}COMP}$ [158], each function representation of a graph gives a transitive orientation of the complement graph. A function representation of $\overline{G}$ from Fig. 2.20 having the depicted transitive orientation of G can be constructed as follows. First, we construct a function representation for each node of the modular tree giving the depicted transitive orientation. Then we replace colored strips representing marker vertices by the corresponding function representations of modules.

dent nodes, we have no edges. For complete nodes, we test whether there exists a linear extension of the partially oriented edges. For prime nodes, we have two possible orientations, one reversal of the other, so we test whether one of them is compatible with the partially oriented edges.

If the partial orientation of some node is non-extendible, then the partial orientation $\to'$ of G is not extendible as well. It remains to translate the extending transitive orientations of the nodes of T into a transitive orientation of G which extends the partial orientation $\to'$. Figure 2.20 shows an example.

We note that a different approach without the modular decomposition is used in [210]. Golumbic [154] described an algorithm running in time $\mathcal{O}((n+m)\cdot\Delta)$ (where Δ is the maximum degree) which constructs a transitive orientation if it is exists. First, it chooses an arbitrary edge and orients it in one way. Then it finds all other edges whose orientation is forced by this choice:

- The orientations $u \to v$ and $v \to w$ force $u \to w$.
- The orientation $u \to v$, for $vw \in \boldsymbol{E}(G)$ and $uw \notin \boldsymbol{E}(G)$, forces $w \to v$.

After that it chooses arbitrarily an orientation of one of the remaining edges, and again finds what is forced. It proceed in this way till the entire graph is transitively oriented, or some edge is forced to be reoriented which means that no transitive orientation exists. This approach is used in [210] to solve OrientExt(COMP, TRANSITIVE): we start with the partial orientation, compute which other orientations are forced, and then proceed with the rest of the algorithm. We note that Golumbic and Shamir [157] used a similar approach to solve a different problem.

Theorem 2.7.1 (Klavík et al. [210])**.** *The problem* OrientExt(COMP, TRANSITIVE) *can be solved in time $\mathcal{O}((n+m)\cdot\Delta)$ where Δ is the maximum degree.*

Concerning the complexity, this problem can be solved faster using modular decomposition. The modular decomposition can be computed in linear time [77, 272, 337]. The extending transitive orientation may be computed in linear time as well. But the bottleneck is checking whether the constructed extending orientation is linear (which is also needed for recognition of comparability graphs), and the fastest known algorithm uses matrix multiplication with the complexity $\mathcal{O}(n^\omega)$.

2.7.3 Permutation Graphs

Theorem 2.7.1 can be directly used to solve the partial representation extension problem for permutation graphs. The reason is that different permutation representations of G (with different orderings of endpoints on two lines) correspond one-to-one to different transitive orderings of G and $\overline{G}$. Recall Section 1.3.4.

Let $\mathcal{R}'$ be a partial permutation representation. Let L_1' be the order of pre-drawn endpoints on one line. For $x, y \in \boldsymbol{V}(G')$, we order $x \to y$ in G or $\overline{G}$, if and only if $x <_{L_1'} y$. These partial orientations of G and $\overline{G}$ fully capture $\mathcal{R}'$ and we get the following:

Theorem 2.7.2 (Klavík et al. [210]). *A partial permutation representation is extendible if and only if both partial orientations of G and $\overline{G}$ are extendible to transitive orientations. We can test this in time $\mathcal{O}(n^3)$.*

We note that using the results of [272], it should be possible to improve the running time to $\mathcal{O}(n+m)$.

2.7.4 Function Graphs

On the other hand, the situation is much more tricky in the case of function graphs. Already in 2010, I observed in my bachelor's thesis that testing whether a partial orientation can be extended to a transitive orientation is not sufficient; see Fig. 2.22. The equality FUN = co-COMP is not robust enough since function representations do not correspond one-to-one to transitive orientations of $\overline{G}$. In my bachelor's thesis and at several conferences, I asked as an open problem what is the complexity of RepExt(FUN). It was solved by Krawczyk and Walczak and the following result was proved in the joined paper [210]:

Theorem 2.7.3 (Klavík et al. [210]). *The problem* RepExt(FUN) *can be solved in polynomial time.*

Extending Partial Representations of Posets. Similarly to co-comparability graphs, every poset $(P,<)$ has a function representation $\big\{\langle u\rangle : u \in P\big\}$ such that $\langle u\rangle \cap \langle v\rangle \neq \emptyset$ if and only if $u,v \in P$ are incomparable (denoted $u \parallel v$), and $\langle u\rangle$ is below $\langle v\rangle$ if and only if $u < v$. Every function representation of a poset is a function representation of the corresponding co-comparability graph with a fixed transitive orientation.

We study the partial representation extension for function representations of posets. Let P' be the subposet represented by a partial representation. If $u < v < w$ and $u,w \in P'$, then necessarily $\langle v\rangle$ is placed in between of $\langle u\rangle'$ and $\langle w\rangle'$. Let $\downarrow \langle u\rangle'$ be the set of all points of $[0,1]\times\mathbb{R}$ below $\langle u\rangle'$, and let $\uparrow \langle u\rangle'$ be the set of all points above it. For each $u \in P \setminus P'$, we get the following restricted *region* $\mathrm{Reg}(v)$ in which $\langle u\rangle$ has to be represented:

$$\mathrm{Reg}(v) = [0,1]\times\mathbb{R} \cap \bigcap\Big\{\downarrow \langle w\rangle' : v < w, w \in P'\Big\} \cap \bigcap\Big\{\uparrow \langle u\rangle' : u < v, u \in P'\Big\}.$$

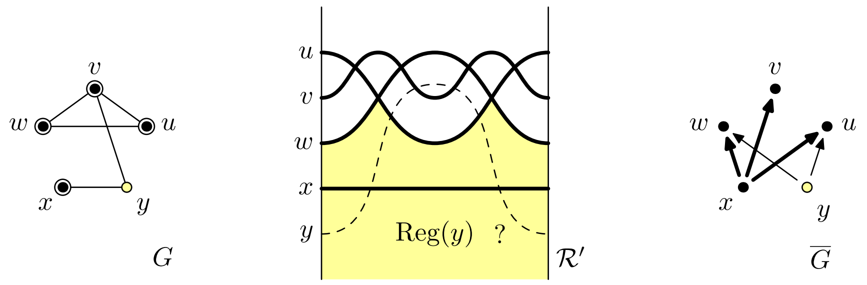


Figure 2.22: A function graph G with a non-extendible partial representation $\mathcal{R}'$ since $\langle u\rangle'$ and $\langle w\rangle'$ together separate $\langle v\rangle'$ and $\langle x\rangle'$. But the corresponding partial orientation of $\overline{G}$ is extendible to a transitive orientation.

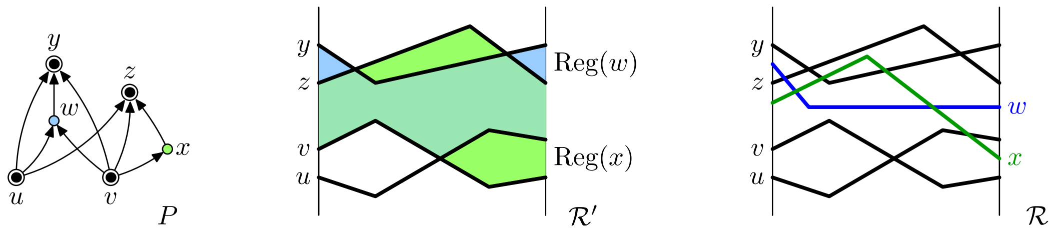


Figure 2.23: A poset P with a partial representation $\mathcal{R}'$ with depicted regions $\mathrm{Reg}(w)$ and $\mathrm{Reg}(x)$. An extending representation $\mathcal{R}$ is on the right.

For each $v \in P'$, let $\mathrm{Reg}(v) = \langle v \rangle'$. For an example, see Fig. 2.23.

Lemma 2.7.4 (Klavík et al. [210]). *A partial function representation of a poset P is extendible if and only if*

$$\forall u, v \in P,\ u \parallel v, \qquad \mathrm{Reg}(u) \cap \mathrm{Reg}(v) \neq \emptyset. \tag{2.7}$$

For instance, in Fig. 2.22, $\mathrm{Reg}(y)$ is the depicted infinite region below both $\langle u \rangle'$ and $\langle w \rangle'$. The partial representation $\mathcal{R}'$ of the poset given by the orientation of $\overline{G}$ is not extendible since $\mathrm{Reg}(y) \cap \mathrm{Reg}(v) = \emptyset$. Since the regions $\mathrm{Reg}(u)$ can be computed in polynomial time and the property can be easily tested, it is possible to solve the partial representation extension problem of posets in polynomial time.

In every extending representation, each $\langle u \rangle \subseteq \mathrm{Reg}(u)$, so the condition (2.7) of non-empty intersections of regions is obviously necessary. For the other direction, we choose functions in such a way that for every incomparable $u, v \in P$, we have $\langle u \rangle \cap \langle v \rangle \neq \emptyset$. Since $\mathrm{Reg}(u) \cap \mathrm{Reg}(v) \neq \emptyset$, we choose some point in it and add it to both functions. It is not difficult to construct a correct extending function representation of P; see [210, Lemma 1] for more detail.

Extending Partial Representations of Functions Graphs. For simplicity, we work in the complement, so we have a partial function representation $\mathcal{R}'$ of $\overline{G}$, and we ask whether there exists a representation $\mathcal{R}$ of $\overline{G}$ extending $\mathcal{R}'$. The partial representation $\mathcal{R}'$ defines a partial orientation of G in which, for all pre-drawn $x, y \in \boldsymbol{V}(G')$, $xy \in \boldsymbol{E}(G)$, we have $x \rightarrow' y$ if and only if $\langle x \rangle'$ is below $\langle y \rangle'$. In other words, precisely the edges of G' are oriented according to the partial representation; see Fig. 2.22.

Every extending representation $\mathcal{R}$ defines some transitive orientation of G extending the partial orientation $\rightarrow'$. By Lemma 2.7.4, we know that the regions for this transitive orientations have to satisfy the condition (2.7) for every $uv \notin \boldsymbol{E}(G)$. But not every transitive orientation of G must satisfy (2.7). In summary, the partial representation extension problem of function graphs reduces to the problem of testing whether there exists a transitive orientation of G extending the partial orientation $\rightarrow'$ which satisfies (2.7).

To solve the latter problem, we use the modular decomposition. From $\rightarrow'$, we derive partial orientations $\rightarrow'$ of the nodes of the modular tree T. We say that a subtree of T is *pre-drawn* if the corresponding strong module contains at least one pre-drawn vertex. Similarly, a strong module a *pre-drawn* if its subtree is pre-drawn.

Reductions. Three reductions are applied which reduce the graph and the modular tree, without changing extendibility of the partial representation. The goal is to reduce the modular tree in such a way that each module has at most two different transitive orientations extending the partial orientation $\to'$.

- Non-predrawn subtrees are replaced by a single vertex u. If there exists an extending representation of the reduced graph, we replace $\langle u \rangle$ by an arbitrary function representation of the subtree, as in Fig. 2.21. After this transformation, every inner node corresponds to a pre-drawn subtree.
- For each complete module with two or more non-predrawn children (each being a singleton leaf node), we remove all but one, called u. If there exists an extending representation of the reduced graph, we can represent the remaining children in parallel with $\langle u \rangle$, right above/below it.
- For each complete module with at least two pre-drawn children and one non-predrawn child, we remove this non-predrawn child. The argument is similar as before.

Aside replacing non-predrawn subtrees by single vertices, no reductions are applied on prime and independent nodes. But this is not needed since they already have at most two different transitive orientations.

Lemma 2.7.5 (Klavík et al. [210])**.** *After the reductions, every node has at most two different transitive orientations extending $\to'$. If it has two transitive orientations, one is the reversal of the other.*

See [210, Lemma 3] for more details. Also, every non-singleton strong module has at least one vertex pre-drawn. Therefore, for any two non-singleton adjacent modules M_1 and M_2, the partial representation $\to'$ determines whether $M_1 \to M_2$, or $M_2 \to M_1$ in every transitive orientation $\to$ extending $\to'$.

Testing (2.7) by a 2-SAT formula. We need to determine possible regions $\mathrm{Reg}(u)$, depending on the chosen transitive orientations of the nodes. Let $u \in \boldsymbol{V}(G)$ be a non-predrawn vertex and let N be the non-singleton node either containing u or being parent of the singleton leaf node $\{u\}$. For every transitive orientation, $\mathrm{Reg}(u)$ depends only on the direction of oriented edges incident with u. But if a pre-drawn vertex v does not belong to the strong module of N, the direction of the edge uv is determined by $\to'$. It is proved in [210, Lemma 4] that $\mathrm{Reg}(u)$ is fully determined by the transitive orientation of the node N, so we have at most two possibilities for every $\mathrm{Reg}(u)$, and we can computed them in polynomial time.

We assign a variable x_N to every node N of the modular tree having two different transitive orientations extending $\to'$. We express the constraints (2.7) by a 2-SAT formula which can be solved in linear time [110, 9]. Therefore, Theorem 2.7.3 is proved.

Partial Functions. In [210], the following generalization of REPEXT(FUN) is also studied. Let G be a comparability graph. For each $u \in \boldsymbol{V}(G)$, a function is prescribed partially on some interval $[a, b] \subseteq [0, 1]$ called the *domain* (possibly empty). We ask whether these partial functions can be extended to full functions on $[0,1]$ such that their graphs define a function representation of G. We get the partial representation extension problem if the pre-drawn functions are prescribed on $[0,1]$ while the non-predrawn ones are prescribed on $\emptyset$.

Instead of a graph G, the problem can be again considered for a poset P. By a suitable generalization of the regions $\mathrm{Reg}(u)$, we get the same result as in Lemma 2.7.4: a representation by partial functions of a poset P is extendible if and only if for every incomparable $u, v \in P$, we have $\mathrm{Reg}(u) \cap \mathrm{Reg}(v) \neq \emptyset$.

To solve the problem for a graph G, we need to decide whether there exists a transitive orientation of G satisfying this constraint. The key difference is that for two adjacent vertices, their ordering in P is prescribed if and only if the domains of their partial functions have non-empty intersection. Therefore, $\mathrm{Reg}(u)$ is not determined by an orientation of one module. By a reduction from 3-SAT, the following is proved in [210]:

Theorem 2.7.6 (Klavík et al. [210]). *The problem of extending representations by partial functions to full functions on* $[0,1]$ *is* NP*-complete.*

2.7.5 Trapezoid Graphs

The first polynomial time algorithm for the partial representation extension problem of trapezoid graphs was proved in a recent breakthrough was by Krawczyk and Walczak [243]:

Theorem 2.7.7 (Krawczyk and Walczak [243]). *The problem* REPEXT(TRAPEZOID) *can be solved in time* $\mathcal{O}(n^5)$.

The algorithm is again based on the modular decomposition, but it is much more involved. It uses the same two basic steps as the algorithm for REPEXT(FUN) described in Section 2.7.4.

First, the partial representation extension problem of trapezoid posets P, i.e., of posets of interval dimension 2, is studied. Recognition algorithms for trapezoid posets make use of *normalization* which is a procedure transforming every representation of a trapezoid poset into a *normalized trapezoid representation* satisfying additional properties. In every normalized trapezoid representation, the left and right sides of the trapezoids form a permutation representation of an appropriately defined 2-dimensional poset called the *split* of P.

The issue with normalization is that a partial trapezoid representation may only have non-normalized extensions. To solve this, the algorithm first transforms the partial representation into a normalized one, then solves the partial representation extension problem for it, and then tries to revert the normalization steps. Conditions necessary and sufficient for these three steps to succeed are described by a 2-SAT formula.

The recognition algorithms for trapezoid graphs [264, 63] use the fact that every transitive orientation of $\overline{G}$ gives a trapezoid poset, so its choice has no influence on existence of a trapezoid representation. Similarly as in Fig. 2.22, this is not true for the partial representation extension problem of trapezoid graphs. In the second step, the question of extending partial trapezoid representations reduces to testing whether there exists a transitive orientation of $\overline{G}$ such that the corresponding trapezoid poset is extendible. This is solved by dynamic programming on the modular tree, where transitive orientations of each node are found by solving another 2-SAT formula.

2.7.6 Proper Circular-arc Graphs

In this section, we briefly discuss different types of orientations and their partial orientation extension problems, described in [20]. We start with a few definitions:

- An orientation is called *acyclic* if it contains no directed cycle.
- A *tournament* is an orientation of a complete graph.
- For an orientation, let $N^-(v)$ and $N^+(v)$ be the sets of out-neighbors and of in-neighbors of v, respectively. An orientation is a *local tournament* if for every vertex v, the sets $N^-(v)$ and $N^+(v)$ induce tournaments.
- Further, if $N^-(v)$ and $N^+(v)$ induce transitive tournaments, then the orientation is called a *locally transitive local tournament.*
- If a locally transitive local tournament is complete, it is called *locally transitive tournament.*
- An orientation is called an *in-tournament* if each $N^+(v)$ induces a tournament.

In 1982, the seminal paper of Skrien [321] linked, in a different language, existence of some of these types of orientations to several classes of intersection graphs. He proved the following characterizations:

- Proper interval graphs are precisely those graphs which can be oriented to an acyclic local tournament; see also [154].
- Connected proper circular-arc graphs are precisely those connected graphs which can be oriented to a local tournament which is precisely when it can be oriented to a locally transitive local tournament; see also [20, Theorem 2.2].
- Chordal graphs are precisely those graphs which can be oriented to an acyclic in-tournament.
- Trivially perfect graphs are precisely those graphs which can be oriented to a transitive in-tournament.

Results of Bang-Jensen et al. [20]. They study the partial orientation extension problems for the mentioned types of orientations and the implications for the partial representation extension problems of the related classes of intersection graphs. First, they prove that the partial representation extension problems for local tournaments and acyclic local tournaments can be solved in polynomial time. The latter result implies a polynomial-time algorithm for REPEXT(PROPER INT) and generalizes the result of [211], described in Section 2.3.

They show that the partial representation extension problem for proper circular-arc graphs can be reduced to the partial orientation extension problem for locally transitive local tournaments which is in general NP-complete. But the particular instances rising from partial representations of proper circular-arc graphs can be solved in polynomial time, so they prove the following:

Theorem 2.7.8 (Bang-Jensen et al. [20])**.** *The* RepExt(PROPER CIRCULAR-ARC) *problem can be solved in polynomial time.*

In conclusions, they discuss several other types of orientations and present open problems. In particular the complexity of the partial orientation extension problem for acyclic in-tournaments, i.e., for chordal graphs, remains open, and solving this problem might give a different perspective to the partial representation extension problem for chordal graphs.

Since the paper [20] heavily relies on the language and the techniques for oriented graphs, it is outside the scope of this thesis to present them in more detail and an interested reader may refer to [321, 20] and the references therein. We note that there are several different approaches for recognizing proper circular-arc graphs, see [253] for references. Can these other approaches be generalized to solve the partial representation extension problem as well. The orientation techniques do not apply to general circular-arc graphs, but it was observed by Zeman [363] that many difficulties are shared between proper circular-arc graphs and circular-arc graphs. So a better understanding of RepExt(PROPER CIRCULAR-ARC) might be fruitful in attacking the main open problem for partial representation extension discussed in Section 2.10: the complexity of RepExt(CIRCULAR-ARC).

Problem 2.7.9. *Is it possible to generalize some other techniques for recognizing proper circular-arc graphs to solve* RepExt(PROPER CIRCULAR-ARC) *in polynomial time?*

2.8 Extending Other Types of Partial Representations

The problem RepExt is defined for intersection representations only. In this section, we describe similar problems for other types of graph representations and we give overview of the known results.

Partial Embedding Extension. The similar problems to partial representation extension were studied even sooner for planar embeddings, where the problems are called *partial embedding extension.* Let G be a planar graph and let $\mathfrak{Rep}$ be a class of admissible planar embeddings. A *partial embedding* $\mathcal{R}'$ prescribes a planar embedding of some subgraph G' of G, so some vertices and edges of G are pre-drawn. A planar embedding $\mathcal{R} \in \mathfrak{Rep}$ *extends* $\mathcal{R}'$ if $\mathcal{R} \in \mathfrak{Rep}(G)$ and it embeds G' the same as $\mathcal{R}'$.

Problem:	Partial embedding extension – EmbedExt($\mathfrak{Rep}$)
Input:	A planar graph G and a partial embedding $\mathcal{R}' \in \mathfrak{Rep}$.
Question:	Is there a representation $\mathcal{R} \in \mathfrak{Rep}$ of G extending $\mathcal{R}'$?

As it turned out, the complexity of EmbedExt($\mathfrak{Rep}$) heavily depends on the class $\mathfrak{Rep}$ of admissible embeddings. Recall that CURVES denotes the class of all planar embeddings where edges are represented by arbitrary curves while STRAIGHTLINE denotes the class of Fary's straight-line embeddings where edges are represented by segments.

The first result for partial embedding extension was proved by Patrignani [292] in 2006. He proved that the problem EmbedExt(STRAIGHTLINE) is NP-hard by a nice geometric reduction from planar 3-Sat. See also [273].

On the other hand, Angelini et al. [4] show that EmbedExt(CURVES) can be solved in linear time. Their algorithm works essentially as follows. Recall the special role of 3-connected planar graphs from Section 1.4.1, namely Whitney's Theorem [359] stating that 3-connected planar graphs have unique embedding up to choice of an outer face. Therefore, partial embedding extension can be easily solved for 3-connected graphs. In Chapter 7, we describe that every graph can be decomposed into 3-connected components forming a tree, which is (for 2-connected graphs) mostly known under the name SPQR trees [95, 96, 97, 167] in the graph drawing community. To deal with general planar graphs, we first find a solution to the partial embedding extension problem on each 3-connected component. The difficult part is to join these partial solutions to extend the partial embedding of the whole graph, which is done from the bottom to the top of the decomposition tree by dynamic programming.

Kuratowski [245] and Wagner [351] characterized planar graphs as those graphs which do not contain K_5 and $K_{3,3}$ as minors. This result was extended to partially embedded planar graphs by Jelínek et al. [200], where a list of minimal forbidden partially embedded minors is constructed. Our characterization of minimal obstructions for partial representation extension of interval graphs, described in Chapter 4, has a similar spirit as this result.

Contact Representations of Planar Graphs. Recall contact representations of planar graphs from Section 1.4.2. Chaplick et al. [55] prove that the partial representation extension problems for these contact representations of planar graphs are NP-hard. The only tractable case is for grid intersection representations when all pre-drawn vertices belong to one part, i.e., either all are represented by horizontal segments, or all are represented by vertical ones.

Extending Visibility Representations. A *visibility representation* $\mathcal{R}$ of G is a collection of sets $\{\langle u\rangle : u \in \boldsymbol{V}(G)\}$ in the plane such that $uv \in \boldsymbol{E}(G)$ if and only if there exists a line of sight between $\langle u\rangle$ and $\langle v\rangle$ which is not obstructed by any other set $\langle w\rangle$. In the most classical setting, considered in [60], the sets $\langle u\rangle$ are horizontal segments in the plane, and such graphs are closely related to planar graphs. The complexity of extending partial visibility representations is studied in [60]. The problems are mostly NP-hard but a simple case solvable in polynomial time is also presented.

2.9 Related Restricted Representation Problems

In restricted representation problems, one asks whether there exists a representation satisfying some additional constraints. For the partial representation extension problems, the constraints are posed by partial representations. In this section, we describe

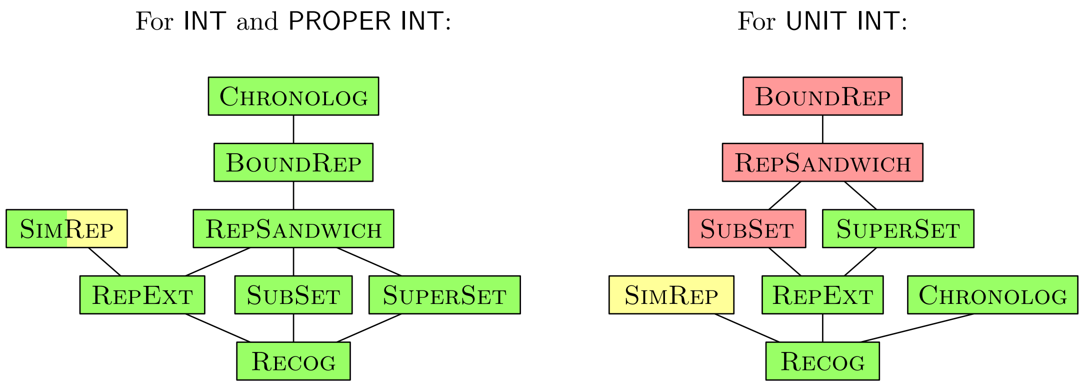


Figure 2.24: The Hasse diagrams of different restricted representation problems for interval graphs. If $\mathcal{P} \le \mathcal{P}'$, then the problem $\mathcal{P}$ can be solved using the problem $\mathcal{P}'$. The problems depicted in green can be solved in polynomial time, the red ones are NP-complete, and the complexity of SimRep for proper and unit interval graphs is open.

the main restricted representation problems which are related to RepExt. Figure 2.24 shows an overview of the complexity of the considered restricted representation problems for interval graphs.

We note that most restricted representation problems were considered for interval graphs or planar graphs, while some of them were also studied for other graph classes. For instance, the problems of construction of interval representations with prescribed lengths of intervals or prescribed lengths of intersections of intervals were studied in [293, 230]. Many other restricted representation problems are described in [323, 324].

2.9.1 Chronological Ordering

Skrien [322] introduced the following problem for interval graphs, motivated by applications in archaeology. The input gives a graph G and a partial ordering $<'$ of the left and right endpoints of the intervals. The task is to construct an interval representation $\mathcal{R}$ in which the left-to-right ordering of endpoints $<$ called a *chronological ordering* extends $<'$. We denote this problem Chronolog. Skrien [322] gives an algorithm solving Chronolog in time $\mathcal{O}(n^3)$.

This problem is not widely known which we believe should change. The paper [322] has over 25 citation in Google Scholar, but most of these cite a different result of Skrien's paper [322] which describes 2-LengthINT with the lengths 0 and 1. To the best of our knowledge, no other results concerning Chronolog are known. Skrien's cubic algorithm [322] is based on [322, Theorem 1] which describes necessary and sufficient conditions for existence of a chronological ordering extending $<'$ in terms of a transitive ordering of $\overline{G}$ (recall from Section 1.3.4 that INT $=$ co-COMP $\cap$ CHOR). This algorithm does not use PQ-trees (see Sections 3.1 and 4.2) or modular trees (see Section 2.7.1), and it is a natural question whether a faster algorithm could be constructed using these structural decompositions.

The problem Chronolog was not considered in the situation that some endpoint may share position, but Skrien's algorithm [322] can likely be modified. In such situations, Chronolog generalizes RepExt(INT) since a partial representation $\mathcal{R}'$ is fully described by its left-to-right ordering of endpoints $<'$.

Using structural results of Section 2.3, Chronolog can be easily solved for proper interval graphs in quadratic time. Notice that, unlike the partial representation extension problems discussed in Section 2.3, Chronolog for unit interval graphs is the same as Chronolog for proper interval graphs since only a left-to-right ordering of endpoints is restricted, not precise rational positions of endpoints. Figure 2.24 shows that Chronolog generalizes most considered restricted representation problems for interval graphs and proper interval graphs, but it generalizes only Recog for unit interval graphs, since geometric constraints for unit interval representations cannot be expressed.

2.9.2 Bounded Representation Problems

The bounded representation problems were first introduced by Klavík et al. [211, 212] in the context of unit interval graphs. For interval graphs, it is defined as follows. Let $\mathfrak{L}_v$ and $\mathfrak{R}_v$ be two intervals prescribed for each $v \in \boldsymbol{V}(G)$. An interval representation $\mathcal{R}$ of G is called a *bounded representation* if $\ell(v) \in \mathfrak{L}_v$ and $r(v) \in \mathfrak{R}_v$ for each $v \in \boldsymbol{V}(G)$. The bounded representation problem is the following decision problem:

Problem:	Bounded representation – BoundRep($\mathcal{C}, \mathfrak{Rep}$)
Input:	A graph $G \in \mathcal{C}$ and intervals $\mathfrak{L}_v$ and $\mathfrak{R}_v$ for each $v \in \boldsymbol{V}(G)$.
Question:	Is there a bounded representation $\mathcal{R} \in \mathfrak{Rep}(G)$?

Again, we just write BoundRep($\mathcal{C}$) when $\mathfrak{Rep}$ is clear from the context. The interval $\mathfrak{L}_v$ is the *left bound* of v and the interval $\mathfrak{R}_v$ is the *right bound* of v, or in both cases just simply a *bound* of v. Bounds are called *solvable* if there exists a bounded representation, and *unsolvable* otherwise. Figure 2.25 shows two examples of BoundRep instances. For unit interval graphs, since $r(v) = \ell(v)+1$, only $\mathfrak{L}_v$ needs to be specified.

Relation to Other Problems. Clearly, the bounded representation problems generalize recognition: if all bounds are set to $(-\infty, +\infty)$, they pose no restriction at all. They also generalize the partial representation extension problems by setting the bounds for pre-drawn vertices as singletons while the bounds for non-predrawn ones as $(-\infty, +\infty)$.

Again, assuming no shared endpoints, Chronolog for interval and proper interval graphs generalizes BoundRep(INT) and BoundRep(PROPER INT) as depicted in Fig. 2.24. The reason is that each collection of bounds gives an interval ordering $<'$ in which two bounds are comparable if and only if one is on the left of the other (see Section 1.3.1). The bounds are solvable if and only if there exists a chronological ordering extending $<'$. So the bounded representation problems are Chronolog for interval orders $<'$. Notice that Chronolog does not generalize BoundRep(UNIT INT).

Definitions for Other Classes. Instead of interval graphs, we may consider the bounded representation problems for several other classes of intersection graphs discussed in this chapter. For circle and circular-arc graphs (and their subclasses), the

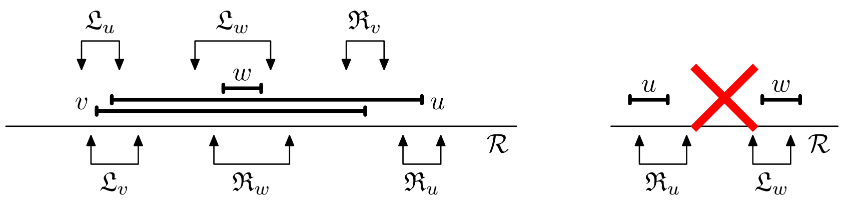


Figure 2.25: On the left, a bounded representation $\mathcal{R}$ of the class INT for the graph K_3. There exists no bounded proper interval representation since $\langle w\rangle$ is always a proper subset of $\langle u\rangle$ and $\langle v\rangle$. On the right, unsolvable bounds for the same graphs since $\langle u\rangle$ cannot intersect $\langle w\rangle$.

bounds $\mathfrak{L}_v$ and $\mathfrak{R}_v$ are two arcs of the circle, restricting the possible endpoints of chords/circular arcs. For permutation graphs, each $\mathfrak{L}_v$ is an interval on the bottom line while $\mathfrak{R}_v$ on the top line. Similarly for triangle and trapezoid graphs, each vertex might be given three or four intervals as bounds, respectively. These bounded representation problems generalize the corresponding partial representation extension problems, and the complexity is mostly open. On the other hand, a reasonable definition of bounded representations for chordal or function graphs is not clear.

Known Results. The results of [211, 212] described in Section 2.3 are mostly stated for bounded representations of unit interval graphs instead of partial representation extension. The original idea was that linear programming can deal with bounds instead of precise positions of endpoints for free. For a prescribed ordering $\blacktriangleleft$, the problem BoundRep(UNIT INT) can be solved in quadratic time [211, 212, 323, 324]. But when the ordering $\blacktriangleleft$ is not known, we get the following:

Theorem 2.9.1 (Klavík et al. [211, 212]). *The problem* BoundRep(UNIT INT) *is* NP*-complete.*

We also note that the bounds turned out to be essential for the left-shifting algorithm of [211, 212] in which the values $r(\mathfrak{L}_u)$ are relaxed to $+\infty$ while the positions $\ell(u)$ are simultaneously minimized.

Not much surprising, yet. As discussed in Section 2.3, it was already observed in [216] that the classes of proper and unit interval graphs behave differently with respect to the partial representation problem; see Fig. 2.8. In [324, 211, 212], the problem RepExt(UNIT INT) was solved in quadratic time by linear programming. So it seemed that this difference is only in some additional numerical problems posed by unit intervals. That is something expected because even recognition algorithms for unit interval graphs which construct unit interval representations, such as in [73], has to do some extra work to compute precise rational positions of endpoints.

This understanding was completely disproved by the paper of Balko et al. [18, 19] which studies the complexity of the bounded representation problems for interval and proper interval graphs. The following results are proved:

Theorem 2.9.2 (Balko et al. [18, 19]). *The both problems* BoundRep(INT) *and* BoundRep(PROPER INT) *can be solved in polynomial time.*

So the geometric version BoundRep(UNIT INT) is much harder than the topological version BoundRep(PROPER INT) dealing only with left-to-right orderings of endpoints.

The problem BoundRep(UNIT CIRCULAR-ARC) is NP-complete since already RepExt(UNIT CIRCULAR-ARC) is NP-complete [363]. Soulignac [323, 324] studies the restricted version in which the circular ordering of the arcs is prescribed, and proves that it can be solved in time $\mathcal{O}(n^2 + r)$ where r is the size of input; see 2.3 for more detail.

2.9.3 Representation Sandwich Problems

Let $\mathcal{C}$ be a class of graphs. Golumbic et al. [157] introduced the *graph sandwich problems* in which we are given two graphs G_1 and G_2 and we ask whether there exists a graph $G \in \mathcal{C}$ such that $G_1 \subseteq G \subseteq G_2$. In other words, some edges are forced in G, some edges are forbidden in G, and we ask whether the remaining edges can be chosen to get a graph in $\mathcal{C}$. For instance, for chordal, interval, permutation, or comparability graphs, the graph sandwich problems are NP-complete [157].

The paper [19] introduced the *representation sandwich problems*:

Problem: Representation Sandwich – RepSandwich($\mathcal{C}, \mathfrak{Rep}$)
Input: A graph $G \in \mathcal{C}$ and two sets A_v, B_v for each $v \in \boldsymbol{V}(G)$.
Question: Is there a representation $\mathcal{R} \in \mathfrak{Rep}$ such that $A_v \subseteq \langle v \rangle \subseteq B_v$ for each $v \in \boldsymbol{V}(G)$?

Also, we consider restricted versions SubSet($\mathcal{C}, \mathfrak{Rep}$), and SuperSet($\mathcal{C}, \mathfrak{Rep}$). Further for SubSet, we put all $A_v = \emptyset$, and for SuperSet, we put all $B_v = U$, where U is an universal set for all sets in representations of $\mathfrak{Rep}$. Again, we omit $\mathfrak{Rep}$ when clear from the context.

Known Results. It is easy to see that the graph sandwich problems can be reduced to the corresponding bounded representation problems if they exist. Therefore, they can be solved in polynomial time for interval and proper interval graphs. Also, RepSandwich always generalizes RepExt. See Fig. 2.24.

For unit interval graphs (and similarly for unit circular-arc graphs), we know that the length of $\langle v \rangle$ is always one. If B_v for SubSet(UNIT INT), respectively A_v for SuperSet(UNIT INT), already has the length one, then the unit interval $\langle v \rangle$ is fixed. Therefore, both SubSet(UNIT INT) and SuperSet(UNIT INT) generalize RepExt(UNIT INT) as depicted in Fig. 2.24. For SuperSet(UNIT INT), the ordering $\blacktriangleleft$ of connected components can be derived similarly as for RepExt(UNIT INT) [212, Theorem 5], so the problem can be solved in quadratic time using [211, 212, 323, 324]. On the other hand, the NP-completeness reduction of Theorem 2.9.1 can be modified for RepSandwich(UNIT INT) and SubSet(UNIT INT). Except for SimRep(UNIT INT), this answers the complexity of all problems in Fig. 2.24.

The problem of extending representations of functions graphs by partial functions of Theorem 2.7.6 is generalized by both SubSet(FUN) and SuperSet(FUN), so these problems are also NP-complete. The complexity of the representation sandwich problems remains open for other classes of intersection graphs.

2.9.4 Simultaneous Representations Problems

Let $\mathcal{C}$ be a class of graphs and let $\mathfrak{Rep}$ be a class of intersection representations. The *simultaneous representations problems*, introduced by Jampani et al. [197], is the following decision problem:

Problem: Simultaneous representations – $\textsc{SimRep}(\mathcal{C}, \mathfrak{Rep})$
Input: Graphs $G_1, \dots, G_k \in \mathcal{C}$ such that $\boldsymbol{V}(G_i) \cap \boldsymbol{V}(G_j) = I$ for all $i \neq j$.
Question: Do there exist representations $\mathcal{R}_1, \dots, \mathcal{R}_k \in \mathfrak{Rep}$ such that $\mathcal{R}_i = \big\{\langle u \rangle_i : u \in \boldsymbol{V}(G_i)\big\}$ represents G_i and for every $u \in I$ we have $\langle u \rangle_i = \langle u \rangle_j$ for all i, j.

As always, we omit $\mathfrak{Rep}$ when clear from the context. So representations $\mathcal{R}_1, \dots, \mathcal{R}_k$ are simultaneous when they represent the common vertices in I the same; see Figure 2.26. The requirement that $\boldsymbol{V}(G_i) \cap \boldsymbol{V}(G_j) = I$ is sometimes called *sunflower intersections* [33]. We may also allow arbitrary intersections between $\boldsymbol{V}(G_i)$ and $\boldsymbol{V}(G_j)$ and require for all $u \in \boldsymbol{V}(G_i) \cap \boldsymbol{V}(G_j)$ that $\langle u \rangle_i = \langle u \rangle_j$.

The paper [197] shows that $\textsc{SimRep}(\mathsf{PERM})$ and $\textsc{SimRep}(\mathsf{COMP})$ (for simultaneous transitive orientations) can be solved in polynomial time for any number of graphs, and $\textsc{SimRep}(\mathsf{CHOR})$ is polynomially solvable for $k = 2$ and NP-complete when k is a part of the input. For $\textsc{SimRep}(\mathsf{INT})$, the paper [198] gives for $k = 2$ an $\mathcal{O}(n^2 \log n)$ algorithm which Bläsius and Rutter [34] improve to $\mathcal{O}(n + m)$, and the complexity is open for other values k, even when k is a part of the input. The problem $\textsc{SimRep}(\mathsf{CIRCLE})$ is NP-complete when k is a part of the input [58, 59] and the complexity is open even for $k = 2$. The last reduction can be modified to show that the problems $\textsc{SimRep}(\mathsf{INT})$ and $\textsc{SimRep}(\mathsf{PROPER\ INT})$ are NP-complete when k is a part of the input and the intersections of graphs are arbitrary.

Reducing RepExt to SimRep. For some graph classes, the simultaneous representations problems are closely related to the partial representation extension problems. We sketch an easy reduction of $\textsc{RepExt}(\mathsf{INT})$ to $\textsc{SimRep}(\mathsf{INT})$ for $k = 2$ of Bläsius and Rutter, see [34, Section 4.1].

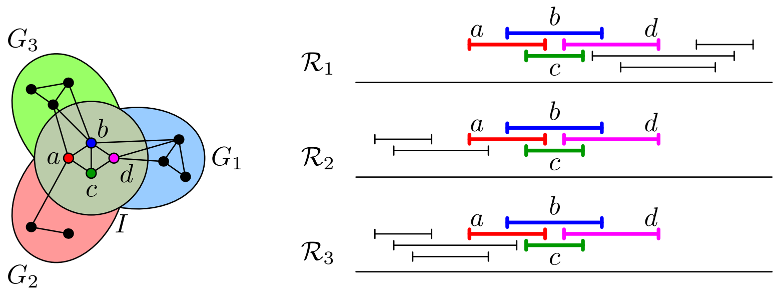


Figure 2.26: An example of three interval graphs with simultaneous representations assigning the same intervals to $I = \{a, b, c, d\}$.

Let G be a graph and let $\mathcal{R}'$ be a partial interval representation. We set $G_1 = G$ and $I = \boldsymbol{V}(G')$. Next, onto $\mathcal{R}'$, add a path going from left to right consisting of short intervals. This represents some interval graph G_2. The key part of the reduction is that G_2 has a unique representation up to reversal, so any representation of I has to be equivalent to $\mathcal{R}'$. Thus $\mathcal{R}'$ is extendible if and only if G_1 and G_2 can be simultaneously represented. The linear-time algorithm for RepExt(INT) of Bläsius and Rutter [34], mentioned in Section 2.2, is based on this reduction.

Such reductions work for some other graph classes including proper interval graphs, circular-arc graphs, circle graphs, and permutation graphs. A natural question is how useful are these reductions. Very little is known about the complexity of simultaneous representations problems, compared to the partial representation extension problems (see Fig. 2.2). If SimRep can be solved in polynomial time, we can usually solve RepExt by a much simple polynomial-time algorithm. Figure 2.24 shows that we are not aware of any reduction in which SimRep would generalize other restricted representation problems of Fig. 2.24 such as BoundRep, for which the techniques developed in solving RepExt may often be directly applied. Also, simpler algorithms for RepExt imply other results. For instance in Chapter 4, we describe minimal obstructions for partial representation extension of interval graphs, and a similar structural understanding of SimRep(INT), for $k = 2$, is nowhere close to be known.

There are also graph classes for which SimRep cannot be used to solve RepExt. For unit interval and unit circular-arc graphs, it is not possible to encode precise rational positions of pre-drawn intervals/arcs by another unit interval or unit circular-arc graph. Similarly, chordal graphs have many subtree-in-tree representations and a partial representation cannot be described in this way. In SimRep(CHOR) for $k = 2$, we are just given two graphs and we are completely free to build a tree in which both are simultaneously represented. But in RepExt(CHOR), a partial representation has to specify a part of the tree in which extending representation is constructed. Such a reduction is for chordal graphs not possible because SimRep(CHOR) for $k = 2$ is solvable in polynomial time [197], but RepExt(CHOR) is NP-complete in all possible settings [213, 214] (see Table 2.1).

Applying RepExt for Small Intersections. When I is small enough, we can often test all possible representations of the vertices in I. To solve the simultaneous representations problem, it remains to decide whether one of these representations of I can be completed to representations $\mathcal{R}_1, \dots, \mathcal{R}_k$ of $G_1, \dots, G_k$. If the corresponding partial representation extension problem can be solved in polynomial time, we can apply it and get an FPT algorithm for SimRep, parameterized by the size of I.

Let $n = \boldsymbol{v}(G_1) + \cdots \boldsymbol{v}(G_k)$, $m = \boldsymbol{e}(G_1) + \cdots + \boldsymbol{e}(G_k)$ and $\ell = |I|$. Such FPT algorithms can be constructed for the following graph classes:

- For proper interval graphs, in time $\mathcal{O}((n+m)(2\ell)!)$, using Theorem 2.3.1. The number of proper interval representations of I is bounded by $(2\ell!)$ which is the total number of left-to-right orderings of 2ℓ endpoints of ℓ intervals.
- For k-nested interval graphs, in time $\mathcal{O}(n^c(2\ell)!)$ for some constant c, using the result of [220].
- For interval graphs, in time $\mathcal{O}((n+m)(2\ell)!)$, using Theorem 2.2.3.

- For circle graphs, in time $\mathcal{O}(n^c(2\ell)!)$ for some constant c, using Theorem 2.6.1. We use interval overlap representations of Fig. 1.23.
- For permutation graphs, in time $\mathcal{O}(n^3(\ell!)^2)$, using Theorem 2.7.2. For ℓ segments in I, we test all possible orderings of ℓ endpoints on both lines.
- For trapezoid graphs, in time $\mathcal{O}(n^5((2\ell)!)^2)$, using Theorem 2.7.7. For ℓ trapezoids in I, we test all possible orderings of 2ℓ endpoints on both lines.
- For proper circular-arc graphs, in time $\mathcal{O}(n^c(2\ell)!)$ for some constant c, using Theorem 2.7.8.

It is unclear whether a similar result works for function graphs.

2.9.5 Allen Algebras and Interval Satisfiability

Allen algebras [3] play an important role in theory of artificial intelligence and time reasoning. See [159, 155] for surveys about time reasoning. We have several events represented by intervals in the timeline. For instance, consider the sentence

"During dinner, Peter reads the newspaper. Afterwards, he goes to bed."

We have three events: dinner, reading the newspaper, and going to bed. Time reasoning studies what information about the timeline can be derived.

Allen Algebra and ISat. Allowing shared endpoints, Allen [3] characterized thirteen *primitive relations* between the events, depicted in Fig. 2.27. A *relation* is a union of several primitive relations. For some pairs of events, we specify relations in which they can occur. For the above sentence, we have 'reading the newspaper' during/starting/finishing 'dinner' and 'dinner' before/meeting 'going to bed'.

In the *interval satisfiability problem*, called ISAT, we have prescribed relations for some pairs of events. We ask whether there exists an interval representation of the

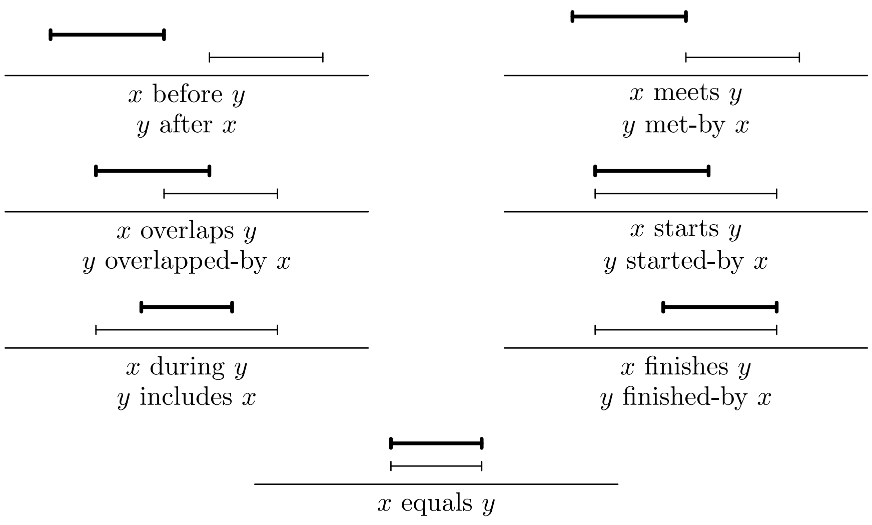


Figure 2.27: Thirteen primitive relations between the thick interval x and the thin interval y.

events such that all prescribed relations are satisfied. Vilian and Kautz [347] proved that ISat is NP-complete. Golumbic and Shamir [159] gave a more simple proof, using the interval graph sandwich problem [157]. Further, we can ask what additional relations can be infered from the input. For instance, for the sentence above, we can infer that 'reading the newspaper' before/meeting 'going to bed'.

We can restrict the ISat problem by allowing only some relations to be used—it gives $2^{8192} = 2^{2^{13}}$ different problems. Adding additional relations makes the problems only harder, and problems solvable in polynomial time imply the same for more restricted problems. Therefore, it suffices to identify maximal subsets of relations for which ISat is solvable in polynomial time. After twenty years of intense studies, a dichotomy was proved. The papers [286, 100] describe eighteen maximal subsets of relations for which ISat is polynomially solvable. Krokhin et al. [244] proved that the ISat problem is NP-complete for all subsets above them.

Restricted ISat. Golumbic and Shamir [159] considered the restricted version of ISat in which a relation is prescribed for every pair of events. Again, we can allow only some relations to be used, and more problems became solvable in polynomial time for restricted ISat. Further, recognition problems for several classes of intersection graphs are equivalent with these problems:

- The problem Recog(PROPER INT) is equivalent with the restricted ISat problem for 2 allowed relations:
  - *for non-edges:* before/after, and
  - *for edges:* overlaps/overlapped-by/meets/met-by/equals.
- The problem Recog(INT) is equivalent with the restricted ISat problem for 2 allowed relations:
  - *for non-edges:* before/after, and
  - *for edges:* the 11 remaining primitive relations.
- The problem Recog(CIRCLE) is equivalent with the restricted ISat problem for 2 allowed relations; recall Fig. 1.23:
  - *for non-edges:* before/after/during/includes, and
  - *for edges:* overlaps/overlapped-by

Also, recognition of interval orders and other problems can be described in this language; see [159].

Relation to RepExt. Similarly, the partial representation extension problems for proper interval graphs, interval graphs and circle graphs can be reduced to the restricted ISat problem for those two relations together with 13 primitive relations as singletons. For edges and non-edges incident with a non-predrawn vertices, we use the same relations. For pairs on pre-drawn vertices, we use the corresponding primitive relations. So these restricted ISat problems are a natural generalization of RepExt(PROPER INT), RepExt(INT), and RepExt(CIRCLE), but the complexity remains open.

2.10 Open Problems

We conclude this chapter with an overview of the main open problems for the partial representation extension problems. Further open problems are mentioned in Problems 2.5.1, 2.7.9, and 3.3.8 and Sections 2.9, 4.8, and 5.4.

Circular-arc Graphs. The main open problem for partial representation extension is the complexity of RepExt(CIRCULAR-ARC):

Problem 2.10.1. *Can the problem* RepExt(CIRCULAR-ARC) *be solved in polynomial time?*

This problem is very interesting for several reasons. All known polynomial-time recognition algorithms are quite complex and construct specific types of representations called *normalized representations* discussed in Section 2.7.5; see [195, 271, 203]. Can be a partial representation normalized similarly as in the case of trapezoid graphs in [243]. To solve RepExt(CIRCULAR-ARC), the structure of all representations needs to be better understood which could lead to a major breakthrough concerning this and other classes.

Consider the original recognition algorithm of Tucker [342]. It identifies two basic cases: a *biclique case* in which $\boldsymbol{V}(G)$ can be partitioned into two cliques, and a *multiclique case* in which more maximal cliques are needed. The latter case seems simpler to generalize for RepExt. So it is natural to concentrate on the biclique case, in which every circular-arc representation has two points on the circle (corresponding to the two cliques) such that every arc contains at least one of them. Tucker's algorithm [342] reduces the biclique case to recognition of a derived permutation graph. Can this reduction be modified to reduce RepExt(CIRCULAR-ARC) in the biclique case into RepExt(PERM) which can be solved in polynomial time? We note that in the case of proper circular-arc graphs, the same two basic cases occur and the biclique case is reduced to recognition of simpler bipartite permutation graphs [253]. At this moment, it is not clear whether the last reduction reduction can be modified for partial representation extension.

Problem 2.10.2. *Can the problem* RepExt(HELLY CIRCULAR-ARC) *be solved in polynomial time?*

Triangle Graphs. Recognition algorithms for this subclass of trapezoid graphs were described only recently [278, 336]. Can techniques in these algorithms be generalized for RepExt(TRIANGLE)? We believe that solving of the following open problem might give a better understanding of triangle graphs.

Problem 2.10.3. *Can the problem* RepExt(TRIANGLE) *be solved in polynomial time?*

String Graphs. Proving that recognition of string graphs is NP-complete was a long standing open problem [237, 314]. Can the techniques of [314] be applied to partial representation extension?

Problem 2.10.4. *Is the problem* RepExt(STRING) NP-*complete?*

Faster Algorithms. It is natural to ask whether the running time of the currently best known algorithms can be improved.

Problem 2.10.5. *Can the problem* RepExt(UNIT INT) *be solved in time $o(n^2 + r)$? Can it be solved in linear time $\mathcal{O}(n + r)$?*

Problem 2.10.6. *Can the problem* RepExt(PERM) *be solved in time $\mathcal{O}(n + m)$?*

Problem 2.10.7. *Can the problem* RepExt(TRAPEZOID) *be solved in time $o(n^5)$?*

3 Extending Partial Interval Representations in Linear Time

This chapter contains:

- *3.1: PQ-trees and Consecutive Orderings of Maximal Cliques.* Fulkerson and Gross [133] described that interval representations correspond to certain linear orderings of maximal cliques of an interval graph called consecutive orderings. Booth and Lueker [39] described a data structure called PQ-trees which efficiently stores all consecutive orderings.
- *3.2: Characterization of Extendible Partial Interval Representations.* We show that a partial representation $\mathcal{R}'$ gives a partial ordering $\lhd$ for maximal cliques such that it is extendible if and only if there exists a consecutive ordering of maximal cliques which extends $\lhd$.
- *3.3: The Reordering Problem of PQ-trees.* It asks whether a PQ-tree can be reordered to extend an input partial ordering. We describe two polynomial-time algorithms: one for general partial orderings and a faster one for interval orderings.
- *3.4: Linear-time Algorithm.* We construct a linear-time algorithm for RepExt(INT) by combining the above results.

`http://pavel.klavik.cz/orgpad/repext_int.html`

3.1 PQ-trees and Consecutive Orderings of Maximal Cliques

In this section, we review well-known properties of interval graphs. First, we describe the consecutive ordering problem and introduce a data structure to deal with these orderings called PQ-trees. Then, we describe characterization of interval graphs in terms of consecutive orderings of maximal cliques.

Consecutive Orderings. An input of the *consecutive ordering problem* consists of a set E of elements and restricting sets $S_1, S_2, \dots, S_k \subseteq E$. A linear ordering $<$ of E is called a *consecutive ordering* if every S_i appears *consecutively* in $<$: there are no $a < b < c$ such that $a, c \in S_i$ and $b \notin S_i$. The consecutive ordering problem asks whether there exists a consecutive ordering of E.

For an example, consider the elements $E = \{a, b, c, d, e, f, g, h\}$ and the restricting sets $S_1 = \{a, b, c\}$, $S_2 = \{d, e\}$, and $S_3 = \{e, f, g\}$. For instance, the orderings *abcdefgh* and *fgedhacb* are feasible. On the other hand, the orderings $\underline{a}\underline{c}defg\underline{b}h$ (violates S_1) and $d\underline{e}f\underline{h}\underline{g}abc$ (violates S_3) are not feasible.

PQ-trees. A PQ-tree is a tree structure invented by Booth and Lueker [39] for solving the consecutive ordering problem efficiently. Moreover, it stores all consecutive orderings for a given input.

The leaves of the tree are in one-to-one correspondence to the elements of E. The inner nodes are of two types: *P-nodes* and *Q-nodes* . The tree is rooted and an order of the children of every inner node is fixed. Also we assume that each P-node has at least two children and each Q-node has at least three children. A PQ-tree T represents one linear ordering $<_T$ called *frontier* , given by the ordering of the leaves from left to right, see Fig. 3.1.

Every PQ-tree T further represents other consecutive orderings. These orderings are frontiers of equivalent PQ-trees. A PQ-tree T' is *equivalent* to T if it can be constructed from T by *reordering* T which is an application of a sequence of *equivalent transformations* of two types: (i) an arbitrary reordering of the children of a P-node, and (ii) a reversal of the order of the children of a Q-node. For example, the PQ-trees in Fig. 3.1 are equivalent. All consecutive orderings are frontiers of an equivalence class of PQ-trees. For instance, the equivalence class in Fig. 3.1 corresponds to the input sets in the above example.

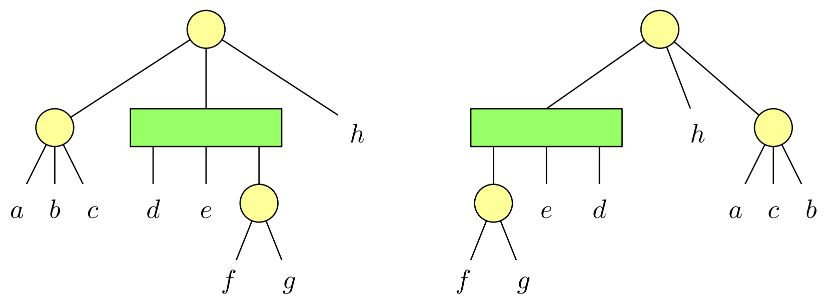


Figure 3.1: PQ-trees having frontiers *abcdefgh* and *fgedhacb*. In all figures, we denote P-nodes by circles and Q-nodes by rectangles.

For the purpose of this thesis, we only need to know the following:

Lemma 3.1.1 (Booth and Lueker [39]). *Let E be a set of elements and $S_1, \dots, S_k \subseteq E$.*

(a) There exists a unique equivalence class of PQ-trees such that their frontiers correspond to all consecutive orderings of E.

(b) A PQ-tree from this equivalence class can be constructed in time $\mathcal{O}(e+k+t)$ where e is the number of elements of E, k is the number of restricting sets and t is the sum of cardinalities of restricting sets. If no consecutive ordering exists, the equivalence class is empty and we detect it in the same running time.

Consecutive Orderings of Maximal Cliques. In Chapters 3, 4 and 5, maximal cliques are denoted by the letters a to f, and vertices by the remaining letters. Fulkerson and Gross [133] proved the following fundamental characterization of interval graphs in terms of maximal cliques; see Fig. 3.2 for an example.

Lemma 3.1.2 (Fulkerson and Gross [133]). *A graph is an interval graph if and only if there exists a linear ordering $<$ of its maximal cliques such that, for each vertex, the maximal cliques containing this vertex appear consecutively.*

Proof. Consider an interval representation of an interval graph. The consecutive ordering $<$ from the statement is obtained by sweeping this representation from left to right. By the Helly property, the intervals of every maximal clique have a non-empty intersection. For all maximal cliques, these intersections are disjoint and ordered from left to right. In the intersection of the intervals of a maximal clique a, we pick one point which we call a *clique-point* $\mathrm{cp}(a)$. The left-to-right ordering of these clique-points gives $<$. Every vertex appears in consecutive maximal cliques since it is represented by an interval in the representation.

On the other hand, given a consecutive ordering $<$, we place the clique-points from left to right according to $<$ and construct an interval representation by placing

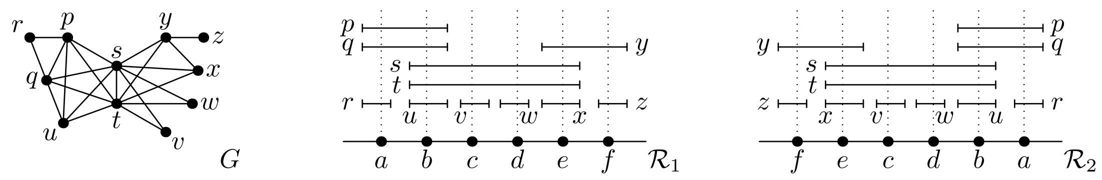


Figure 3.2: An interval graph G and two of its representations with different left-to-right orderings $<$ of the maximal cliques. Some choices of clique-points are depicted on the real lines.

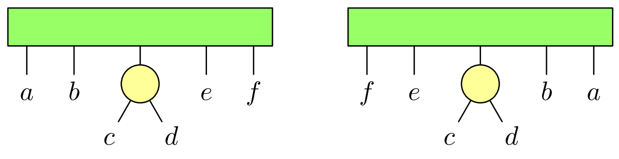


Figure 3.3: Two equivalent PQ-trees with frontiers $a < b < c < d < e < f$ and $f < e < c < d < b < a$, respectively.

each interval on top of its clique-points and no others. This can be done because the ordering places the maximal cliques containing each of the vertices consecutively. □

Two interval representations are considered the same, if one can be transformed into the other by continuously changing the intervals while preserving the correctness of a representation. We note that there is different consecutive orderings of maximal cliques correspond one-to-one to different interval representations of an interval graph.

The following simple lemma is useful later in proving Theorem 3.2.2:

Lemma 3.1.3. *Let G be an interval graph and let $S \subseteq \boldsymbol{V}(G)$ induce a connected subgraph of G. Then the maximal cliques of G containing at least one vertex of S appear consecutively in every consecutive ordering $<$ of the maximal cliques of G.*

Proof. Consider the interval representation given by $<$ with some choice of clique-points. The union $\langle S \rangle$ of the intervals of S is a closed interval, so it is connected. For every clique a, its clique-point $\mathrm{cp}(a)$ is placed on $\langle S \rangle$ if and only if a contains at least one vertex from S. Therefore the set of maximal cliques containing at least one vertex of S appears consecutively, with the remaining cliques on one side or the other. □

PQ-trees for Interval Graphs. By combining the above results, Booth and Lueker [39] constructed the first linear-time recognition algorithm for interval graphs:

Lemma 3.1.4 (Booth and Lueker [39])**.** *The problem* RECOG(INT) *can be solved in time $\mathcal{O}(n+m)$ where n is the number of vertices and m is the number of edges.*

Proof. Every chordal graph has at most $\mathcal{O}(n)$ maximal cliques of total size $\mathcal{O}(n+m)$ and they can be found in linear time [311]. Since every interval graph is chordal, this applies to them as well. If this subroutine fails, the input graph G is not an interval graph.

Otherwise, let E be the set all maximal cliques and for each $v \in \boldsymbol{V}(G)$, let $S_v = \{a \in E : v \in a\}$. By Lemma 3.1.1, we construct the corresponding PQ-tree in time $\mathcal{O}(n+m)$ if it exists. By Lemma 3.1.2, the PQ-tree exists if and only if G is an interval graph. □

In what follows, the elements E of PQ-trees always correspond to maximal cliques of an interval graph. The constructed PQ-tree describes all different interval representations which is essential for solving the partial representation extension problem.

3.2 Characterization of Extendible Partial Representations by Maximal Cliques

In this section, we derive a characterization of extendible partial representations of interval graphs. A partial representation $\mathcal{R}'$ gives a certain partial ordering $\lhd$. We show that a representation extending $\mathcal{R}'$ exists if and only if there exists a consecutive ordering of maximal cliques which extends $\lhd$.

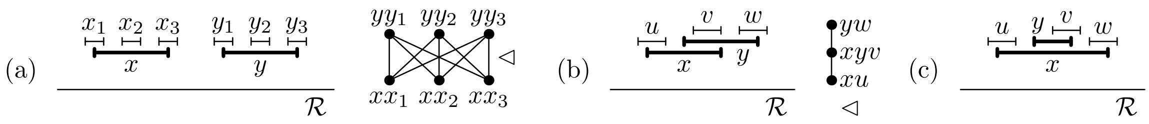


Figure 3.4: Possible relative positions of pre-drawn intervals $\langle x\rangle'$ and $\langle y\rangle'$, and some examples of the Hasse diagrams of the posed constraints.

(a) All maximal cliques containing x have to be on the left of those containing y.

(b) All maximal cliques containing x have to be on the left of those containing both x and y, which are on the left of those containing only y.

(c) An inclusion of pre-drawn intervals poses no constraints. A maximal clique containing only x can be either on the left, or on the right of the maximal cliques containing both x and y.

Restricting Clique-points. Suppose that there exists a representation $\mathcal{R}$ extending $\mathcal{R}'$. Then $\mathcal{R}$ gives some consecutive ordering $<$ of the maximal cliques from left to right. We want to show that the pre-drawn intervals give constraints in the form of a partial ordering $\lhd$, defined below. Figure 3.4 illustrates examples of constraints posed by a pair of pre-drawn intervals.

For a maximal clique a, let $P(a)$ denote the set of all pre-drawn intervals that are contained in a. For every extending representation $\mathcal{R}$, $P(a)$ restricts the possible position of the clique-point $\mathrm{cp}(a)$ to only those points x of the real line which are covered in $\mathcal{R}'$ by the pre-drawn intervals of $P(a)$ and no others. We denote the set of these admissible positions by $\downarrow_a$. Formally:

$$\downarrow_a = \Big\{x : x \in \mathbb{R} \text{ and } x \in \langle u\rangle' \iff u \in P(a)\Big\};$$

for examples see Fig. 3.5a. Equivalently, $\downarrow_a$ is defined in [18, 19] as

$$\downarrow_a = \Big(\bigcap_{u\in P(a)} \langle u\rangle'\Big) \setminus \Big(\bigcup_{v\notin P(a)} \langle v\rangle'\Big).$$

We are interested in the extremal points of $\downarrow_a$. By $\curvearrowleft(a)$ (resp. $\curvearrowright(a)$), we denote the infimum (resp. the supremum) of $\downarrow_a$. We use an open interval $I_a = (\curvearrowleft(a), \curvearrowright(a))$ to represent $\downarrow_a$. We note that this does not imply that $\downarrow_a$ contains all points between $\curvearrowleft(a)$ and $\curvearrowright(a)$; see $\downarrow_b$ in Fig. 3.5. Notice that when $P(a) = \emptyset$, then $I_a = \mathbb{R}$.

The Relation $\lhd$. For two distinct maximal cliques a and b, we write $a \lhd b$ if $\curvearrowright(a) \le \curvearrowleft(b)$, or in other words, if I_a is on the left of I_b. We put $a \lhd a$ when

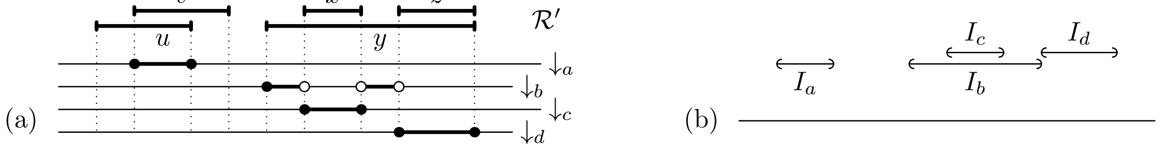


Figure 3.5: (a) Four maximal cliques a, b, c, and d with $P(a) = \{u, v\}$, $P(b) = \{y\}$, $P(c) = \{x, y\}$, and $P(d) = \{y, z\}$. The possible positions $\downarrow_a$, $\downarrow_b$, $\downarrow_c$, and $\downarrow_d$ of their clique-points are illustrated. (b) The corresponding open intervals I_a, I_b, I_c, and I_d.

$\downarrow_a = \emptyset$. The definition of $\lhd$ is quite natural, since $a \lhd b$ implies that every extending representation $\mathcal{R}$ has to place $\mathrm{cp}(a)$ to the left of $\mathrm{cp}(b)$. For instance, in Fig. 3.5, we get that $a \lhd b \lhd d$ and $a \lhd c \lhd d$, but b and c are incomparable.

All maximal cliques a with $\downarrow_a \neq \emptyset$ can be represented by open intervals I_a. This representation describes $\lhd$, since $a \lhd b$ if and only if I_a and I_b are disjoint and I_a is on the left of I_b. Recall from Section 1.3.1 that an ordering is called an interval ordering if it can be represented by closed intervals in the above manner: $a < b$ if and only if the interval of a is on the left of the interval of b. We get the following:

Lemma 3.2.1. *The relation $\lhd$ is an interval ordering if and only if no maximal clique a has $\downarrow_a = \emptyset$ and there are no two distinct maximal cliques a and b such that $\downarrow_a = \downarrow_b = \{x\}$.*

Proof. When $\downarrow_a = \emptyset$, then $a \lhd a$, so $\lhd$ is not an ordering. Similarly, when $\downarrow_a = \downarrow_b = \{x\}$, we have $a \lhd b \lhd a$. Otherwise, $\lhd$ can be represented by the closure of the intervals I_a, where we add small gaps between touching pairs of right and left endpoints that belong to distinct intervals. □

The reader might be wondering why we use open intervals I_a to represent maximal cliques of an interval graph represented by closed intervals. In every interval representation, its clique-points are strictly ordered from left to right, with no two sharing their positions. So even when $\curvearrowright(a) = \curvearrowleft(b)$, the clique-point $\mathrm{cp}(a)$ is always placed to the left of $\mathrm{cp}(b)$. Therefore it is natural to represent the interval orders $\lhd$ by open intervals.

Now, we are ready to prove the main structural theorem of this chapter:

Theorem 3.2.2. *A partial representation $\mathcal{R}'$ is extendible if and only if there exists a consecutive ordering of the maximal cliques that extends $\lhd$.*

Proof. A representation $\mathcal{R}$ extending $\mathcal{R}'$ gives some consecutive ordering of the maximal cliques. It is easy to observe that the constraints given by $\lhd$ are necessary, so this consecutive ordering has to extend $\lhd$. It remains to show the other implication.

To show the other implication, suppose that we have a consecutive ordering $<$ of the maximal cliques which extends $\lhd$. We construct a representation $\mathcal{R}$ extending $\mathcal{R}'$. We place the clique-points according to $<$ from left to right, in the following greedy manner. Suppose that we want to place a clique-point $\mathrm{cp}(a)$. Let $\mathrm{cp}(b)$ be the last placed clique-point. Consider all the points where the clique-point $\mathrm{cp}(a)$ can be placed and that are to the right of the clique-point $\mathrm{cp}(b)$. If there is a single such point, we place $\mathrm{cp}(a)$ there. Otherwise $\curvearrowleft(a) < \curvearrowright(a)$, and we take the infimum of all such points and place the clique-point $\mathrm{cp}(a)$ to its right by an appropriate epsilon, for example the distance of two closest distinct endpoints of pre-drawn intervals divided by n.

We prove by contradiction that this greedy procedure cannot fail; see Fig. 3.6. Let $\mathrm{cp}(a)$ be the clique-point for which the procedure fails. It is not possible that $\downarrow_a = \emptyset$, since in this case $a \lhd a$ and $<$ cannot extend $\lhd$. Since $\mathrm{cp}(a)$ cannot be placed, there are some clique-points placed on $\curvearrowright(a)$ or to its right. Let $\mathrm{cp}(b)$ be the leftmost among them. If $\curvearrowleft(b) \ge \curvearrowright(a)$, we obtain $a \lhd b$, which contradicts $b < a$ because $\mathrm{cp}(b)$

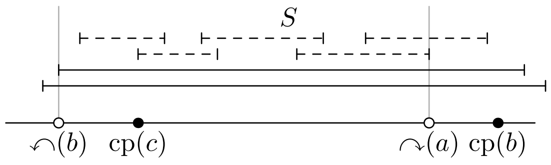


Figure 3.6: An illustration of the proof. The positions of the clique-points $\mathrm{cp}(b)$ and $\mathrm{cp}(c)$, the intervals of S are dashed.

was placed before $\mathrm{cp}(a)$. Thus, we know that $\curvearrowleft(b) < \curvearrowright(a)$. To get contradiction, we question why the clique-point $\mathrm{cp}(b)$ was not placed on the left of $\curvearrowright(a)$.

The clique-point $\mathrm{cp}(b)$ was not placed on the left of $\curvearrowright(a)$ because all these positions were either blocked by some other previously placed clique-points, or they are covered by some pre-drawn interval not in $P(b)$. There is at least one clique-point placed to the right of $\curvearrowleft(b)$, since otherwise we could place $\mathrm{cp}(b)$ at $\curvearrowleft(b)$ or right next to it. Let $\mathrm{cp}(c)$ be the right-most clique-point placed between $\curvearrowleft(b)$ and $\mathrm{cp}(b)$. Every point between $\mathrm{cp}(c)$ and $\curvearrowright(a)$ is covered by a pre-drawn interval not in $P(b)$. Consider the set S of all the pre-drawn intervals not contained in $P(b)$ intersecting $[\mathrm{cp}(c), \curvearrowright(a)]$ (see the dashed intervals in Fig. 3.6).

Let C be the set of all maximal cliques containing at least one vertex from S. Since S induces a connected subgraph, by Lemma 3.1.3, all maximal cliques of C appear consecutively in $<$. Now, a and c both belong to C, but b does not. Since $c < b$, we have $a < b$, which contradicts our original assumption $b < a$. □

No Single Overlaps. The following results play an important role in Chapter 4. We say that a pair of intervals I_a and I_b *single overlaps* if $I_a \neq I_b$ and either $\curvearrowleft(a) \le \curvearrowleft(b) < \curvearrowright(a) \le \curvearrowright(b)$, or $\curvearrowleft(b) \le \curvearrowleft(a) < \curvearrowright(b) \le \curvearrowright(a)$. Using Allen algebra described in Section 2.9.5, two intervals single overlap if they are in the relation overlaps/overlapped-by/starts/started-by/finishes/finished-by.

Lemma 3.2.3. *No pair of intervals I_a and I_b single overlaps.*

Proof. Assume without loss of generality that $\curvearrowleft(a) \le \curvearrowleft(b)$. If $\curvearrowright(a) \le \curvearrowleft(b)$, then I_a and I_b are disjoint and do not single overlap. Suppose now that $\curvearrowleft(a) \le \curvearrowleft(b) < \curvearrowright(a)$. Since all intervals of $P(a)$ cover $[\curvearrowleft(a), \curvearrowright(a)]$, we get $P(a) \subseteq P(b)$.

The position of $\curvearrowright(a)$ can be defined as a result of two distinct situations:

- If some pre-drawn interval of $P(a)$ ends in $\curvearrowright(a)$, then $\curvearrowright(b) \le \curvearrowright(a)$, since the same pre-drawn interval is contained in $P(b)$.
- Otherwise, there exists a sequence of pre-drawn intervals not contained in $P(a)$ that covers the whole portion between $\curvearrowright(a)$ and the leftmost right endpoint of the intervals of $P(a)$. The left endpoints of these intervals are on or to the right of $\curvearrowright(a)$. Since the left endpoints of the intervals in $P(b)$ are to the left of $\curvearrowright(a)$, the pre-drawn intervals of the sequence are not contained in $P(b)$. Thus, $\curvearrowright(b) \le \curvearrowright(a)$.

In both cases, $\curvearrowleft(a) \le \curvearrowleft(b) \le \curvearrowright(b) \le \curvearrowright(a)$, so I_b is contained in I_a. □

If no single overlaps are allowed, every pair of intervals is either disjoint, or one interval is contained in the other (possibly the intervals are equal). This type of interval orderings is very simple and has not been much studied. We note that graphs having interval representations with no single overlaps are called trivially perfect. By examining the above proof, we get the following useful result:

Lemma 3.2.4. *If $I_a \subseteq I_b$, then $P(a) \supseteq P(b)$. Further, strict containments correspond to strict inclusions.* □

If I_a and I_b are disjoint, then we only know that at least one of the sets $P(a) \setminus P(b)$ and $P(b) \setminus P(a)$ is non-empty. They both might be non-empty, or the sets $P(a)$ and $P(b)$ might be in inclusion. See Fig. 3.5 for examples.

3.3 The Reordering Problem of PQ-trees

Suppose that T is a PQ-tree and $\lhd$ is a partial ordering of its elements (leaves). We say that a reordering T' of the PQ-tree T is *compatible* with $\lhd$ if the ordering $<_{T'}$ extends $\lhd$, i.e., $a \lhd b$ implies $a <_{T'} b$. In this section, we deal with the following computational problem:

Problem:	The reordering problem – $\textsc{Reorder}(T, \lhd)$
Input:	A PQ-tree T and a partial ordering $\lhd$.
Question:	Is there a reordering T' of T compatible with $\lhd$?

3.3.1 The Reordering Problem for General Orderings

A *(rooted) subtree* of a PQ-tree T consists of a node and all its descendants. The subtrees of a node N are those subtrees having the children of N as the roots. For two subtrees T_i and T_j, we write $T_i \lhd T_j$ if and only if there exists $a \in T_i$ and $b \in T_j$ such that $a \lhd b$. For a node N, let $T[N]$ denote the subtree of T with the root N.

Local Solutions. A PQ-tree defines a hierarchical structure on its elements.

Observation 3.3.1. *Let S be a subtree of a PQ-tree T. Then the elements of E contained in S appear consecutively in $<_T$.*

We start with a lemma which states the following: If we can solve the reordering problem locally (inside of some subtree), then this local solution is always correct; either there exists no solution to the problem at all, or this local solution can be extended to a solution for the whole tree.

Lemma 3.3.2. *Let S be a subtree of a PQ-tree T. If T can be reordered compatibly with $\lhd$, then every local reordering of the subtree S compatible with $\lhd$ can be extended to a reordering of the whole tree T compatible with $\lhd$.*

Proof. Let T' be a reordering of the whole PQ-tree T compatible with $\lhd$. According to Observation 3.3.1, all elements contained in S appear consecutively in $<_{T'}$. Therefore, we can replace this local ordering of S by any other local ordering of S satisfying all constraints given by $\lhd$ on S. We obtain another reordering of the whole tree T which is compatible with $\lhd$ and extends the prescribed local ordering of S. □

The Reordering Algorithm. We describe the following algorithm:

Proposition 3.3.3. *The problem* Reorder$(T, \lhd)$ *can be solved in time* $\mathcal{O}(e + m)$*, where* e *is the number of elements and* m *is the number of comparable pairs in* $\lhd$.

Proof. The algorithm is the following greedy procedure. We represent the ordering $\lhd$ by a digraph having m edges. We reorder the nodes from the bottom to the root and modify the digraph by contractions. When we finish reordering a subtree, the order is fixed and never changed in the future; by Lemma 3.3.2, either this local reordering will be extendible, or there is no correct reordering of the whole tree at all. When we finish reordering a subtree, we contract the corresponding vertices in the digraph. We process a node of the PQ-tree when all its subtrees are already processed and the digraphs representing them are contracted to single vertices.

For a P-node, we check whether the subdigraph induced by the vertices corresponding to its children is acyclic. If it is acyclic, we reorder the children according to any topological sort of the subdigraph. Otherwise, there exists a cycle, no feasible ordering exists and the algorithm returns "no". For a Q-node, there are two possible orderings and we check whether one of them is feasible. For an example, see Fig. 3.7.

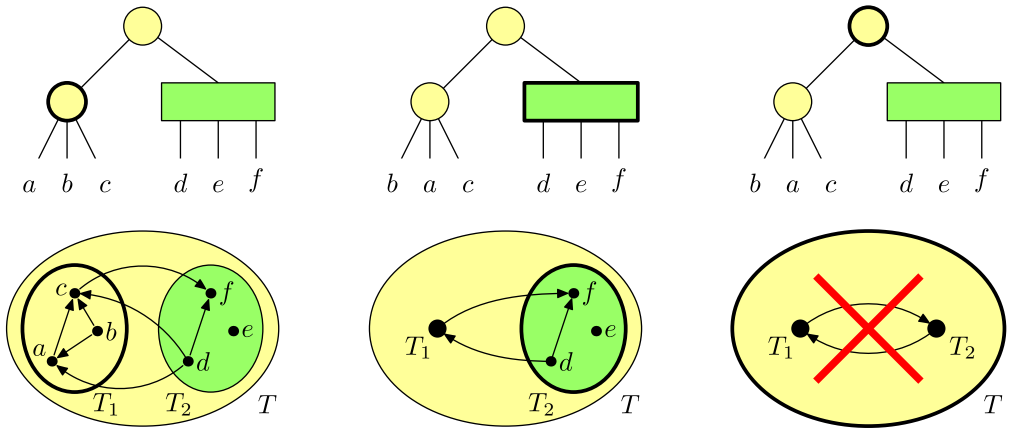


Figure 3.7: We show from left to right the way in which the reordering algorithm works. We depict comparable pairs of maximal cliques by directed edges. The processed trees are contracted into vertices.

First, we reorder the highlighted P-node on the left. The subdigraph induced by a, b and c is ordered $b \to a \to c$. We contract this subtree T_1 into a vertex. Next, we keep the order of the highlighted Q-node and contract its subtree T_2 into a vertex. When we reorder the root P-node, the algorithm finds a two-cycle between T_1 and T_2, and outputs "no".

Algorithm 1: Reordering a PQ-tree – REORDER$(T, \lhd)$

Require: A PQ-tree T and a partial ordering $\lhd$.
Ensure: A reordering T' of T such that $<_{T'}$ extends $\lhd$ if it exists.

```
 1: Construct the digraph of $\lhd$.
 2: Process the nodes of $T$ from the bottom to the root:
 3: for a processed node $N$ do
 4:    Consider the subdigraph induced by the children of $N$.
 5:    if the node $N$ is a P-node then
 6:       Find a topological sort of the subdigraph.
 7:       If it exists, reorder $N$ according to it, otherwise output “no”.
 8:    else if the node $N$ is a Q-node then
 9:       Test whether the current ordering or its reversal are compatible
          with the subdigraph.
10:       If at least one is compatible, reorder the node, otherwise output “no”.
11:    Contract the subdigraph into a single vertex.
12: return  A reordering $T'$ of $T$.
```

We need to argue the correctness. The algorithm processes the tree from the bottom to the top. For every subtree S, it finds some reordering of S compatible with $\lhd$. If no such reordering of S exists, the whole tree T cannot be reordered according to $\lhd$. If a reordering of S exists, it is correct according to Lemma 3.3.2. The algorithm runs in linear time with respect to the size of the PQ-tree and the partial ordering $\lhd$ which is $\mathcal{O}(e + m)$. Each edge of the digraph $\lhd$ is processed exactly once before it is contracted. □

We note that the described algorithm works even for a general relation $\lhd$. For example, $\lhd$ does not have to be transitive (as in the example in Fig. 3.7) or even acyclic (but in such a case, of course, no solution exists).

3.3.2 The Reordering Problem for Interval Orderings

In this section, we establish a faster algorithm for the reordering problem for a slight generalization of interval orders. Let E be the set of elements and let $\Big\{I_a = (\ell_a, r_a) : a \in E\Big\}$ be a collection of open intervals. For the purpose of Section 3.4, we also allow empty intervals with $\ell_v = r_v$.

These intervals represent the following relation $\lhd$ on E: we put $a \lhd b$ if and only if the intervals I_a and I_b do not intersect and I_a is on the left of I_b. If two empty intervals I_a and I_b share position (so $\ell_a = r_a = \ell_b = r_b$), we get $a \lhd b \lhd a$. If this is not the case, as argued in Section 3.2, the relation $\lhd$ is an interval ordering. See Fig. 3.8 for an example.

Faster Reordering of PQ-trees. Let e be the number of elements of E and let $\lhd$ be an interval order on E represented by $\{I_a : a \in E\}$. We assume the representation of $\lhd$ is sorted, which means that we know the order of all endpoints of the open

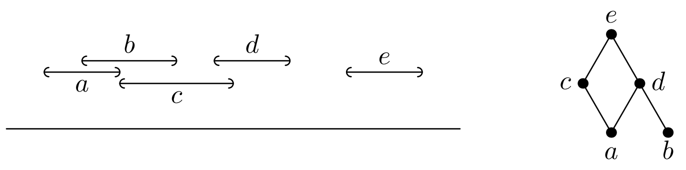


Figure 3.8: A collection of open intervals and the Hasse diagram of the interval order $\lhd$ represented by these intervals.

intervals from left to right. In the rest of this section, we show that we can solve $\textsc{Reorder}(T, \lhd)$ faster:

Proposition 3.3.4. *If $\lhd$ is an interval ordering given by a sorted representation, we can solve the problem $\textsc{Reorder}(T, \lhd)$ in time $\mathcal{O}(e)$ where e is the number of elements of T.*

For the following, let $\lessdot$ be the linear ordering of the endpoints ℓ_e and r_e of the intervals according to their appearance from left to right in the representation. To ensure that $a \lhd b$ if and only if $r_a \lessdot \ell_b$, we need to deal with endpoints sharing position. For them, we place in $\lessdot$ first the right endpoints (ordered arbitrarily) and then the left endpoints (again ordered arbitrarily). For a sorted representation, this ordering $\lessdot$ can be computed in time $\mathcal{O}(e)$. For example in Fig. 3.8 we get

$$\ell_a \lessdot \ell_b \lessdot r_a \lessdot \ell_c \lessdot r_b \lessdot \ell_d \lessdot r_c \lessdot r_d \lessdot \ell_e \lessdot r_e.$$

The general outline of the algorithm is exactly the same as before. We process the nodes of the PQ-tree from the bottom to the root and reorder them according to the local constraints. Using the interval representation of $\lhd$, we can implement all steps faster than before.

Informally speaking, the main trick is that we do not construct the digraph explicitly. Instead, we just work with sets of intervals corresponding to the elements of subtrees and compare them with respect to $\lhd$ fast. When we process a node with subtrees with the elements corresponding to sets $\mathcal{I}_1, \dots, \mathcal{I}_k \subseteq E$ which we already processed before. We test efficiently in time $\mathcal{O}(k)$ whether we can reorder these k subtrees according to $\lhd$. If it is not possible, then the algorithm stops and outputs "no". If the reordering succeeds, we put all the sets together $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \cdots \cup \mathcal{I}_k$, and proceed further. We now describe everything in details.

Comparing Subtrees. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be sets of intervals. We say $\mathcal{I}_1 \lhd \mathcal{I}_2$ if there exist $a \in \mathcal{I}_1$ and $b \in \mathcal{I}_2$ such that $a \lhd b$. We want to show that using the interval representation and some precomputation, we can decide whether $\mathcal{I}_1 \lhd \mathcal{I}_2$ in constant time. The following lemma states that we just need to compare the "leftmost" interval of $\mathcal{I}_1$ with the "rightmost" interval of $\mathcal{I}_2$; see Fig. 3.9.

Lemma 3.3.5. *Suppose that $a \lhd b$ for $a \in \mathcal{I}_1$ and $b \in \mathcal{I}_2$. Then for every $a' \in \mathcal{I}_1$ with $r_{a'} \lessdot r_a$ and every $b' \in \mathcal{I}_2$ with $\ell_b \lessdot \ell_{b'}$, it holds that $a' \lhd b'$.*

Proof. From the definition, $a \lhd b$ if and only if $r_a \lessdot \ell_b$. We have $r_{a'} \lessdot r_a \lessdot \ell_b \lessdot \ell_{b'}$, and thus $a' \lhd b'$. □

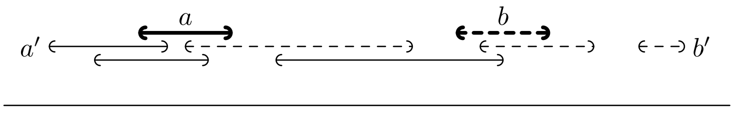


Figure 3.9: The normal intervals belong to $\mathcal{I}_1$ and the dashed intervals belong to $\mathcal{I}_2$. If $a \lhd b$, then also $a' \lhd b'$.

Using this lemma, we just need to compare a having the leftmost r_a to b having the rightmost ℓ_b since $\mathcal{I}_1 \lhd \mathcal{I}_2$ if and only if $a \lhd b$. To simplify the description, these special endpoints of intervals used for comparisons are called *handles*. More precisely, for a set of intervals $\mathcal{I}$, we define a *lower handle* and an *upper handle*:

$$\mathrm{LH}(\mathcal{I}) = \min\{r_x : x \in \mathcal{I}\} \qquad \text{and} \qquad \mathrm{UH}(\mathcal{I}) = \max\{\ell_x : x \in \mathcal{I}\}. \tag{3.1}$$

We note that $\mathrm{LH}(\mathcal{I}) \lessdot \mathrm{UH}(\mathcal{I})$ if $\mathcal{I}$ is not a clique. Using handles, we can compare sets of intervals fast. By Lemma 3.3.5, we have:

$$\mathcal{I}_1 \lhd \mathcal{I}_2 \qquad \text{if and only if} \qquad \mathrm{LH}(\mathcal{I}_1) \lessdot \mathrm{UH}(\mathcal{I}_2). \tag{3.2}$$

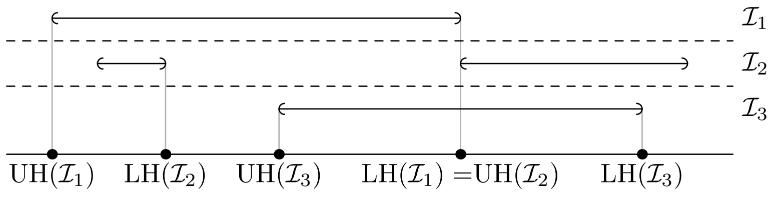
For an example, see Fig. 3.10.

Throughout the algorithm, we efficiently compute these handles for each processed subtree, and we do not need to remember which specific intervals are contained in the subtree. The handles serve in the same manner as the contraction operation of digraphs in the proof of Proposition 3.3.3.

Reordering Nodes. We describe fast reordering of the children of a processed node using the handles. Let $\mathcal{I}_1, \dots, \mathcal{I}_k$ be the sets of intervals corresponding to the subtrees of this node. Suppose that we know their handles and have them ordered according to $\lessdot$ as in Fig. 3.10. Let $\tilde{\lessdot}$ be the ordering $\lessdot$ restricted to the handles of $\mathcal{I}_1, \dots, \mathcal{I}_k$.

A linear ordering $<$ of the sets $\mathcal{I}_1, \dots, \mathcal{I}_k$ is called a *topological sort* if $\mathcal{I}_i \lhd \mathcal{I}_j$ implies $\mathcal{I}_i < \mathcal{I}_j$ for every $i \neq j$. The set $\mathcal{I}_j$ is *minimal* if there is no $\mathcal{I}_i$ such that $\mathcal{I}_i \lhd \mathcal{I}_j$. We use minimal elements to characterize all topological sorts. For every topological sort $1 < \dots < k$, the ℓ-th element restricted to $\{\ell, \ell{+}1, \dots, k\}$ is minimal. We describe this classical characterization in details since it is important for our algorithm.

Every topological sort can be constructed as follows. We repeatedly detect all minimal element $\mathcal{I}_i$ and always pick one. (For different choices we get different topological sorts). We stop when all elements are placed in the topological sort. If in some step no minimal element exists, then no topological sort exists.



$\mathcal{I}_1$
$\mathcal{I}_2$
$\mathcal{I}_3$
$\mathrm{UH}(\mathcal{I}_1)$
$\mathrm{LH}(\mathcal{I}_2)$
$\mathrm{UH}(\mathcal{I}_3)$
$\mathrm{LH}(\mathcal{I}_1) = \mathrm{UH}(\mathcal{I}_2)$
$\mathrm{LH}(\mathcal{I}_3)$

Figure 3.10: The handles for sets $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{I}_3$. We have $\mathrm{UH}(\mathcal{I}_1) \lessdot \mathrm{LH}(\mathcal{I}_2) \lessdot \mathrm{UH}(\mathcal{I}_3) \lessdot \mathrm{LH}(\mathcal{I}_1) \lessdot \mathrm{UH}(\mathcal{I}_2) \lessdot \mathrm{LH}(\mathcal{I}_3)$. According to (3.2), we get $\mathcal{I}_1 \lhd \mathcal{I}_2$, $\mathcal{I}_2 \lhd \mathcal{I}_3$, and $\mathcal{I}_1 \not\lhd \mathcal{I}_3$; so the relation $\lhd$ on sets of intervals is not necessarily transitive.

The following lemma describes minimal elements in terms of the ordering $\tilde{\lessdot}$:

Lemma 3.3.6. *The set $\mathcal{I}_j$ is minimal if and only if there is no lower handle* $\mathrm{LH}(\mathcal{I}_i)$ *for $i \neq j$ such that* $\mathrm{LH}(\mathcal{I}_i) \tilde{\lessdot} \mathrm{UH}(\mathcal{I}_j)$.

Proof. By (3.2), $\mathcal{I}_i \lhd \mathcal{I}_j$ if and only if $\mathrm{LH}(\mathcal{I}_i) \tilde{\lessdot} \mathrm{UH}(\mathcal{I}_j)$. If there is no such $\mathcal{I}_i$, then $\mathcal{I}_j$ is minimal. □

We can use this lemma to identify all minimal elements:

- If the ordering $\tilde{\lessdot}$ starts with two lower handles $\mathrm{LH}(\mathcal{I}_i)$ and $\mathrm{LH}(\mathcal{I}_j)$, there exists no minimal element. The reason is that all upper handles are larger, and so both $\mathcal{I}_i$ and $\mathcal{I}_j$ are smaller than everything else; specifically, we get $\mathcal{I}_i \lhd \mathcal{I}_j \lhd \mathcal{I}_i$.
- Otherwise if the first element of the ordering $\tilde{\lessdot}$ is $\mathrm{LH}(\mathcal{I}_i)$, then $\mathcal{I}_i$ is the unique candidate for a minimal element. We just need to check whether there is some other $\mathrm{LH}(\mathcal{I}_j)$ smaller than $\mathrm{UH}(\mathcal{I}_i)$, and if so, no minimal element exists.[1]
- If $\tilde{\lessdot}$ starts with a consecutive group of upper handles, we have several minimal elements. All $\mathcal{I}_i$'s of these upper handles are minimal elements. If the lower handle following the group of upper handles is $\mathrm{LH}(\mathcal{I}_j)$, then $\mathcal{I}_j$ is a candidate for a minimal element. As above, $\mathcal{I}_j$ is minimal if there is no other lower handle smaller than $\mathrm{UH}(\mathcal{I}_j)$.

When constructing a topological sort, we remove the handles of the picked minimal elements $\mathcal{I}_i$ from $\tilde{\lessdot}$ and append $\mathcal{I}_i$ to the sort.

Using the above, we can construct the following certifying subroutine for reordering of a node according to $\lhd$:

Lemma 3.3.7. *Suppose that we know $\tilde{\lessdot}$ for the handles of the sets $\mathcal{I}_1, \dots, \mathcal{I}_k$ of subtrees of $T_1, \dots, T_k$ of a node of the PQ-tree. We can decide in time $\mathcal{O}(k)$ whether $T_1, \dots, T_k$ can be reordered compatibly with $\lhd$, i.e, $T_i < T_j$ if $T_i \lhd T_j$.*

- *If the answer is "yes", we find a reordering of the node.*
- *If the answer is "no", for a P-node, we find two subtrees T_i and T_j such that $T_i \lhd T_j \lhd T_i$. And for a Q-node, four subtrees T_i, T_j, $T_{i'}$, and $T_{j'}$ (possibly not distinct) such that $i < j$, $i' < j'$, $T_i \rhd T_j$, and $T_{i'} \lhd T_{j'}$.*

Proof. For a P-node, we just need to find any topological sort by repeated removing of minimal elements in any way. As described, if no minimal element exists at some point, the ordering $\tilde{\lessdot}$ starts with two lower handles $\mathrm{LH}(\mathcal{I}_i)$ and $\mathrm{LH}(\mathcal{I}_j)$ such that $T_i \lhd T_j \lhd T_i$, so we output "no" with T_i and T_j as a certificate.

For a Q-node, we test whether the current ordering or its reversal are topological sorts. We iterate through each of the two prescribed orderings, check whether each element is a minimal element, and then remove its handles from $\tilde{\lessdot}$. In both cases, if we find a correct topological sort, we use it to reorder the children of the node. Otherwise,

[1]This can be done in constant time if we remember in each moment the positions of two leftmost lower handles in the ordering, and update this information after removing one of them from $\tilde{\lessdot}$.

for the current ordering, we find that $\mathcal{I}_i$ is not minimal because of $\mathcal{I}_j$ such that $i < j$ and $T_i \rhd T_j$, and similarly for the reversal, we find that $\mathcal{I}_{j'}$ is not minimal because of $\mathcal{I}_{i'}$ such that $i' < j'$ and $T_{i'} \lhd T_{j'}$. No reordering is possible and the algorithm outputs "no" together with T_i, T_j, $T_{i'}$ and $T_{j'}$ as a certificate.

The subroutine clearly runs in time $\mathcal{O}(k)$. □

The Reordering Algorithm. We are ready to show that our algorithm allows to find a reordering of the PQ-tree T compatible with an interval order $\lhd$ with a sorted representation in time $\mathcal{O}(e)$:

Proposition 3.3.4. We first deal with details of the implementation. We precompute the handles for every set of intervals corresponding to a subtree of an inner node of T. For each leaf, the handles are the endpoints. We process the tree from the bottom to the root. Suppose that we have an inner node corresponding to the set $\mathcal{I}$ of intervals and it has k children corresponding to $\mathcal{I}_1, \dots, \mathcal{I}_k$ for which we already know their handles. Then we calculate the handles of $\mathcal{I}$ using

$$\mathrm{LH}(\mathcal{I}) = \min\Big\{\mathrm{LH}(\mathcal{I}_i)\Big\} \qquad \text{and} \qquad \mathrm{UH}(\mathcal{I}) = \max\Big\{\mathrm{UH}(\mathcal{I}_i)\Big\}. \tag{3.3}$$

This can clearly be computed in time $\mathcal{O}(e)$, and we also note for each endpoint a list of nodes for which it is a handle. Using these list, we can sweep the sorted representation and compute all orderings $\widetilde{\lessdot}$ for all inner nodes of T, again in $\mathcal{O}(e)$ time.

Using Lemma 3.3.7, we test for each inner node of T with its ordering $\widetilde{\lessdot}$ whether its subtrees can be reordered according to $\lhd$. Correctness of the algorithm is proved the same as in Proposition 3.3.3, based on Lemmas 3.3.5 and 3.3.6.

Concerning the time complexity, we already discussed that we are able to compare sets of intervals using handles in constant time, by Lemma 3.3.5. The precomputation of all orderings $\widetilde{\lessdot}$ takes time $\mathcal{O}(e)$. We spend time $\mathcal{O}(k)$ in each node with k children. Thus the algorithm runs in linear time in the size of the tree, which is $\mathcal{O}(e)$. □

For a pseudocode, see Algorithm 2. We note that when the orderings $\widetilde{\lessdot}$ are constructed for all inner nodes, we do not need to process the tree from the bottom to the top. We can process them independently in parallel and a reordering T' of T exists if and only if we succeed in reordering every inner node.

We note that by Lemma 3.2.3, we know that interval orderings $\lhd$ rising from partial interval representations $\mathcal{R}'$ further have no single overlaps. It is an open problem whether this property can be used to further simplify the algorithm:

Problem 3.3.8. *Can the linear-time algorithm for* Reorder$(T, \lhd)$ *described in Proposition 3.3.4 be simplified when the representation of* $\lhd$ *contains no single overlaps?*

Algorithm 2: Reordering a PQ-tree, with an interval ordering – REORDER($T, \lhd$)

Require: A PQ-tree T and an interval ordering $\lhd$ with a sorted representation.
Ensure: A reordering T' of T such that $<_{T'}$ extends $\lhd$ if it exists.

```
1: Calculate the handles for each individual leaf of $T$ and initiate an empty list for each endpoint.
2: Process the nodes of $T$ from the bottom to the root:
3: for a processed node $N$ do
4:     Compute the handles of $N$ using (3.3).
5:     Add the node $N$ to the lists of the two endpoints which are the handles of $N$.
6: Iterate the sorted representation and construct all orderings $\tilde{\lessdot}$ for all inner nodes $N$.
7: Again process the nodes from the bottom to the root:
8: for a processed node $N$ with the ordering $\tilde{\lessdot}$ do
9:     if the node $N$ is a P-node then
10:        Find any topological sort by removing minimal elements from $\tilde{\lessdot}$.
11:        If it exists, reorder $N$ according to it, otherwise output “no”.
12:    else if the node $N$ is a Q-node then
13:        Test whether the current ordering or its reversal are topological sorts.
14:        Process the prescribed ordering from left to right, check for every element whether it is minimal and remove its handles from $\tilde{\lessdot}$.
15:        If at least one ordering is correct, reorder the node, otherwise output “no”.
16: return A reordering $T'$ of $T$.
```

3.4 Linear-time Algorithm

In this section, we describe an algorithm solving REPEXT(INT) in time $\mathcal{O}(n+m)$. We modify the algorithm of Lemma 3.1.4 to test the characterization of Theorem 3.2.2 using Proposition 3.3.4 as a subroutine.

Computed Sorted Representation. First, we show a sorted representation of the interval order $\lhd$, required by the assumptions of Proposition 3.3.4, can be computed in linear time:

Lemma 3.4.1. *For a sorted partial representation $\mathcal{R}'$, we can compute the sorted representation of $\lhd$ in time $\mathcal{O}(n+m)$.*

Proof. We sweep the real line from left to right and compute the sorted representation of $\lhd$. As stated above, the interval graph has $\mathcal{O}(n)$ maximal cliques containing in total $\mathcal{O}(n+m)$ vertices. We compute for every pre-drawn vertex the list of the maximal cliques containing it, and for every maximal clique a the number $|P(a)|$ of pre-drawn vertices it contains. We initiate an empty list W, a counter i of pre-drawn intervals covering the currently sweeped point.

When sweeping, there are two types of events. If we encounter a set of endpoints of pre-drawn intervals sharing a point, we first process the left endpoints, then we update $\curvearrowleft$ and $\curvearrowright$ for this point,[2] and then we process the right endpoints. If we sweep over a part, we just update $\curvearrowleft$ and $\curvearrowright$. In details:

- *If we encounter a left endpoint ℓ_u*, then we increase the counter i. For every clique a containing u, we increase its counter. If some clique a has all pre-drawn intervals placed over ℓ_u, we add a into the list W of watched cliques.
- *If we encounter a right endpoint r_u*, we decrease the counter of pre-drawn intervals. We ignore all maximal cliques containing u till the end of the procedure, and naturally we also remove them from W if there are any.
- *The update of $\curvearrowleft$ and $\curvearrowright$* is done for all cliques $a \in W$ such that $|P(a)| = i$. Notice that we currently sweep over exactly i pre-drawn intervals, and therefore we have to sweep over exactly the pre-drawn intervals of $P(a)$. We update $\curvearrowleft(a)$ to the current point or the infimum of the current part if it is not yet initialized. And we update $\curvearrowright(a)$ to the supremum.

In the end, we output the computed $\curvearrowleft$ and $\curvearrowright$ naturally sorted from left to right. If for some maximal clique a, the value $\curvearrowleft(a)$ was not initialized, the clique-point $\mathrm{cp}(a)$ cannot be placed and we output "no".

We argue that the procedure can be implemented in linear time. We have the cliques in W partitioned according to the number of predrawn intervals contained in them. When we sweep over i pre-drawn intervals, there is no $a \in W$ such that $|P(a)| > i$, and if there are $a, b \in W$ such that $|P(a)| = |P(b)| = i$, then necessarily $P(a) = P(b)$. But then $\curvearrowleft(a) = \curvearrowleft(b)$ and $\curvearrowright(a) = \curvearrowright(b)$, so we can ignore b for the rest of the sweep procedure and set the values $\curvearrowleft(b)$ and $\curvearrowright(b)$ according to the clique a in the end. Thus each update costs $\mathcal{O}(1)$. This implementation clearly runs in $\mathcal{O}(n + m)$. □

The Algorithm. Let T be the PQ-tree corresponding to the input interval graph and let $\lhd$ be the interval order obtained from the input partial representation $\mathcal{R}'$. The partial representation extension algorithm is based on the characterization of Theorem 3.2.2 which has the following algorithmic reformulation:

Corollary 3.4.2. *A partial representation $\mathcal{R}'$ is extendible if and only if the problem* Reorder$(T, \lhd)$ *admits a solution.* □

Below, we describe the first part of the linear-time certifying algorithm for RepExt(INT) of Theorem 2.2.3 which solves the problem and certifies "yes" answers. It proceeds in the following five steps, where the last two steps produce the certificate:

1. Using [311] and Lemma 3.1.1, we find all maximal cliques and construct a PQ-tree T representing all consecutive orderings of the maximal cliques.
2. We construct the sorted representation of the interval order $\lhd$ by Lemma 3.4.1.

[2]We also need to update here since it might happen that the interval $(\curvearrowleft(a), \curvearrowright(a))$ is empty for some maximal clique a. This can happen only if some pre-drawn interval of $P(a)$ is a singleton.

3. From Using Proposition 3.3.4, we test whether there is a reordering T' of the PQ-tree T compatible with $\lhd$. By Corollary 3.4.2, the partial representation $\mathcal{R}'$ is extendible if and only if there exists a solution to the reordering problem.
4. As in the proof of Theorem 3.2.2, we place the clique-points from left to right according to $<_{T'}$ on the real line, greedily as far to the left as possible.
5. Using these clique-points, construct a representation $\mathcal{R}$ extending $\mathcal{R}'$.

Step 1 is the original recognition algorithm of Lemma 3.1.4. In Step 2, we compute splitting of the real line into parts and construct a sorted representation of $\lhd$. In Step 3, we apply the algorithm of Proposition 3.3.4. Step 4 is the greedy procedure from the proof of Theorem 3.2.2. In Step 5, we construct intervals representing the vertices of $G \setminus G'$ as in Fig. 3.2; we construct each such interval on top of the corresponding clique-points. See Algorithm 3 for a pseudocode.

We are ready to prove the first part of Theorem 2.2.3.

Proposition 3.4.3. *If the partial representation $\mathcal{R}$ is given sorted from left to right, then the problem* RepExt(INT) *can be solved in time $\mathcal{O}(n+m)$, where n is the number of vertices and m is the number of edges. We certify the "yes" answers by finding a representation $\mathcal{R}$ extending $\mathcal{R}'$.*

Proof. Correctness of the algorithm is implied by Proposition 3.3.4 and Corollary 3.4.2. Correctness of the constructed certificate follows from the proof of Theorem 3.2.2.

Concerning the complexity, by [311], the total size of all maximal cliques is at most $\mathcal{O}(n+m)$ and the PQ-tree can be constructed in this time by Lemma 3.1.1. Using Lemma 3.4.1, we construct the sorted representation of $\lhd$ in time $\mathcal{O}(n+m)$. According to Proposition 3.3.4, the PQ-tree can be reordered according to $\lhd$ in time $\mathcal{O}(n+m)$.

Algorithm 3: Partial Representation Extension of Interval Graphs – RepExt(INT)

```
Require: An interval graph $G$ and a partial representation $\mathcal{R}'$.
Ensure: A representation $\mathcal{R}$ extending $\mathcal{R}'$ if it exists.
1: Compute maximal cliques and construct a PQ-tree.
2: Sweep $\mathcal{R}'$ from left to right and construct the sorted representation of $\lhd$.
3: Use Algorithm 2 to reorder the PQ-tree according $\lhd$.
4: If any of these steps fails, no representation exists and output "no".
5: Place the clique-points according to the ordering $<_{T'}$ from left to right:
6: for a clique-point cp($a$) placed after cp($b$) do
7:     Compute the infimum of all points of the real line on the right of cp($b$) where
       cp($a$) can be placed.
8:     If there is single such point, place cp($a$) there.
9:     Otherwise place cp($a$) by $\varepsilon$ on the right of the infimum, where $\varepsilon$ is the size of
       the smallest part divided by $n$.
10: Construct $\mathcal{R}$ for the remaining intervals on top of the placed clique-points.
11: return A representation $\mathcal{R}$ extending $\mathcal{R}'$.
```

Finally, the certificate which is a representation $\mathcal{R}$ extending $\mathcal{R}'$ is constructed in time $\mathcal{O}(n+m)$. $\square$

4 Minimal Obstructions for Partial Representation Extension of Interval Graphs

This chapter contains:

- *4.1: Definition of Minimal Obstructions.* We formally describe all minimal obstructions mentioned in Theorem 2.2.1 and prove that they are minimal and non-extendible.
- *4.2: MPQ-trees and Basic Tools.* We introduce MPQ-trees which are an augementation of PQ-trees described in Section 3.1. Also, we describe basic tools used in deriving minimal obstructions, namely Sliding Lemma.
- *4.3: Strategy for Finding Minimal Obstructions.* We describe our strategy for the proof that every non-extendible partial representation contains one of the described obstructions.
- *4.4, 4.5, and 4.6: Three Main Cases.* We divide the argument according to the type of obstructed node. The case of Q-nodes is most involved.
- *4.7: Proofs of Main Results.* We prove Theorem 2.2.1 and construct the linear-time certifying algorithm for RepExt(INT) of Theorem 2.2.3.
- *4.8: Conclusions.* We conclude with comments and several open problems.

http://pavel.klavik.cz/orgpad/minobstr.html

4.1 Definition of Minimal Obstructions

In this section, we formally define all twelve classes of minimal obstructions which make a partial interval representation non-extendible. Since the list is long, the description is quite technical. In the rest of the chapter, we prove that the list is complete.

Definition. Every *obstruction* consists of a graph H and a non-extendible partial representation $\mathcal{R}'_H$. This obstruction is *contained* in G and $\mathcal{R}'$ if the following holds:

(i) There exists an injective mapping $\varphi : \boldsymbol{V}(H) \to \boldsymbol{V}(G)$ such that $uv \in \boldsymbol{E}(H)$ if and only if $\varphi(u)\varphi(v) \in \boldsymbol{E}(G)$. So $\varphi(H)$ is an induced subgraph of G.
(ii) The pre-drawn vetices of H are mapped by φ to pre-drawn vertices of G.
(iii) The endpoints in $\mathcal{R}'_H$ have the same left-to-right order as the endpoints of the corresponding pre-drawn vertices in $\mathcal{R}'$.

For an example, see Fig. 4.1. If G and $\mathcal{R}'$ contain an obstruction, then $\mathcal{R}'$ is clearly non-extendible.

We give H by *descriptions* using finitely many vertices, edges, and induced paths. For inner vertices of the induced paths, we specify their adjacencies with the remainder of H. Since these induced paths do not have fixed lengths, each description having at least one induced path defines an infinite class of forbidden subgraphs H. Unlike LB obstructions, most classes of minimal obstructions need infinitely many different descriptions. For instance, each FAT obstruction has k induced paths, and different values of k need different descriptions.

By $P_{x,y}$ we denote an induced path from x to y; its length is the number of edges. If H contains an induced path $P_{x,y}$, and x and y are allowed to be adjacent, then $P_{x,y}$ can be a single edge. When $N[x] = N[y]$, we allow the length of $P_{x,y}$ to be zero, i.e., $x = y$.

Minimality. An obstruction is *minimal* if $\mathcal{R}'_H$ becomes extendible when any vertex of $\boldsymbol{V}(H)$ is removed or any pre-drawn interval of is made *free* by removing it from the partial representation $\mathcal{R}'_H$, while keeping the corresponding vertex in $\boldsymbol{V}(H)$.

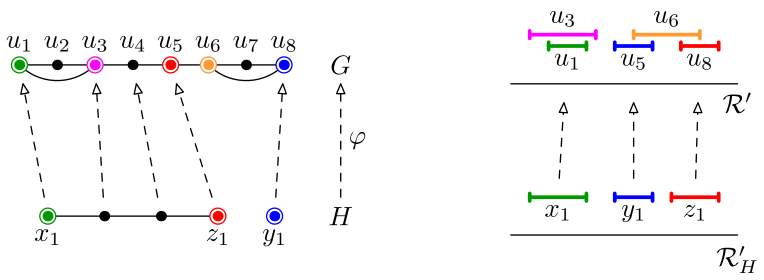


Figure 4.1: On top, an interval graph G with a non-extendible partial representation. This is certified by containing a 1-FAT obstruction H and $\mathcal{R}'_H$, depicted on bottom. Notice that the preimage of u_3 is not predrawn and nothing is mapped to u_6, which means that $\mathcal{R}'$ is non-extendible even when we free both $\langle u_3 \rangle'$ and $\langle u_6 \rangle'$.

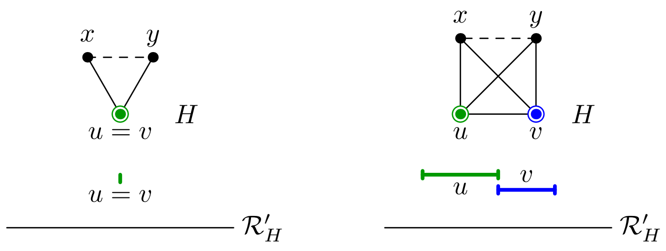


Figure 4.2: The SE obstructions, on the left with $u = v$, on the right with $u \neq v$.

4.1.1 List of Minimal Obstructions

In Fig. 2.6, we have already described minimal LB obstructions of [249] with $\mathcal{R}'_H = \emptyset$. There are eleven other classes of minimal obstructions we describe now. To understand most of this chapter, the reader should carefully read the definition of k-FAT obstructions (and possibly k-BI obstructions), while the definitions of the remaining classes can be checked when they appear.

SE obstructions. We start with two simple *shared endpoint obstructions* which deal with shared endpoints in $\mathcal{R}'$; see Fig. 4.2. We have two pre-drawn vertices u and v such that $r(u) = \ell(v)$ (possibly $u = v$, so only one interval may be pre-drawn). Further, there are two non-adjacent vertices x and y, both adjacent to u and v. These representations cannot be extended because $\langle x \rangle$ and $\langle y \rangle$ would need to pass through $r(u) = \ell(v)$, but in this case x and y would be adjacent. If $u \neq v$, the minimality requires that $\ell(u) < \ell(v) = r(u) < r(v)$.

***k*-FAT obstructions.** The class of *forced asteroidal triple obstructions* is defined inductively; the first two obstructions 1-FAT and 2-FAT are depicted in Fig. 2.7a and b, respectively.

A 1-FAT obstruction consists of three pre-drawn non-adjacent vertices x_1, y_1 and z_1 such that $\langle y_1 \rangle'$ is between $\langle x_1 \rangle'$ and $\langle z_1 \rangle'$. Further, x_1 and z_1 are connected by an induced path P_1 and y_1 is non-adjacent to the inner vertices of P_1. See Fig. 2.7a. These representations cannot be extended because, for at least one of the inner vertices u of P_1, $\langle u \rangle$ would intersect $\langle y_1 \rangle'$, but then u and y_1 would be adjacent.

A k-FAT obstruction (for $k > 1$) is defined as follows; see Fig. 4.3a. Let H_{k-1} be the graph of a $(k-1)$-FAT obstruction. To get H_k, we add to H_{k-1} two vertices x_k and t_k connected by an induced path P_k. Concerning edges, t_k is adjacent to all vertices of H_{k-1}, except for x_{k-1}. All vertices of H_{k-1} are non-adjacent to x_k and to the inner vertices of P_k. We put $y_k = z_{k-1}$ and $z_k = y_{k-1}$. A k-FAT obstruction has three pre-drawn vertices x_k, y_k and z_k such that $\langle y_k \rangle'$ is between $\langle x_k \rangle'$ and $\langle z_k \rangle'$. Finally, for $k > 1$, we allow P_1 to be a single edge, so x_1 can be adjacent to z_1.

We next see that these representations cannot be extended. Without loss of generality, we assume that $\langle x_k \rangle'$ is to the left of $\langle y_k \rangle'$, which in turn is to the left of $\langle z_k \rangle'$; see Fig. 4.3b. Since the inner vertices of P_k are not adjacent to any vertex of H_{k-1}, $\ell(t_k) < \ell(y_k)$and $r(t_k) > \ell(z_k)$. Since x_{k-1} is not adjacent to t_k, $\langle x_{k-1} \rangle$ is either to the right or to the left of $\langle t_k \rangle$. If it was to its left, then it would be to the left of $\langle x_k \rangle'$. Since the inner vertices of P_k are not adjacent to any vertex of H_{k-1}, we would

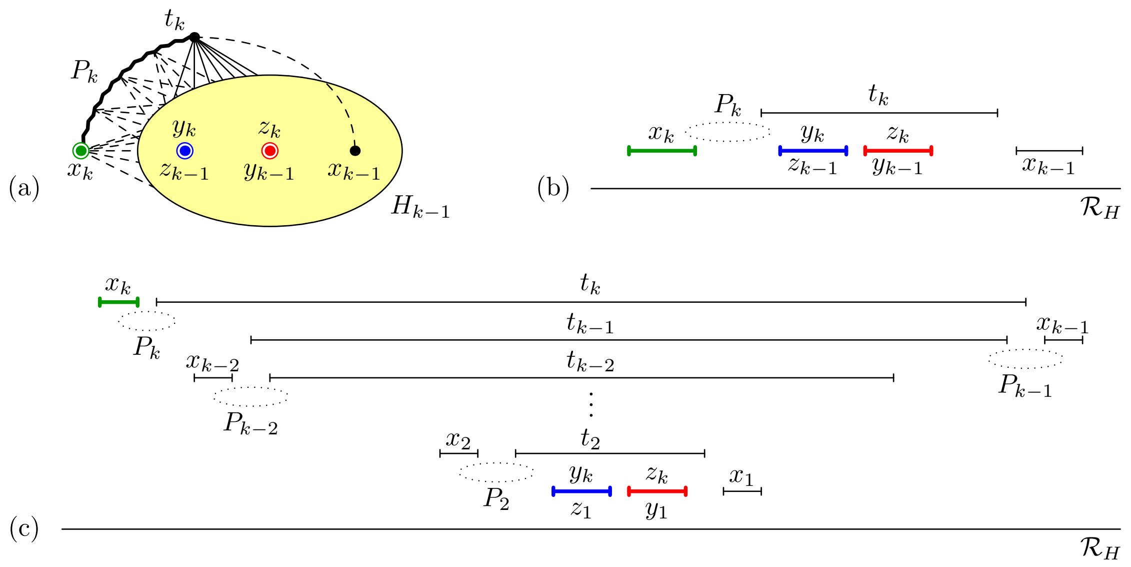


Figure 4.3: (a) A k-FAT obstruction is created from a $(k-1)$-FAT obstruction. It consists of the vertices $x_1, \dots, x_k$, $t_2, \dots, t_k$, y_k, z_k, and the induced paths $P_1, \dots, P_k$. The adjacencies are defined inductively.
(b) In every representation $\mathcal{R}_H$, the pre-drawn interval $\langle x_k \rangle'$ together with P_k and t_k forces $\langle x_{k-1} \rangle$ to be placed on the right of $\langle z_k \rangle'$. Therefore, the induced $(k-1)$-FAT obstruction is forced.
(c) The global zig-zag pattern forced by a k-FAT obstruction, with k nested levels going across $\langle y_k \rangle'$ and $\langle z_k \rangle'$. It is an obstruction since P_1 going from x_1 to z_1 with all inner vertices non-adjacent to y_1 cannot be placed.

obtain that H_{k-1} is disconnected, reaching a contradiction. Therefore, $\langle x_{k-1} \rangle$ is to the right of $\langle t_k \rangle$ and, in consequence, to the right of $\langle z_k \rangle'$. In general, $\langle x_{k-1} \rangle$ is forced to be placed on the other side of $\langle z_k \rangle' = \langle y_{k-1} \rangle'$ than $\langle y_k \rangle' = \langle z_{k-1} \rangle'$. Then, the $(k-1)$-FAT obstruction of H_{k-1} is forced. Since 1-FAT is a obstruction, we obtain that k-FAT, for $k > 1$, is also an obstruction. The global structure forced by a k-FAT obstruction is depicted in Fig. 4.3c. The main structural lemma of this chapter, k-FAT Lemma 4.6.3, explains in which situations these complicated k-FAT obstructions occur.

The remaining nine classes, depicted in Fig. 4.4, are derived from k-FAT obstructions. Let H_k denote the graph of a k-FAT obstruction. Except for the class of (k, ℓ)-CE obstructions, we create the graphs $\widetilde{H}_k$ of the remaining obstructions by adding a few vertices to H_k. These vertices are pre-drawn, and their positions force x_k, y_k and z_k to be represented as in their pre-drawn positions in a k-FAT obstruction. Since partial representations of k-FAT obstructions cannot be extended, partial representations of these derived classes cannot be extended either.

For $k = 1$, when one of x_1 and z_1 is not pre-drawn, we also allow x_1 to be adjacent to z_1. Additionally, it is possible that the added vertices already belong to H_k; for instance, a k-BI obstruction may have $u = t_k$ or $v = t_k$. Also, we do not specify in details the edges between the added vertices and $H_k \setminus \{x_k, y_k, z_k\}$. An accurate description would be too lengthy and the reader may derive it from Fig. 4.3c. We finally remark that we also consider all of the obstructions from the list horizontally flipped.

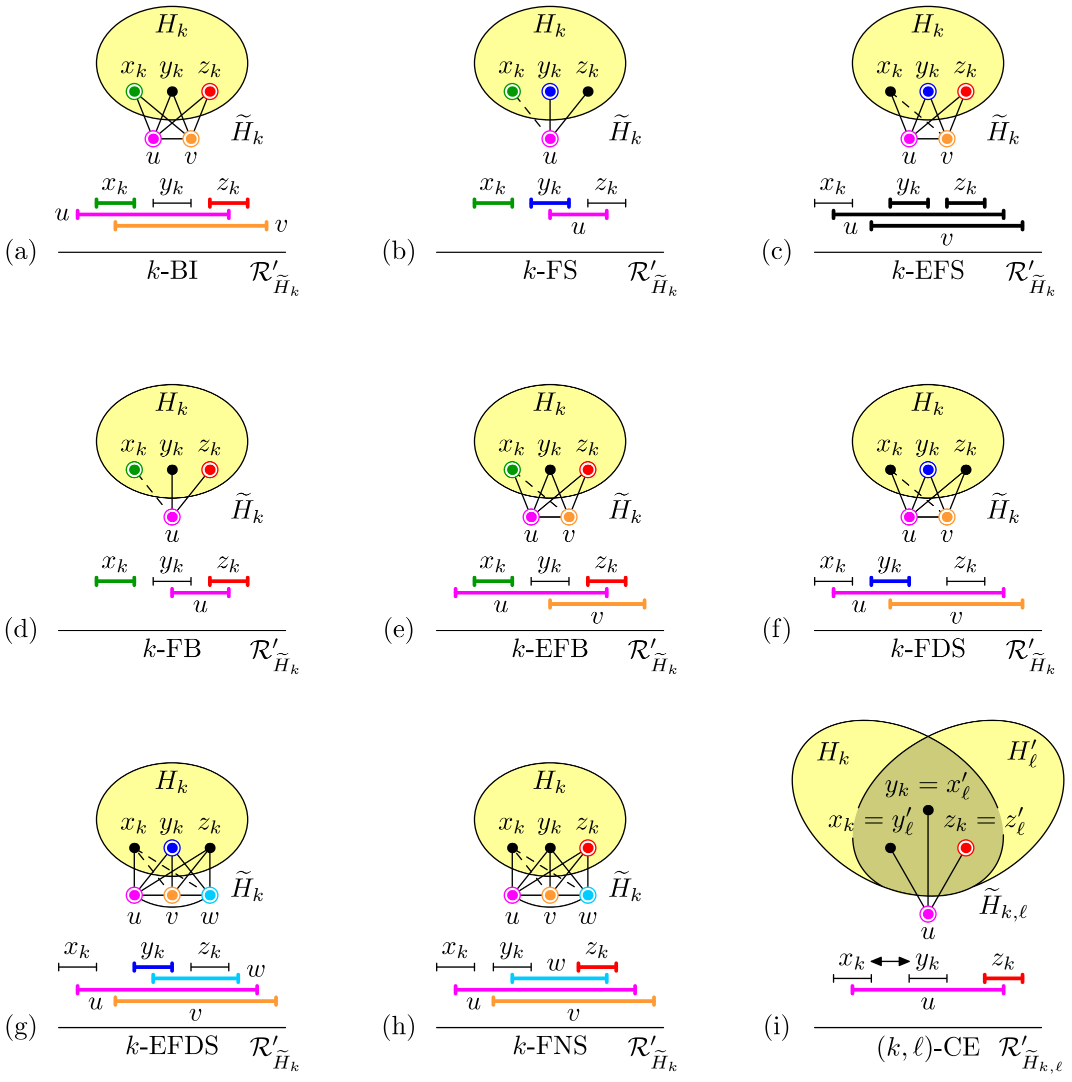

(a) k-BI $\mathcal{R}'_{\widetilde{H}_k}$
(b) k-FS $\mathcal{R}'_{\widetilde{H}_k}$
(c) k-EFS $\mathcal{R}'_{\widetilde{H}_k}$
(d) k-FB $\mathcal{R}'_{\widetilde{H}_k}$
(e) k-EFB $\mathcal{R}'_{\widetilde{H}_k}$
(f) k-FDS $\mathcal{R}'_{\widetilde{H}_k}$
(g) k-EFDS $\mathcal{R}'_{\widetilde{H}_k}$
(h) k-FNS $\mathcal{R}'_{\widetilde{H}_k}$
(i) (k,ℓ)-CE $\mathcal{R}'_{\widetilde{H}_{k,\ell}}$
H_k
H'_ℓ
x_k y_k z_k
u v w
$\widetilde{H}_k$
$\widetilde{H}_{k,\ell}$
$y_k = x'_\ell$
$x_k = y'_\ell$
$z_k = z'_\ell$

Figure 4.4: Nine classes of obstructions derived from k-FAT obstructions.

***k*-BI obstructions.** The class of *blocked intersection obstructions* is shown in Fig. 4.4a. To create $\widetilde{H}_k$ from H_k, we add two vertices u and v adjacent to x_k, y_k and z_k. Then the partial representation contains four pre-drawn vertices x_k, z_k, u and v. We have $\ell(u) \le \ell(v) < r(u) \le r(v)$, $\langle x_k \rangle'$ covering $\ell(v)$, and $\langle z_k \rangle'$ covering $r(u)$. We allow $u = v$.

The minimality further implies that $k \le 2$. The reason is that a k-BI obstruction with $k > 3$ contains a smaller $(k,1)$-CE obstruction by removing v and freeing $\langle x_k \rangle'$ (this follows from Lemma 4.6.4).

Concerning 1-BI, we allow $x_1 = z_1$. We illustrate all distinct obstructions only for this particular case ($k = 1$), so that the reader can understand the complexity of these classes; see Fig. 4.5.

For $k = 2$, we know by Lemma 4.6.4 that x_2 is adjacent to t_2. The pre-drawn

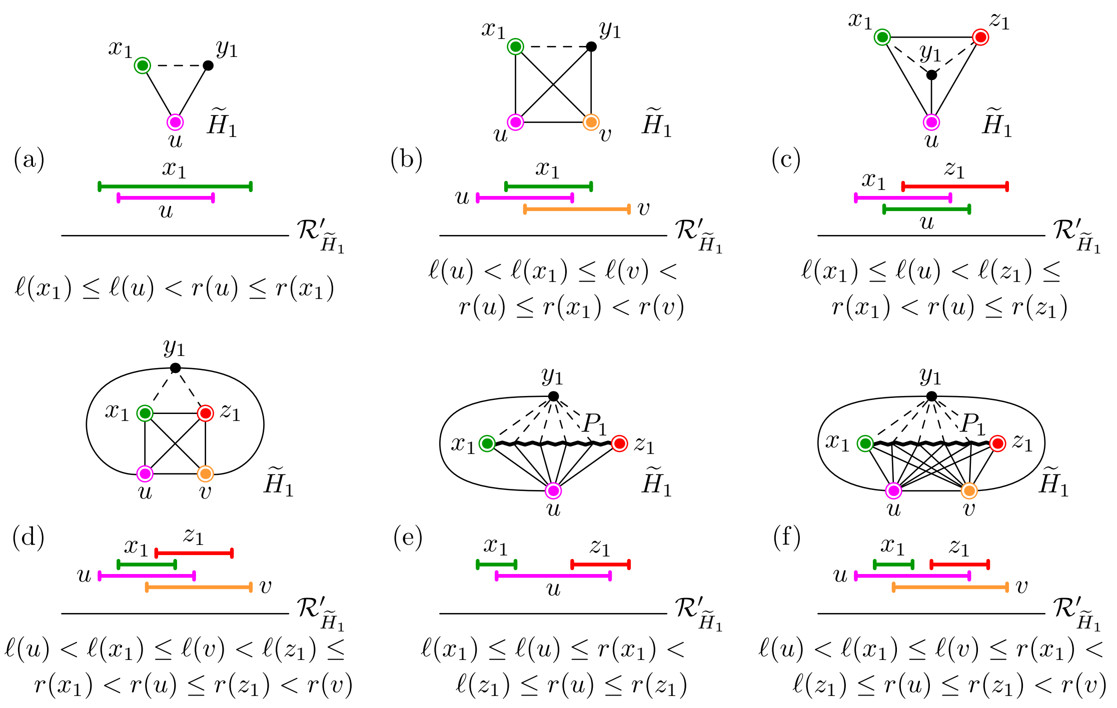

(a)
x_1 y_1 u $\widetilde{H}_1$
$\mathcal{R}'_{\widetilde{H}_1}$
$\ell(x_1) \le \ell(u) < r(u) \le r(x_1)$
(b)
x_1 y_1 u v $\widetilde{H}_1$
$\mathcal{R}'_{\widetilde{H}_1}$
$\ell(u) < \ell(x_1) \le \ell(v) < r(u) \le r(x_1) < r(v)$
(c)
x_1 z_1 y_1 u $\widetilde{H}_1$
$\mathcal{R}'_{\widetilde{H}_1}$
$\ell(x_1) \le \ell(u) < \ell(z_1) \le r(x_1) < r(u) \le r(z_1)$
(d)
y_1 x_1 z_1 u v $\widetilde{H}_1$
$\mathcal{R}'_{\widetilde{H}_1}$
$\ell(u) < \ell(x_1) \le \ell(v) < \ell(z_1) \le r(x_1) < r(u) \le r(z_1) < r(v)$
(e)
y_1 P_1 x_1 z_1 u $\widetilde{H}_1$
$\mathcal{R}'_{\widetilde{H}_1}$
$\ell(x_1) \le \ell(u) \le r(x_1) < \ell(z_1) \le r(u) \le r(z_1)$
(f)
y_1 P_1 x_1 z_1 u v $\widetilde{H}_1$
$\mathcal{R}'_{\widetilde{H}_1}$
$\ell(u) < \ell(x_1) \le \ell(v) \le r(x_1) < \ell(z_1) \le r(u) \le r(z_1) < r(v)$

Figure 4.5: All minimal 1-BI obstructions are depicted. Each gives several 1-BI obstructions with different $\mathcal{R}'_{\widetilde{H}_1}$, since there are several possible orderings of the endpoints satisfying the given inequalities. We have $x_1 = z_1$ for (a) and (b), and $x_1 \ne z_1$ otherwise. We have $u = v$ for (a), (c) and (e), and $u \ne v$ otherwise.

intervals are as follows:

$$\begin{aligned} \ell(x_2) \le \ell(u) \le r(x_2) < \ell(z_2) \le r(u) \le r(z_2), &\qquad \text{for } u = v,\\ \ell(u) < \ell(x_2) \le \ell(v) \le r(x_2) < \ell(z_2) \le r(u) \le r(z_2) < r(v), &\qquad \text{for } u \ne v. \end{aligned}$$

The position of $\langle u\rangle'$ and $\langle v\rangle'$ forces $\langle y_2\rangle$ to be placed between $\langle x_2\rangle'$ and $\langle y_2\rangle'$ in every extending representation, which forces a 2-FAT obstruction.

***k*-FS obstructions.** The class of *forced side obstructions* is shown in Fig. 4.4b. To create $\widetilde{H}_k$ from H_k, we add a vertex u adjacent to y_k and z_k. The partial representation contains three pre-drawn vertices x_k, y_k and u. We have $\ell(y_k) \le \ell(u) \le r(y_k) < r(u)$, and $\langle x_k\rangle'$ is on the left of $\langle y_k\rangle'$.

***k*-EFS obstructions.** The class of *extended forced side obstructions* is similar to that of k-FS obstructions; see Fig. 4.4c. To create $\widetilde{H}_k$ from H_k, we add u adjacent to x_k, y_k, and z_k, and v adjacent to y_k and z_k. The partial representation contains four vertices y_k, z_k, u and v pre-drawn as follows:

$$\ell(u) < \ell(v) < \ell(y_k) \le r(y_k) < \ell(z_k) \le r(z_k) < r(u) \le r(v).$$

***k*-FB obstructions.** The class of *forced betweenness obstructions* is similar to k-BI obstructions with $u = v$; see Fig. 4.4d. To create $\widetilde{H}_k$ from H_k, we add u adjacent to y_k and z_k. The partial representation contains three pre-drawn vertices x_k, z_k, and u. We have $\ell(u) < \ell(z_k) \le r(u) \le r(z_k)$, and $\langle x_k\rangle'$ is pre-drawn on the left of $\langle u\rangle'$.

***k*-EFB obstructions.** The class of *extended forced betweenness obstructions* is similar to those of k-BI and k-FB obstructions; see Fig. 4.4e. To create $\widetilde{H}_k$ from H_k, we add u adjacent to x_k, y_k and z_k, and v adjacent to y_k and z_k. The partial representation contains four pre-drawn vertices x_k, z_k, u, and v. We have

$$\ell(u) < \ell(x_k) \le r(x_k) < \ell(v) < \ell(z_k) \le r(u) \le r(z_k) < r(v).$$

***k*-FDS obstructions.** The class of *forced different sides obstructions* is shown in Fig. 4.4f. To create $\widetilde{H}_k$ from H_k, we add u adjacent to x_k, y_k, and z_k, and v adjacent to y_k and z_k. The partial representation contains three vertices y_k, u and v pre-drawn as follows:

$$\ell(u) < \ell(y_k) \le \ell(v) \le r(y_k) < r(u) \le r(v).$$

***k*-EFDS obstructions.** The class of *extended forced different sides obstructions* is similar to that of k-FDS obstructions; see Fig. 4.4g. To the construction of k-FDS, we further add w adjacent to y_k and z_k. The partial representation contains four vertices y_k, u, v and w pre-drawn as follows:

$$\ell(u) < \ell(v) < \ell(y_k) \le \ell(w) \le r(y_k) < r(w) < r(u) \le r(v).$$

***k*-FNS obstructions.** The class of *forced nested side obstructions* is constructed similarly to that of k-EFDS obstructions. In this case, z_k is pre-drawn instead of y_k; see Fig. 4.4h. In $\mathcal{R}'_{\widetilde{H}_k}$, we have

$$\ell(u) < \Big\{\ell(v), \ell(w)\Big\} \le \ell(z_k) \le r(w) < r(u) \le r(v),$$

where $\ell(v)$ and $\ell(w)$ are ordered arbitrarily.

(k, ℓ)-CE obstructions. The class of *covered endpoint obstructions* is created from a k-FAT obstruction glued to an ℓ-FAT obstruction; see Fig. 4.4i. To create $\widetilde{H}_{k,\ell}$, we glue H_k to H'_ℓ, with $k \ge \ell$. We put $z_k = z'_\ell$, $x_k = y'_\ell$ and $y_k = x'_\ell$, and some other vertices of these obstructions may also be shared. We add u adjacent to x_k, y_k, and z_k. The partial representation contains two pre-drawn intervals $\langle z_k \rangle'$ and $\langle u \rangle'$ such that $\ell(u) < \ell(z_k) \le r(u) \le r(z_k)$. These representations cannot be extended because, if $\langle x_k \rangle$ is placed to the left of $\langle y_k \rangle$, we get a k-FAT obstruction, while if $\langle x_k \rangle$ is placed to the right of $\langle y_k \rangle$, we get an ℓ-FAT obstruction.

By Lemma 4.6.4, for $k > 2$, if we swap the positions of $\langle x_k \rangle'$ and $\langle y_k \rangle'$ in a k-FAT obstruction, we obtain a 1-FAT obstruction. For $k > 2$, this implies that a minimal (k, ℓ)-CE obstruction is a $(k, 1)$-CE obstruction and it is formed by the graph H_k of a k-FAT obstruction together with an added vertex u. (By a careful analysis, either u adjacent to all vertices of H_k, or $u = t_k$.) Hence, the only (k, ℓ)-CE obstructions that are minimal are $(k, 1)$-CE obstructions and $(2, 2)$-CE obstructions; in other words, either $\ell = 1$, or $k = \ell = 2$. The remaining (k, ℓ)-CE obstructions, where $2 \ge k \ge \ell$, are analysed in Lemma 4.6.6 and depicted in Fig. 4.6.

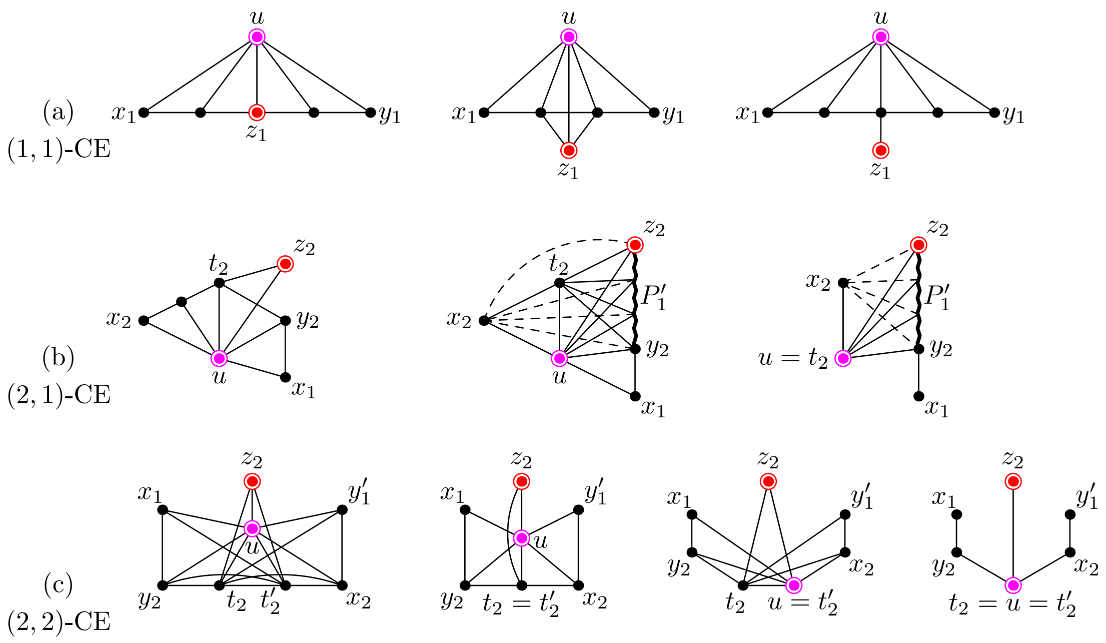


Figure 4.6: In this figure, for understandability, we omit most non-edges.
(a) There are three minimal $(1, 1)$-CE obstructions, and they are all finite graphs. The reason is that if, say, a path P_1 was very long, we could replace x_1 by an inner vertex of P_1; so such an obstruction would not be minimal.
(b) There are three minimal $(2, 1)$-CE obstructions. Due to the minimality, P_2 is a path of length at most two. When x_2 and t_2 are non-adjacent, then $P'_1 = y_2t_2z_2$ is a path avoiding $N[x_2]$. When x_2 and t_2 are adjacent, there are two cases, namely, $u \neq t_2$, and $u = t_2$. The path P'_1 is of length at least two.
(c) There are four minimal $(2, 2)$-CE obstructions, and they are all finite graphs. Necessarily x_2t_2 and $y_2t'_2$ are edges, and we can choose x_2 in such a way that it is adjacent to y'_1 (similarly, y_2 is adjacent to x_1). There are four different graphs, because the vertices u, t_2 and t'_2 might be distinct or not.

4.1.2 Proofs of Non-extendibility and Minimality

We sketch proofs that the defined obstructions are non-extendible and minimal. This implies the first part of Theorem 2.2.1. We establish the harder implication in Sections 4.3, 4.4, 4.5, 4.6, and 4.7.

Lemma 4.1.1. *Every k-FAT obstruction is non-extendible and minimal.*

Proof. We prove the claim by induction. For $k = 1$, non-extendibility and minimality are clear. For $k > 1$, assume that $\langle x_k\rangle'$ is on the left of $\langle y_k\rangle'$, and $\langle y_k\rangle'$ is on the left of $\langle z_k\rangle'$. In every representation of k-FAT, $\langle t_k\rangle$ covers $[\ell(y_k), \ell(z_k)]$. We know that $\langle x_{k-1}\rangle$ cannot be on the left of $\langle x_k\rangle'$, since H_{k-1} is connected and x_k is non-adjacent to all vertices of H_{k-1}. Therefore $\langle x_{k-1}\rangle$ has to be placed on the right of $\langle z_k\rangle'$. We get a $(k-1)$-FAT obstruction, which is non-extendible by the induction hypothesis.

It remains to argue the minimality. If one of $\langle x_k\rangle'$, $\langle y_k\rangle'$ and $\langle z_k\rangle'$ is made free, we can place them in such a way that $\langle z_k\rangle$ is between $\langle x_k\rangle$ and $\langle y_k\rangle$. This makes the partial representation extendible: It works for $k = 1$, and for $k > 1$, we can place x_{k-1}

on the right of $\langle y_k \rangle$, which makes the induced H_{k-1} extendible. If we remove one of the vertices or induced paths, the argument is similar. $\square$

Lemma 4.1.2. *The following obstructions are non-extendible and minimal:*

- *SE, k-FS, k-EFS, k-FB, k-EFB, k-FDS, k-EFDS, and k-FNS obstructions,*
- *k-BI obstructions for $k \leq 2$,*
- *(k,ℓ)-CE obstructions where either $\ell = 1$, or $k = \ell = 2$.*

Proof. For SE obstructions, the proof is trivial. For the remaining classes (aside k-BI and (k,ℓ)-CE), we proceed as follows. Non-extendibility follows from the fact that, in all cases, $\langle y_k \rangle$ is forced to be placed between $\langle x_k \rangle$ and $\langle z_k \rangle$. To show minimality, we use the minimality of k-FAT obstructions. Then it is easy to show that freeing any added pre-drawn interval or removing any added vertex results in the possibility of placing $\langle z_k \rangle$ between $\langle x_k \rangle$ and $\langle y_k \rangle$.

Consider a k-BI where $k \leq 2$. Non-extendibility follows from the fact that $\langle y_k \rangle$ has to be placed between $\langle x_k \rangle'$ and $\langle z_k \rangle'$, thus forcing the k-FAT obstruction, which is non-extendible by Lemma 4.1.1. By removing a vertex or an induced path of H_k, it becomes extendible as argued in Lemma 4.1.1. By freeing $\langle u \rangle'$ or $\langle z_k \rangle'$, we can place $\langle y_k \rangle$ on the right of $\langle z_k \rangle'$ which makes the partial representation extendible. By freeing $\langle v \rangle'$ or $\langle x_k \rangle'$, we can place $\langle y_k \rangle$ on the left of $\langle x_k \rangle'$, which also makes it extendible because $k \leq 2$ and x_2 is adjacent to t_2.

For (k,ℓ)-CE, either $\langle y_k \rangle$ is between $\langle x_k \rangle$ and $\langle z_k \rangle'$ (non-extendible due to the k-FAT obstruction), or $\langle y_\ell \rangle$ is between $\langle x_\ell \rangle$ and $\langle z_\ell \rangle'$ (non-extendible due to the ℓ-FAT obstruction). Minimality is also easy: Removing or freeing u allows to place $\langle z_k \rangle'$ between $\langle x_k \rangle$ and $\langle y_k \rangle$, which is extendible. And removing anything from one of the FAT obstructions allows one of the orderings of $\langle x_k \rangle$ and $\langle y_k \rangle$ to be extendible. $\square$

We note that the list of minimal obstructions is unique. Indeed, every minimal obstruction itself corresponds to a valid input, which cannot be obstructed by a distinct obstruction due to the minimality. Therefore, it is not possible to construct a smaller list of minimal obstructions, or to argue that if the partial representation contains a particular obstruction, then it also contains an additional one.

4.2 MPQ-trees and Basic Tools

In the first part of this section, we describe a generalization of PQ-trees, called MPQ-trees, by Korte and Möhring [235]. We also prove some simple structural results concerning MPQ-trees. In the second part, we describe several tools used in finding minimal obstructions, namely Sliding Lemma for partial interval representations. We assume that the reader is familiar with Sections 3.1 and 3.2.

4.2.1 MPQ-trees

It is useful to have more information about the way in which the vertices of the interval graph are related to the structure of the PQ-tree. This additional information is

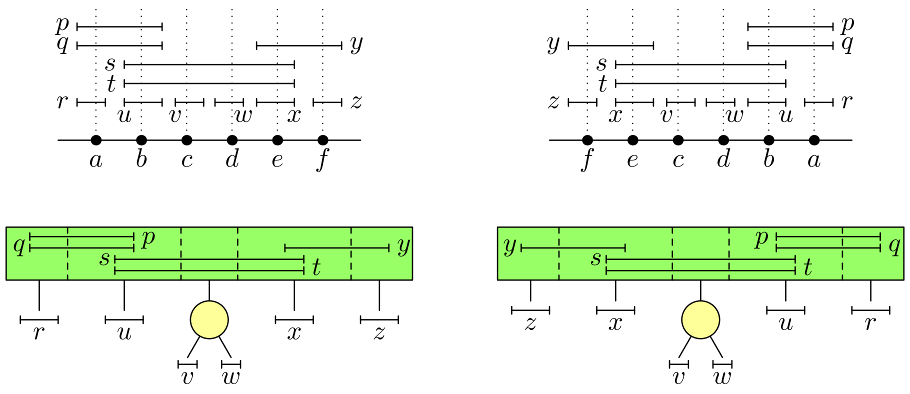


Figure 4.7: On top, two interval representations of an interval graph from Fig. 3.2. On bottom, the corresponding MPQ-trees, fully describing the structure of these interval representations. These MPQ-trees are augementations of the PQ-trees in Fig. 3.3.

contained in the *modified PQ-tree* (MPQ-tree), introduced by Korte and Möhring [235]. Their original motivation was to simplify the linear-time algorithm for recognizing interval graphs of Booth and Lueker [39] (see Lemma 3.1.4). MPQ-trees link positions of intervals directly to its nodes, which gives a great structural insight into an interval graph. We note that the same idea is already present in the earlier paper of Colbourn and Booth [68], in order to study automorphism groups of interval graphs.

Definition. The MPQ-tree is an augmentation of the PQ-tree in which the nodes of T have assigned subsets of $\boldsymbol{V}(G)$ called *sections*. To a leaf representing a clique a, we assign one section $s(a)$. Similarly, to each P-node P, we assign one section $s(P)$. For a Q-node Q with n children, we have n sections $s_1(Q), \dots, s_n(Q)$, each corresponding to one subtree of Q. Let $s(Q) = s_1(Q) \cup s_2(Q) \cup \cdots \cup s_n(Q)$.

Recall that $T[N]$ is the subtree of T having the node N as the root. Sections are defined as follows:

- *The section $s(a)$ of a maximal clique a* consists of all vertices contained in the maximal clique a and no other maximal clique.
- *The section $s(P)$ of a P-node P* consists of all vertices that are contained in all maximal cliques of $T[P]$ and in no other maximal clique.
- *The sections $s_i(Q)$ of a Q-node Q* are more complicated. Let $T_1, \dots, T_n$ be the subtrees of the children of Q. Let x be a vertex contained only in maximal cliques of $T[Q]$, and suppose that it is contained in maximal cliques of at least two subtrees of Q. Then x is contained in every section $s_i(Q)$ such that some maximal clique of T_i contains x.

Figure 4.7 depicts the sections for the example in Fig. 3.2.

Korte and Möhring [235] state the following properties:

- Every vertex x is placed in sections of exactly one node of T. In the case of a Q-node, it is placed in consecutive sections of this node.
- For a Q-node Q, if x is placed in a section $s_i(Q)$, then x is contained in all cliques of T_i.

- Every section of a Q-node is non-empty. Moreover, two consecutive sections have a non-empty intersection.
- A maximal clique contains exactly those vertices contained in the sections encountered when we traverse the tree from the leaf corresponding to this clique to the root.

Structure of MPQ-trees. Next, we show several structural properties used in building minimal obstructions which are quite easy to prove:

Lemma 4.2.1. *Let Q be a Q-node. Then $s_i(Q) \neq s_j(Q)$ for every $i \neq j$. Further, if $s_i(Q) \subsetneq s_{i+1}(Q)$, then at least one section of T_i is non-empty.*

Proof. If $s_i(Q) = s_j(Q)$, then we could exchange T_i and T_j and we would obtain a valid MPQ-tree for G. Since $n \geq 3$, this yields a contradiction with the fact that the only possible transformation of a Q-node is reverting the order of its children.

For the latter part, let a and b respectively be maximal cliques of a leaf in T_i and a leaf in T_{i+1}. Then $a \setminus b \neq \emptyset$ and every $x \in a \setminus b$ belongs to sections of T_i. □

The second part of Lemma 4.2.1 has the following reformulation. Either some interval starts and some interval ends in $s_i(Q)$, or some interval belongs to sections of T_i. Since the assumption cannot hold for $s_1(Q)$ and $s_n(Q)$, we know that some intervals belong to sections of T_1 and T_n.

For a subtree T of the MPQ-tree, let $G[T]$ be the interval graph induced by the vertices of the sections of T. For a node N, let $G[N] = G[T[N]]$. and its subtrees are the subtrees which have the children of N as the roots. For a (partial) representation $\mathcal{R}$, we have $\mathcal{R}[T] = \mathcal{R}[G[T]]$ and $\mathcal{R}[N] = \mathcal{R}[T[N]]$.

Lemma 4.2.2. *Let N be an inner node of an MPQ-tree.*

(i) If N is a Q-node, then $G[N]$ is connected.
(ii) If N is a P-node, then $G[N]$ is connected if and only if $s(N)$ is non-empty. Furthermore, for every child T_i of N, the graph $G[T_i]$ is connected.

Proof. (i) It follows from the facts that the vertices in any section form a clique, and that any two consecutive sections of N have non-empty intersection.

(ii) The first statement is clear. For the second part, notice that if $G[T_i]$ was not connected, we could permute the connected components of $G[T_i]$ arbitrarily with the other children of N. Therefore T_i would not be a child of N, but N would have one child per each connected component of $G[T_i]$. □

Let Q be a Q-node and $i < j$. Let x and y be two vertices of $G[Q]$, where x is either in T_i, or $s_i(Q)$, and y is either in T_j, or $s_j(Q)$. A path $P_{x,y}$ is called *Q-monotone* if all inner vertices of the path belong to the sections of Q, their leftmost sections strictly increase, and their rightmost ones strictly increase as well.

Lemma 4.2.3. *Let H be an induced subgraph of $G[Q]$ such that $x, y \in \boldsymbol{V}(H)$ belong to one component. Then every shortest path $P_{x,y}$ in H is Q-monotone.*

Proof. It is easy to see that any path from x to y that is not Q-monotone can be shortened. $\square$

Let Q be a Q-node with sections $s_1(Q), \dots, s_n(Q)$ and u be a vertex, belonging with Q to a common path from a leaf to the root.

- If u does not belong to sections of $T[Q]$, let $s_u^{\leftarrow}(Q) = s_1(Q)$ and $s_u^{\rightarrow}(Q) = s_q(Q)$.
- If $u \in s(Q)$, let $s_u^{\leftarrow}(Q)$ be the leftmost section of Q containing u and $s_u^{\rightarrow}(Q)$ be the rightmost one.
- If u belongs to sections of a subtree T_i of Q, we put $s_u^{\leftarrow}(Q) = s_u^{\rightarrow}(Q) = s_i(Q)$.

If $s_u^{\rightarrow}(Q)$ is on the left of $s_v^{\leftarrow}(Q)$, then we say that u *is on the left of* v and v *is on the right of* u. Also, u *and* v *are on the same side of* w if they are both on the left of w, or both on the right of w. Similarly, v *is between* u *and* w if either u is on the left and w is on the right of v, or u is on the right and w is on the left of v. For a maximal clique $a \in T_i$, we say that u *is on the left of* a when $s_u^{\rightarrow}(Q)$ is on the left of $s_i(Q)$, and similarly the other relations.

4.2.2 Basic Tools

Recall the definitions and results from Section 8.2. Next, we prove several tools which we use repeatedly in the proof of Theorem 2.2.1.

Non-adjacencies of Maximal Cliques. Maximal cliques of interval graphs have the following special property:

Lemma 4.2.4. *Let H be a connected subgraph of an interval graph and let c be a maximal clique with no vertex in $\boldsymbol{V}(H)$. There exists $x \in c$ non-adjacent to all vertices of $\boldsymbol{V}(H)$.*

Proof. Consider an interval representation $\mathcal{R}$. It places all intervals of H to one side of $\mathrm{cp}(c)$, say on the left. Let x be the interval of c having the rightmost left endpoint. If x is adjacent to some vertex $y \in \boldsymbol{V}(H)$, then every vertex of c is adjacent to y. Since c is maximal, it follows that $y \in c$, contradicting the assumption. So x is non-adjacent to all vertices of $\boldsymbol{V}(H)$. $\square$

Sliding Lemma. We introduce some notation. We denote by $P^{\hookleftarrow}(a)$ and $P^{\mapsto}(a)$ respectively the subsets of $P(a)$ containing the pre-drawn intervals with left-most right endpoints, and with right-most left endpoints. If $u \in P^{\hookleftarrow}(a)$ and $v \in P^{\mapsto}(a)$, then $\langle u \rangle' \cap \langle v \rangle' = \bigcap_{w \in P(a)} \langle w \rangle'$, thus I_a is a subinterval of $\langle u \rangle' \cap \langle v \rangle'$.

Single overlaps of pre-drawn intervals pose more constraints than containment (for comparison, see Fig. 3.4b and c). Therefore, single overlaps are more powerful in building obstructions. The following lemma states that, under some assumptions, we can turn a containment of pre-drawn intervals into a single overlap of other pre-drawn intervals; see Fig. 4.8.

Lemma 4.2.5 (Sliding)**.** *Let I_a be on the left of I_b, $P(a) \subsetneq P(b)$ and $r \in P(b) \setminus P(a)$.*

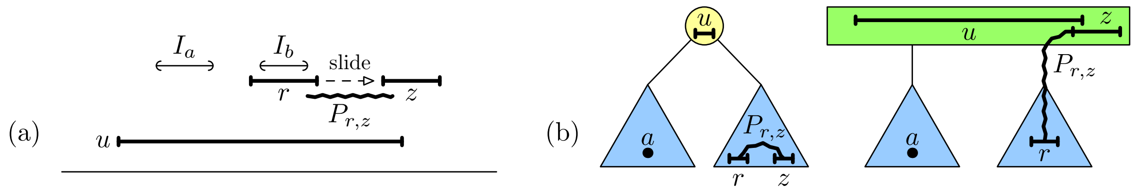


Figure 4.8: (a) With the assumption satisfied, we can slide r to z which covers $r(u)$. (b) The relative positions in the MPQ-trees of z and r with respect to a are the same.

(i) *There exists a pre-drawn interval $\langle z\rangle'$ on the right of I_a covering $r(u)$, for $u \in P^{\hookleftarrow}(a)$. Further, there exists an induced path $P_{r,z}$ from r to z whose vertices are all pre-drawn and not contained in $P(a)$.*

(ii) *Consider the smallest subtree having a and the sections containing r. If the root of this subtree is a P-node, then z and r are contained in the same subtree. If the root is a Q-node, then z and r appear on the same side of a.*

Proof. (i) By the assumptions and the definition of I_a, we get that $(\curvearrowright(a), r(u)]$ is not empty and all points in $(\curvearrowright(a), r(u)]$ are covered by pre-drawn intervals not in $P(a)$. Among these intervals, we choose z covering $r(u)$. Since r is also one of these intervals, we can construct an induced path from r to z, consisting of pre-drawn intervals not in $P(a)$.

(ii) It follows from the existence of $P_{r,z}$ not contained in a. □

We note that possibly $r = z$. The above lemma is repeatedly used for constructing minimal obstructions. The general idea is the fact that $\langle r\rangle'$ properly contained inside $\langle u\rangle'$ restricts the partial representation less than $\langle z\rangle'$ covering $r(u)$. The lemma says that we can assume that such z exists and use it instead of r.

Flip Operation. We say that we *flip* the partial representation *vertically* when we map every $x \in \mathbb{R}$ to $-x$. This reverses the ordering $\lhd$. Clearly, there exists an obstruction in the original partial representation if and only if the flipped obstruction is present in the flipped partial representation. The purpose of this operation is to decrease the number of cases in the proofs.

4.3 Strategy for Finding Minimal Obstructions

In this section, we describe the general strategy to show that every non-extendible partial representation contains one of the obstructions described in Section 4.1. In the remaining sections, unless stated otherwise, by a clique, we always mean a maximal clique.

Recall that we write $T_i \lhd T_j$ if and only if there exist cliques $a \in T_i$ and $b \in T_j$ such that $a \lhd b$. We assume that the reader is familiar with the reordering algorithm for general orderings, described in Subsection 3.3.1. Suppose that some node cannot be reordered by this algorithm, and we call this node *obstructed*. A set of maximal cliques *creates* an obstruction if already the ordering of these cliques in $\lhd$ makes the node obstructed.

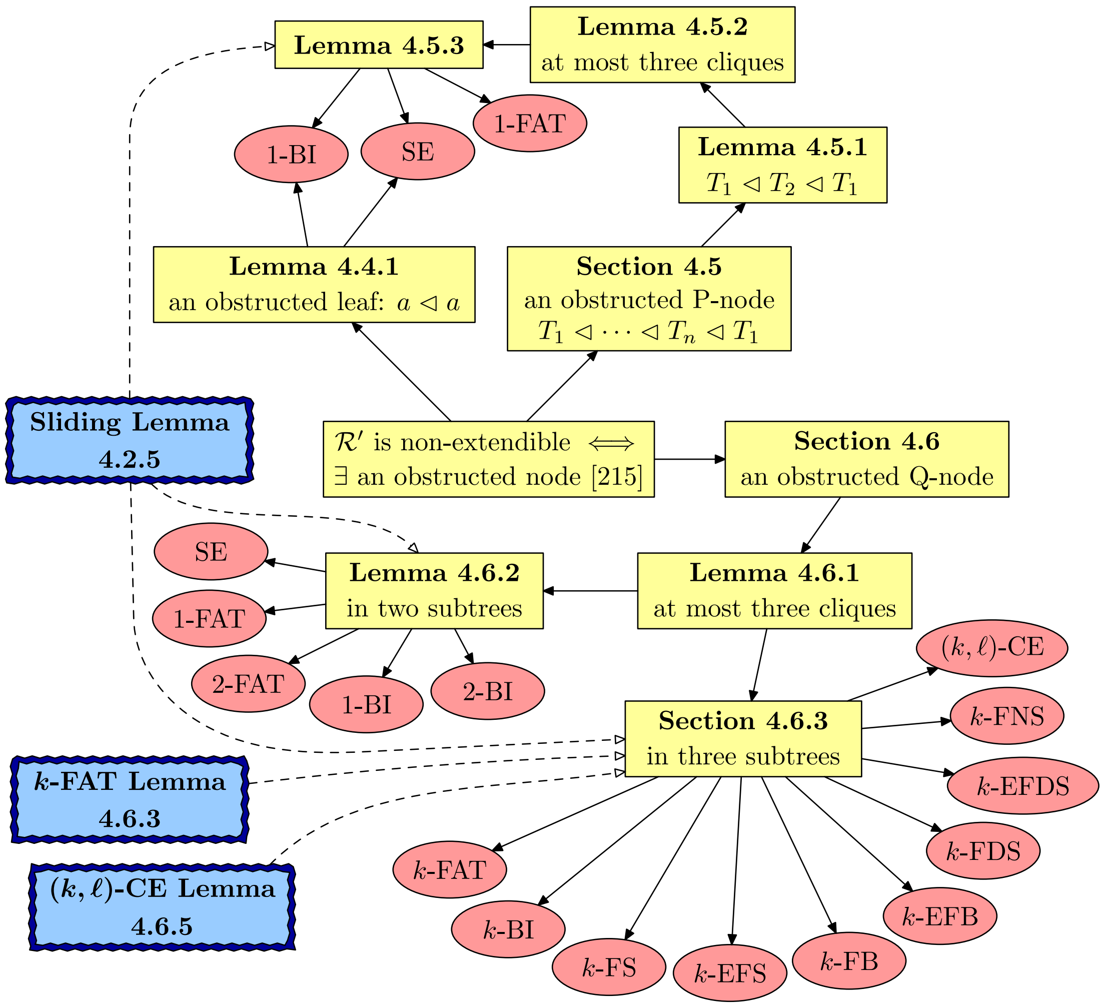


Figure 4.9: Overview of the main steps of the proof, it starts in the middle. The obtained obstructions are highlighted in gray, and three tools are depicted with highlighted borders. The most involved case is in Section 4.6.3.

Strategy. Suppose that a partial representation $\mathcal{R}'$ is non-extendible. By Theorem 3.2.2 and Proposition 3.3.3, we know that there exists an obstructed node in the MPQ-tree. We divide the argument into three cases, according to the type of this node: *an obstructed leaf* (Section 4.4), *an obstructed P-node* (Section 4.5), and *an obstructed Q-node* (Section 4.6). Figure 4.9 shows an overview of the proof.

First, we argue that there exist at most three maximal cliques creating an obstruction. Then, we consider their positions in the MPQ-tree and their open intervals from the definition of $\lhd$. We use tools of Section 4.2 to derive positions of several pre-drawn intervals forming one of the obstructions.

In Section 4.6.2, we prove a key tool called k-FAT Lemma 4.6.3: If three non-adjacent vertices x_k, y_k, and z_k are pre-drawn in an order that is different from their order in the sections of a Q-node, then they induce a k-FAT obstruction. The proof is done by induction for k, and it explains why complicated obstructions are needed.

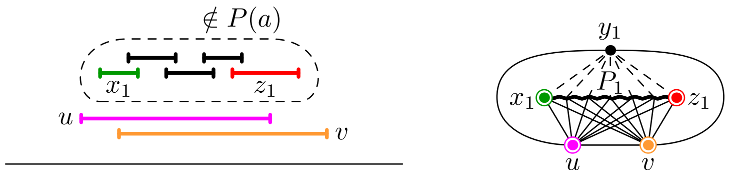


Figure 4.10: A construction leading to a 1-BI obstruction.

4.4 Obstructed Leaves

Suppose that some clique-point a cannot be placed, so $\downarrow_a = \emptyset$. In terms of $\lhd$, we get $a \lhd a$. Since $\lhd$ is a strict partial ordering, this already makes the partial representation non-extendible.

Lemma 4.4.1 (The leaf case)**.** *If a leaf is obstructed, then G and $\mathcal{R}'$ contain an SE, or 1-BI obstruction.*

Proof. We name the vertices as in the definition of the 1-BI obstructions. Suppose that the leaf corresponds to a maximal clique a such that $\downarrow_a = \emptyset$.

Let $u \in P^{\leftarrow}(a)$ and $v \in P^{\mapsto}(a)$ (possibly $u = v$). Since I_a is a subinterval of $\bigcap_{w \in P(a)} \langle w \rangle'$ and $\downarrow_a = \emptyset$, every point of $[\ell(v), r(u)]$ is covered by some pre-drawn interval not contained in $P(a)$. Let $\langle x_1 \rangle'$ be one such interval covering $\ell(v)$ and let $\langle z_1 \rangle'$ be one such interval covering $r(u)$ (again, possibly $x_1 = z_1$); see Fig. 4.10. Let P_1 be a shortest path from x_1 to z_1 consisting of pre-drawn intervals not in $P(a)$.

We prove that the relative pre-drawn position of u, v, x_1, and z_1 makes the partial representation non-extendible. The maximal clique a does not contain any vertex of P_1. Since the vertices of P_1 induce a connected subgraph, by Lemma 4.2.4 there exists $y_1 \in a$ which is non-adjacent to all vertices of P_1. Hence, these (at most) five vertices together with P_1 create either a 1-BI obstruction (when $\ell(u) < r(v)$), or an SE obstruction (when $\ell(u) = r(v)$, for which $x = x_1 = z_1$ and we can free it). We note that this obstruction might not be minimal, in which case we can remove some vertices and get one of the minimal obstructions illustrated in Fig. 4.2 and 4.5. $\square$

4.5 Obstructed P-nodes

If a P-node is obstructed, then it has some subtrees $T_1, \dots, T_n$ forming the cycle $T_1 \lhd T_2 \lhd \cdots \lhd T_n \lhd T_1$. We start by showing that the specific structure of $\lhd$ forces the existence of a two-cycle, so we can assume that $n = 2$.

Lemma 4.5.1. *If a P-node is obstructed, then it has two subtrees T_1 and T_2 such that $T_1 \lhd T_2 \lhd T_1$.*

Proof. The proof is illustrated in Fig. 4.11a. Let $T_1 \lhd \cdots \lhd T_n \lhd T_1$ be a shortest cycle for the obstructed P-node. To get a contradiction, we assume $n \geq 3$. Since $T_1 \lhd T_2$, there exist $a \in T_1$ and $b \in T_2$ such that $a \lhd b$. Similarly, there exist $c \in T_2$

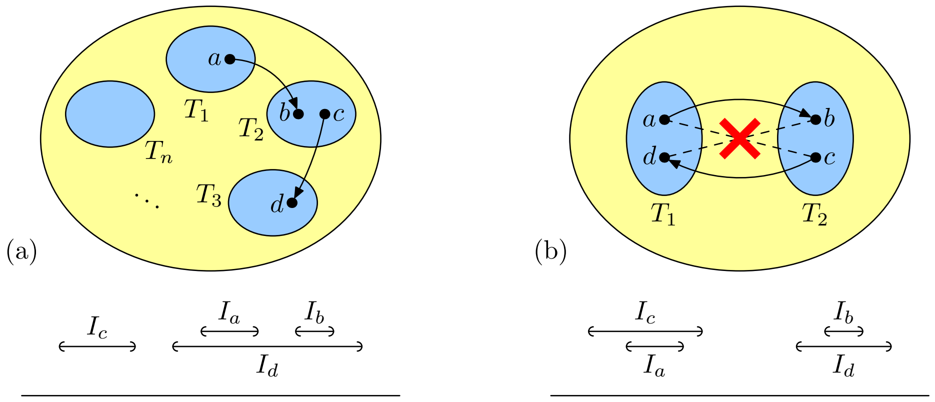


Figure 4.11: (a) At the top, a shortest n-cycle of $\lhd$ on the children of a P-node. At the bottom, the derived positions of the open intervals. (b) At the top, the four cliques involved in a two-cycle in $\lhd$. The cliques a and c are incomparable, and so are b and d. At the bottom, one of the four possible configurations of the open intervals.

and $d \in T_3$ such that $c \lhd d$. We know that I_a is on the left of I_b, and I_c is on the left of I_d. We analyze the remaining relative positions.

First, I_d is not on the right of I_a, since otherwise $T_1 \lhd T_3$ and a shorter cycle would exist. Additionally, I_d is not on the left of I_b, since we would get $T_3 \lhd T_2$, and T_2 and T_3 would form a two-cycle. According to Lemma 3.2.3, no single overlaps of open intervals are allowed, so I_d necessarily contains both I_a and I_b; see Fig. 4.11a. Therefore, I_c is on the left of I_a, so $T_2 \lhd T_1$ and we get a two-cycle. □

We note that an alternative proof follows directly from Lemma 3.3.7.

To create a two-cycle, at most four cliques are enough. Aside from Lemma 3.2.3, so far we have not used that $\lhd$ arises from a partial interval representation. Next, we use properties of the MPQ-tree.

Lemma 4.5.2. *A two-cycle $T_1 \lhd T_2 \lhd T_1$ is created by at most three maximal cliques.*

Proof. The proof is depicted in Fig. 4.11b. Suppose that this two-cycle is given by four cliques $a, d \in T_1$ and $b, c \in T_2$ such that $a \lhd b$ and $c \lhd d$. Assume for contradiction that no three of these cliques define the two-cycle, i.e., a and c are incomparable, and so are b and d. According to Lemma 3.2.3, $I_a \subseteq I_c$ or $I_a \supseteq I_c$, and analogously for I_b and I_d. In all of the four cases, I_c is on the left of I_b, and I_d is on the right of I_a.

We look at the case where $I_a \subseteq I_c$ and $I_b \subseteq I_d$, as in Fig. 4.11b. By Lemma 3.2.4, we have $P(c) \subsetneq P(a)$. Therefore, $P(c)$ contains no vertices from the sections of T_2. Similarly, $P(d) \subsetneq P(b)$, and $P(d)$ contains no vertices from the sections of T_1. Therefore $P(c) = P(d)$, which implies $I_c = I_d$, a contradiction. The other cases can be analyzed similarly. □

It remains to put these results together and characterize the possible obstructions.

Lemma 4.5.3 (The P-node case)**.** *If a P-node is obstructed, then G and $\mathcal{R}'$ contain an SE, 1-FAT, or 1-BI obstruction.*

Proof. According to Lemma 4.5.1, the obstructed P-node has a two-cycle in $\lhd$. By Lemma 4.5.2, there are at most three maximal cliques defining this cycle. First assume that this cycle is defined by two cliques $a \in T_1, b \in T_2$ such that $a \lhd b \lhd a$. According to the definition of $\lhd$, this implies that $I_a = I_b$, both of lenght zero. Therefore $P(a) = P(b)$. Let $u \in P^{\mapsfrom}(a)$ and $v \in P^{\mapsto}(a)$ (possibly $u = v$); we have that $\langle u\rangle' \cap \langle v\rangle'$ is a singleton. Since a and b are two maximal cliques, there exists $x \in a \setminus b$ and $y \in b \setminus a$. We get an SE obstruction.

It remains to deal with the case where three cliques define the two-cycle. Let $a, c \in T_1$ and $b \in T_2$ such that $a \lhd b \lhd c$. We have three non-intersecting intervals whose left-to-right order is I_a, I_b and I_c. Since I_a and I_c are disjoint, one of the sets $P(a) \setminus P(c)$ and $P(c) \setminus P(a)$ is non-empty. Without loss of generality, we assume that $P(a) \setminus P(c) \neq \emptyset$. Let $p \in P(a) \setminus P(c)$; then p belongs to sections of T_1, and as a consequence $p \notin P(b)$. Therefore $\langle p\rangle'$ is on the left of I_b. We distinguish two cases.

Case 1: $P(b) \setminus P(c) \neq \emptyset$. We choose $q \in P(b) \setminus P(c)$. Then q belongs to sections of T_2, and $\langle q\rangle'$ is between $\langle p\rangle'$ and I_c. In the next paragraph we show that there also exists $r \in P(c) \setminus P(b)$. Then r belongs to sections of T_1, and $\langle r\rangle'$ is on the right of $\langle q\rangle'$, as in Fig. 4.12a. By Lemma 4.2.2(ii), $G[T_1]$ is connected; let P_1 be a shortest path from p to r in $G[T_1]$. We obtain a 1-FAT obstruction for $x_1 = p$, $y_1 = q$ and $z_1 = r$.

It remains to show that such r exists. Suppose for contradiction that $P(c) \setminus P(b) = \emptyset$. Since $P(c) \subsetneq P(b)$, no vertex of $P(c)$ appears in sections of T_1, and we get $P(c) \subsetneq P(a)$. Consequently, every pre-drawn interval of $P(c)$ contains $[\curvearrowleft(a), \curvearrowright(c)]$. The position of I_c implies that every point of $[\curvearrowleft(a), \curvearrowleft(c))$ is covered by some pre-drawn interval not contained in $P(c)$. In particular, there exists a path from p to q consisting of such intervals. Since p belongs to sections of T_1 and q belongs to sections of T_2, every path from p to q contains a vertex of the section of the P-node, or of a section above it; hence, the path contains a vertex belonging to c. We obtain a contradiction.

Case 2: $P(b) \setminus P(c) = \emptyset$. Then there exists $r \in P(c) \setminus P(b)$. We again observe that $\langle r\rangle'$ is on the right of I_b, as depicted in Fig. 4.12b. Furthermore, $P(b) \subseteq P(a) \cap P(c)$, so every pre-drawn interval of $P(b)$ contains $[\curvearrowleft(a), \curvearrowright(c)]$.

We construct a 1-BI obstruction and we name the vertices as in the definition.

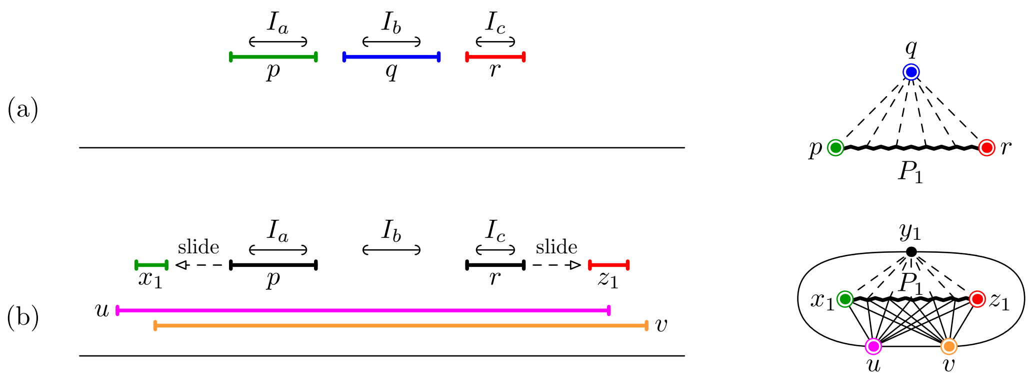


Figure 4.12: The two cases of the proof of Lemma 4.5.3. (a) Case 1 leads to a 1-FAT obstruction. (b) Case 2 leads to a 1-BI obstruction.

Let $u \in P^{\leftarrow}(b)$ and $v \in P^{\mapsto}(b)$ (possibly $u = v$). Since p does not necessarily cover $\ell(v)$ and r does not necessarily cover $r(u)$, we might not be able to construct a 1-BI obstruction with $x_1 = p$ and $z_1 = q$. We instead use Sliding Lemma 4.2.5. By applying it (flipped) to I_b, I_a and p, we obtain a pre-drawn interval x_1 covering $\ell(v)$ (possibly $x_1 = p$). By applying it to I_b, I_c and r, we obtain a pre-drawn interval z_1 covering $r(u)$ (possibly $z_1 = r$). Furthermore, x_1 and z_1 belong to sections of T_1. Since $G[T_1]$ is connected by Lemma 4.2.2(ii), there exists a shortest path P_1 from x_1 to z_1 containing no vertex of b. By Lemma 4.2.4, there exists $y_1 \in b$ non-adjacent to all vertices of P_1. We obtain a 1-BI obstruction. □

4.6 Obstructed Q-nodes

Suppose that a Q-node with subtrees $T_1, \dots, T_n$ is obstructed. Then the two possible orderings of this Q-node are not compatible with $\lhd$. Notice that at most four cliques are sufficient to create the obstruction. We next prove that at most three cliques are already sufficient.

Lemma 4.6.1. *If a Q-node is obstructed, there exists an obstruction created by at most three maximal cliques.*

Proof. Suppose that an obstruction is created by four cliques $a \in T_\alpha$, $b \in T_\beta$, $c \in T_\gamma$ and $d \in T_\delta$ such that $\alpha < \beta$, $\gamma < \delta$, $a \lhd b$, and $c \rhd d$. We know that I_a is on the left of I_b, and I_c is on the right of I_d. Notice that the four subtrees T_α, T_β, T_γ and T_δ are not necessarily distinct. We classify all possible orderings $<$ of α, β, γ, δ in two general cases, namely, $\alpha \neq \gamma$ and $\alpha = \gamma$. In the first case, we may assume without loss of generality that $\alpha < \gamma$. (Otherwise, we apply the flip operation and swap a with c and b with d in the argument.)

Case 1: $\alpha < \gamma < \delta$ (see Fig. 4.13a). Consider the relative positions of I_c and I_d with respect to I_a. If I_d is to the left of I_a, we have $d \lhd a \lhd b$, and these three cliques already create an obstruction. If I_c is to the right of I_a, then we get $a \lhd c$ and $c \rhd d$, creating an obstruction. If neither happens, then I_c and I_d are subintervals of I_a. Thus $c, d \lhd b$. If $\beta \leq \gamma$, we have $a \lhd b$ and $b \rhd d$, creating an obstruction. If $\beta > \gamma$, then $d \lhd c \lhd b$, which also creates an obstruction.

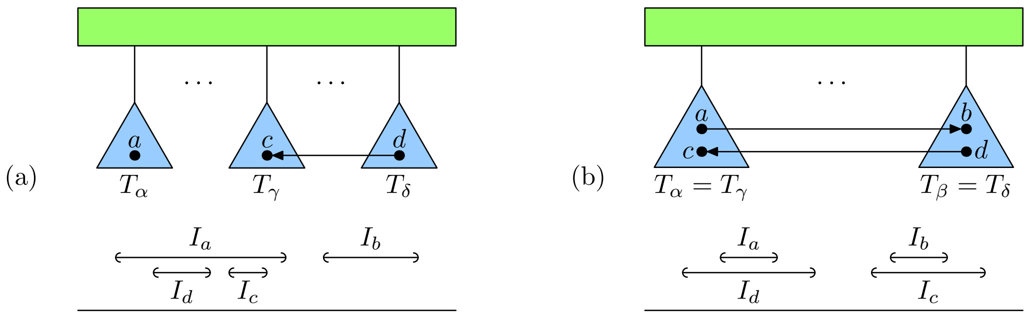


Figure 4.13: Two cases of the proof of Lemma 4.6.1. The Q-node is depicted in the top, while in the bottom we have the relative positions of the intervals.

Case 2: $\alpha = \gamma$ (see Fig. 4.13b). If I_c does not intersect I_b, or I_d does not intersect I_a, it is easy to see that three of the cliques already create an obstruction. Suppose next that these intersections occur. Then $d \lhd b$. If $\delta < \beta$ or $\beta < \delta$, it is again easy to show that three cliques are enough to create an obstruction. It only remains to consider the case where $\alpha = \gamma < \beta = \delta$.

Since the intervals I_c and I_a are non-intersecting, we may assume without loss of generality that there exists $x \in P(a) \setminus P(c)$. This vertex x belongs to sections of T_α. Thus $x \notin P(d)$, and we get that $I_a \subsetneq I_d$. By Lemma 3.2.4, $P(d) \subsetneq P(a)$; in particular, $P(d)$ contains no private pre-drawn interval from sections of T_β, and all pre-drawn intervals of $s_\beta(Q)$ are also contained in $s_\alpha(Q)$.

Since $P(d) \setminus P(b) = \emptyset$, there exists $y \in P(b) \setminus P(d)$ which is contained in sections of T_β. We next apply the argument in the previous paragraph, and obtain $y \notin P(c)$, $I_b \subsetneq I_c$, and $P(c) \subsetneq P(b)$. Consequently, $P(c)$ contains no private pre-drawn intervals from sections of T_α, and all pre-drawn intervals of $s_\alpha(Q)$ are contained in $s_\beta(Q)$. We conclude that $P(c) = P(d)$ and $I_c = I_d$, which gives a contradiction. $\square$

In summary, we can assume that a minimal obstruction involves at most three maximal cliques. These three cliques belong to either two or three different subtrees.

In the rest of the section, many figures describe positions of derived pre-drawn intervals in sections of the Q-node and its subtrees; for instance Fig. 4.14. Some of these intervals necessarily belong to sections of the Q-node, since they belong to maximal cliques of several subtrees; for instance t_2 in Fig. 4.14. But for the remaining intervals, it is not important to distinguish whether they belong to sections of the Q-node or one of its subtrees, only their relative positions in the Q-node matter; for instance q and x_1 in Fig. 4.14.

4.6.1 Cliques in Two Different Subtrees

In this subsection, we deal with the case where the maximal cliques belong to two different subtrees.

Lemma 4.6.2 (The Q-node case, Two Subtrees)**.** *If at most three cliques creating the obstruction belong to two different subtrees, then G and $\mathcal{R}'$ contain an SE, 1-FAT, 2-FAT, 1-BI, or 2-BI obstruction.*

Proof. The proof is similar to that of Lemma 4.5.3, but more involved. If two maximal cliques create an obstruction, we can argue as in the first paragraph of the proof of Lemma 4.5.3, and we obtain an SE obstruction. It remains to deal with the case of three maximal cliques a, b, and c.

We can assume that $a \lhd b \lhd c$ and that, for some $i < j$, we have $a, c \in T_i$ and $b \in T_j$. Furthermore, without loss of generality, there exist $p \in P(a) \setminus P(c)$. Since p belongs to sections of T_i, then $p \notin P(b)$, and thus $\langle p \rangle'$ lies to the left of I_b. We distinguish two cases.

Case 1: $P(b) \setminus P(c) \neq \emptyset$. Then there exists $q \in P(b) \setminus P(c)$ such that $\langle q \rangle'$ lies between $\langle p \rangle'$ and I_c. Since q is non-adjacent to p, it belongs to sections of either Q or

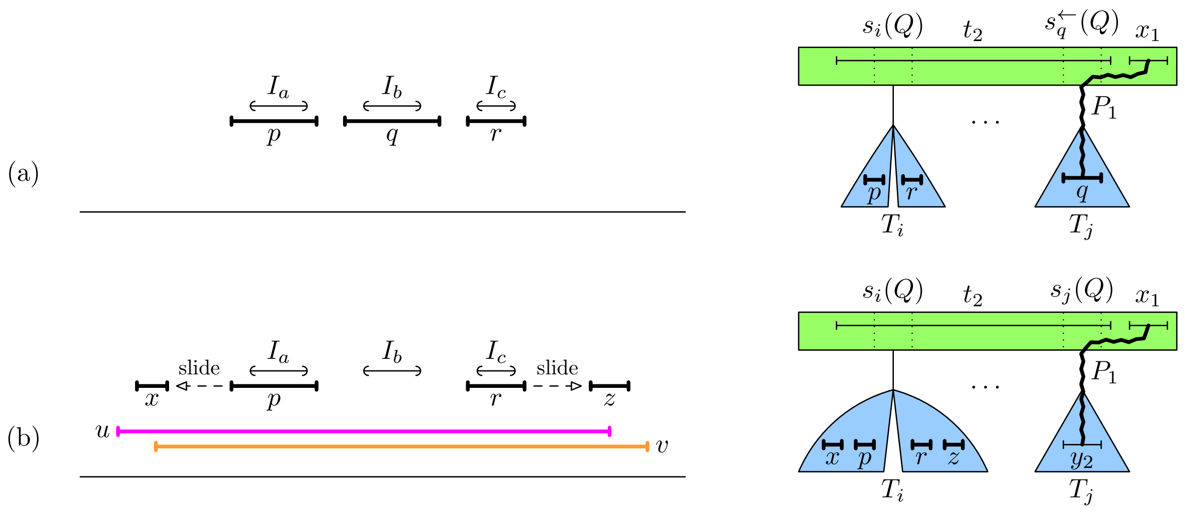


Figure 4.14: (a) Case 1: The pre-drawn intervals and the situation in the MPQ-tree for $s_i(Q) \subsetneq s_q^{\leftarrow}(Q)$. (b) Case 2: The pre-drawn intervals and the situation when there exists no path from x to z avoiding the vertices of b.

T_j. Notice that in any case $s_q^{\leftarrow}(Q)$ is on the right of $s_i(Q)$. Arguing as in Case 1 of the proof of Lemma 4.5.3, we observe that there exists $r \in P(c) \setminus P(b)$. Furthermore, it follows that $\langle r \rangle'$ lies to the right of $\langle q \rangle'$; see Fig. 4.14a on the left.

If there exists a path P_1 from p to r avoiding $N[q]$, we get a 1-FAT obstruction for $x_1 = p$, $y_1 = q$, $z_1 = r$ and P_1. By Lemma 4.2.1, we know that $s_i(Q) \neq s_q^{\leftarrow}(Q)$. If $s_i(Q) \not\subseteq s_q^{\leftarrow}(Q)$, then there exists some $w \in s_i(Q) \setminus s_q^{\leftarrow}(Q)$. Therefore, $P_1 = pwr$ is such a path. It remains to deal with the case where no such path P_1 exists, which implies that $s_i(Q) \subsetneq s_q^{\leftarrow}(Q)$; see Fig. 4.14a on the right.

Consider the set $W = s_i(Q)$. Let t_2 be a vertex of W whose section $s_{t_2}^{\rightarrow}(Q)$ is leftmost. Let C be the component of $G[Q] \setminus W$ containing q. Since $s_q^{\leftarrow}(Q) \setminus W$ is non-empty, C consists of the vertices of at least two subtrees of the Q-node. If t_2 was adjacent to all vertices of C, it would be possible to flip the ordering of this component, contradicting the fact that there are only two possible orderings for Q. Therefore, t_2 is not adjacent to all vertices of C. We choose $x_1 \in C \setminus N[t_2]$ whose section $s_{x_1}^{\leftarrow}(Q)$ is leftmost. Let P_1 be a shortest path from q to x_1 whose inner vertices are adjacent to t_2. It follows that $x_2 = p$, $y_2 = q$, $z_2 = r$, $P_2 = x_2t_2$, t_2, x_1, and P_1 define a 2-FAT obstruction. (By Lemma 4.2.3, all inner vertices of P_1 are adjacent to t_2.)

Case 2: $P(b) \setminus P(c) = \emptyset$. Then there exists $r \in P(c) \setminus P(b)$. Since $\langle r \rangle'$ lies on the right of I_b, the vertex r is not contained in a and it belongs to sections of T_i. We use the same approach as in Case 2 of the proof of Lemma 4.5.3. Since $P(b) \subseteq P(a) \cap P(c)$, every pre-drawn interval of $P(b)$ covers $[\curvearrowleft(a), \curvearrowright(c)]$. Let $u \in P^{\mapsfrom}(b)$ and $v \in P^{\mapsto}(b)$ (possibly $u = v$).

By applying Sliding Lemma 4.2.5 twice, we get $x, z \notin P(b)$ such that $\langle x \rangle'$ covers $\ell(v)$ and $\langle z \rangle'$ covers $r(u)$; see Fig. 4.14b on the left.

Suppose that there exists a path $P_{x,z}$ from x to z avoiding all vertices of b. Let $x_1 = x$, $z_1 = z$, and P_1 be a shortest path from x_1 to z_1 in $G[Q] \setminus b$. By Lemma 4.2.4, there exists $y_1 \in b$ non-adjacent to P_1. We obtain a 1-BI obstruction.

Suppose next that there is no path $P_{x,z}$ avoiding b. We know that x and z belong to sections of T_i, since there exist paths $P_{x,p}$ and $P_{r,z}$ avoiding b, from the above applications of Sliding Lemma 4.2.5. Since no path $P_{x,z}$ avoiding b exists, we have $s_i(Q) \subsetneq s_j(Q)$. As in Case 1, let $W = s_i(Q)$, and let t_2 be a vertex of W whose section $s^{\rightarrow}_{t_2}(Q)$ is leftmost (possibly $t_2 = u$ or $t_2 = v$). We again infer that t_2 is not adjacent to all vertices of C, where C is the component of $G[Q] \setminus W$ containing $b \setminus W$. We choose $x_1 \in C \setminus N[t_2]$ whose section $s^{\leftarrow}_{x_1}(Q)$ is leftmost. Since $s_i(Q) \subsetneq s_j(Q) \subseteq b$, there exists $y_2 \in b$ non-adjacent to x and z. We get a 2-BI obstruction for $x_2 = x$, y_2, $z_2 = z$, u, v, a shortest path P_1 from y_2 to x_1 in C, and $P_2 = x_2t_2$. (By Lemma 4.2.3, all inner vertices of P_1 are adjacent to t_2.) □

4.6.2 k-FAT and (k,ℓ)-CE Lemmas

In this subsection, we give two tools for the case, analyzed in Section 4.6.3, where the three maximal cliques creating the obstruction belong to three different subtrees. These tools give insight into the structure of the Q-nodes, and explain the way in which complex obstructions such as k-FAT and (k,ℓ)-CE obstructions are formed.

***k*-FAT Lemma.** First, we present a useful lemma that allows to locate k-FAT obstructions. The key idea of the proof is similar to Case 1 of the proof of Lemma 4.6.2, but applied inductively for k.

Lemma 4.6.3 (k-FAT)**.** *Let Q be a Q-node with children $T_1, \dots, T_n$, and let a, b and c be three cliques of $T[Q]$ contained respectively in T_α, T_β and T_γ, for $\alpha < \beta < \gamma$. Let $x_k \in P(a)$, $y_k \in P(c)$ and $z_k \in P(b)$ be three disjoint pre-drawn intervals such that $\langle y_k\rangle'$ is between $\langle x_k\rangle'$ and $\langle z_k\rangle'$. Then $G[Q]$ and $\mathcal{R}'[\{x_k, y_k, z_k\}]$ contain a k-FAT obstruction.*

Proof. The proof, illustrated in Fig. 4.15, is by induction. We always denote the vertices as in the definition of k-FAT obstructions. If we find a 1-FAT or 2-FAT obstruction, the statement is true. Otherwise, we recurse on a smaller part of the Q-node, where we find a structure identical to a $(k-1)$-FAT obstruction, except for the fact that the vertex x_{k-1} is free. Together with some vertices in the remainder of the Q-node, we obtain a k-FAT obstruction. We next provide the details.

Let k be some yet unspecified integer, determined by the recursion. We want to argue that $G[Q]$ contains a k-FAT obstruction because the ordering of $\langle x_k\rangle'$, $\langle y_k\rangle'$ and

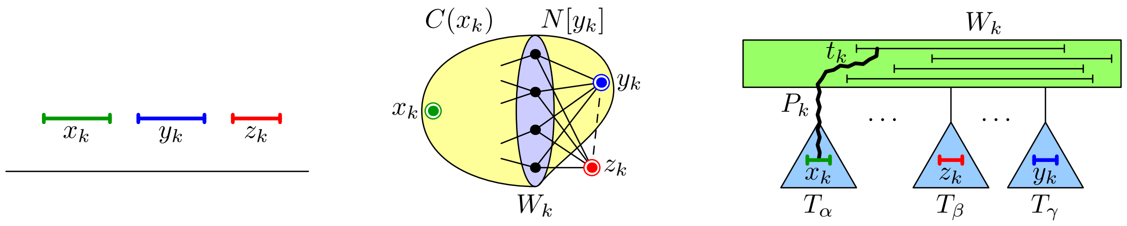


Figure 4.15: On the left, the position of the pre-drawn intervals. In the middle, the construction of $W_k \subsetneq N[y_k]$ in $G[Q]$. On the right, the Q-node with the three subtrees and the intervals of W_k depicted in its sections.

$\langle z_k \rangle'$ is incorrect (in every representation, $\langle z_k \rangle$ is between $\langle x_k \rangle$ and $\langle y_k \rangle$). Suppose that there exists a path from x_k to z_k whose inner vertices are non-adjacent to y_k. Then we obtain a 1-FAT obstruction. It remains to deal with the harder situation where no such path exists.

Let $C(x_k)$ be the connected component of $G[Q] \setminus N[y_k]$ containing x_k. By our assumption, $z_k \notin C(x_k)$. We denote by W_k the subset of $N[y_k]$ containing those vertices that are adjacent to some vertex of $C(x_k)$; see Fig. 4.15, middle. Notice that the vertices of $C(x_k)$ appear only in sections and subtrees to the left of $s_\beta(Q)$. Therefore, every vertex of W_k lies in the sections of Q and stretches from the left of $s_\beta(Q)$ to $s_\gamma(Q)$; see Fig. 4.15, right. In other words, $W_k \subseteq s_\beta(Q) \cap s_\gamma(Q)$ and every vertex of W_k is adjacent to z_k.

Let C be a connected component of $G[Q] \setminus W_k$. If C contains a vertex from some section of Q, we call it *big*. Notice that in this situation C has a vertex contained in two consecutive sections of Q and their subtrees. Otherwise, C consists of some vertices of a subtree of Q, and we call it *small*. The section above a subtree containing a small component is a subset of W_k. Additionally, if two small components are placed in two different subtrees, the two sections above these subtrees are different.

The graph $G[Q] \setminus W_k$ is disconnected, as x_k and z_k belong to different components. Let us denote the connected component containing y_k by $C(y_k)$, and the one containing z_k by $C(z_k)$. Let t_k be a vertex of W_k whose section $s^{\rightarrow}_{t_k}(Q)$ is leftmost. Let P_k be a shortest path from x_k to t_k in $G[C(x_k) \cup \{t_k\}]$; see Fig. 4.15, right. We distinguish two cases.

Case 1: $C(y_k) \neq C(z_k)$. This case is very similar to the proof of Lemma 4.6.2; see Fig. 4.16a. Every vertex of W_k is adjacent to some vertex of $C(x_k)$ and to some vertex of $C(y_k)$. Therefore, it is also adjacent to every vertex of $C(z_k)$. If $C(z_k)$ was big, then we could reverse its sections in the Q-node, contradicting the fact that there are only two possible orderings for a Q-node. Therefore, $C(z_k)$ is small. Notice that then $C(y_k)$ is not small, since otherwise we would get $s_\beta(Q) = W_k = s_\gamma(Q)$, contradicting Lemma 4.2.1. Thus, $C(y_k)$ is big.

Let us set $y_{k-1} = z_k$ and $z_{k-1} = y_k$. The vertex t_k is not universal for $C(y_k)$; otherwise, every vertex of W_k would be universal and this would give additional orderings of $C(y_k)$ in Q. Let x_{k-1} be a vertex of $C(y_k) \setminus N[t_k]$ whose section $s^{\leftarrow}_{x_{k-1}}(Q)$

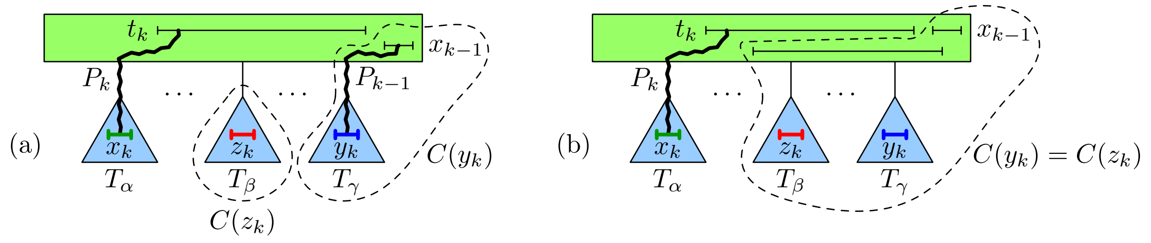


Figure 4.16: (a) In Case 1, there exists a path P_{k-1} from x_{k-1} to y_k whose inner vertices avoid z_k. (b) In Case 2, we have $C(y_k) = C(z_k)$ and such a path might no longer exist. For instance, every path from x_{k-1} to y_k in $C(y_k)$ might use the depicted interval in the sections of Q, which is also adjacent to z_k.

is leftmost. Notice that $s^{\leftarrow}_{x_{k-1}}(Q)$ is always the next section to $s^{\rightarrow}_{t_k}(Q)$. Let P_{k-1} be a shortest path from x_{k-1} to z_{k-1} in $C(y_k)$. By Lemma 4.2.3, all inner vertices of P_{k-1} are adjacent to t_k. Since this path lies in $C(y_k)$, the inner vertices are non-adjacent to y_{k-1}, x_k and P_k. We have constructed a 2-FAT obstruction.

Case 2: $C(y_k) = C(z_k)$. In this case, the component $C(y_k)$ is big; see Fig. 4.16b. Therefore, similarly as above, t_k is not universal for $C(y_k)$. We put $y_{k-1} = z_k$ and $z_{k-1} = y_k$. We choose $x_{k-1} \in C(y_k) \setminus N[t_k]$ in the same way as in Case 1. Notice that x_{k-1} is a non-neighbor of y_{k-1}, since otherwise it would be a neighbor of t_k. On the other hand, x_{k-1} might be adjacent to z_{k-1} or not. If it is, we get a 2-FAT obstruction for $k = 2$ with $P_1 = x_{k-1}z_{k-1}$. If it is not, we proceed as follows.

As before, every shortest path from x_{k-1} to z_{k-1} has all inner vertices adjacent to t_k. Since all vertices of $C(y_k)$ are non-adjacent to x_k and the inner vertices of P_k, every shortest path satisfies this as well. There exists a shortest path from x_{k-1} to z_{k-1} in $C(y_k)$, but we cannot guarantee that the inner vertices of this path are non-adjacent to y_{k-1}. We solve this issue by applying the entire argument of the proof recursively to $C(y_k)$.

In every representation extending the partial representation, the intervals of $C(x_k)$ form a connected subset of the real line placed to the left of $\langle y_k\rangle'$. Therefore, $\langle t_k\rangle$ stretches from $C(x_k)$ to $\langle z_k\rangle'$, covering $\langle y_k\rangle'$. Thus $\langle x_{k-1}\rangle$ is placed to the right of $\langle z_k\rangle' = \langle y_{k-1}\rangle'$ in every extending representation (see Fig. 4.3b). Again, $\langle y_{k-1}\rangle'$ has to be placed between $\langle x_{k-1}\rangle$ and $\langle z_{k-1}\rangle'$. We assume that $\langle x_{k-1}\rangle$ is pre-drawn on the right of $\langle y_{k-1}\rangle'$ and repeat the same argument for $C(y_k)$ and the MPQ-tree restricted to these vertices. The role of x_k, y_k and z_k is played by x_{k-1}, y_{k-1} and z_{k-1}, respectively. (The ordering of the pre-drawn intervals is flipped.)

The paragraphs above show the induction step of our proof (by induction on, say, the number of considered sections of Q). By the induction hypothesis, we find a $(k-1)$-FAT obstruction. By making x_{k-1} free and adding x_k, t_k and P_k, we get a k-FAT obstruction in the original partial representation. Clearly t_k is adjacent to the entire $(k-1)$-FAT obstruction with the exception of x_{k-1}, since all further vertices are contained in a section to the left of $s^{\leftarrow}_{x_{k-1}}(Q)$. The reason is that we always use shortest paths which are Q-monotone by Lemma 4.2.3. By the same reason, they are non-adjacent to the inner vertices of P_k and to x_k, as required.

To make the argument complete, we should check that all the assumptions used throughout the proof apply recursively, in particular the arguments concerning non-universality of t_{k-1} and reversing big components. This is true because both components $C(y_{k-1})$ and $C(z_{k-1})$ of $C(y_k) \setminus W_{k-1}$ appear to the left of x_{k-1}, so t_k and the other vertices of W_k are universal for them. This property is preserved throughout the recursion, so $C(y_\ell)$ and $C(z_\ell)$ are adjacent to all vertices of $W_k, W_{k-1}, \dots, W_{\ell+1}$. Similarly, the rest of the inductive proof can be formalized. $\square$

The above proof shows that the structure of a Q-node can be highly complicated, leading to complicated obstructions such as k-FAT. Actually, k-FAT Lemma 4.6.3 is a very useful tool because it can be also applied in situations where not all x_k, y_k, and z_k are pre-drawn, to build other obstructions. Fig. 4.17 shows an example.

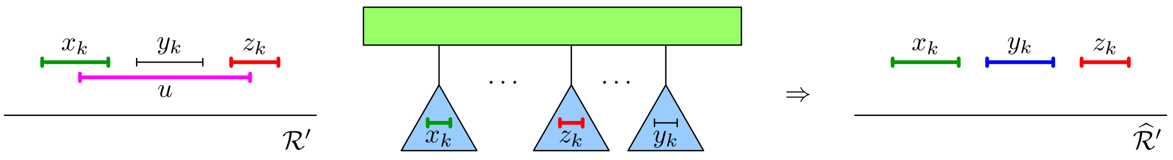


Figure 4.17: Suppose that we show that a partial representation $\mathcal{R}'$ has three pre-drawn intervals as on the left, and that there is a vertex y_k adjacent to u and non-adjacent to x_k and z_k. Then $\langle y_k \rangle$ has to be placed between $\langle x_k \rangle'$ and $\langle z_k \rangle'$ in every extending representation. Thus, we can assume it is pre-drawn there and obtain a modified partial representation $\widehat{\mathcal{R}}'$. If we further show that x_k, y_k and z_k are placed in appropriate sections of $G[Q]$ for some Q-node Q, we can apply k-FAT Lemma 4.6.3 and we get a k-FAT obstruction in $G[Q]$ and $\widehat{\mathcal{R}}'[\{x_k, y_k, z_k\}]$. Together with $\langle u \rangle'$, this forms a k-BI obstruction in G and $\mathcal{R}'$.

Lemma 4.6.4. *Consider a k-FAT obstruction H_k for $k > 2$. If we swap the positions of $\langle x_k \rangle'$ and $\langle y_k \rangle'$, then we obtain a new obstruction which contains a 1-FAT obstruction for $x'_1 = y_k$, $y'_1 = x_k$, and $z'_1 = z_k$. Further, if $k = 2$ and this does not happen, then x_2 is adjacent to t_2.*

Proof. For $k \geq 3$, the graph $H_k \setminus N[x_k]$ is connected; in particular, there exists a path $P'_1 = y_k t_{k-1} z_k$ avoiding $N[x_k]$. For $k = 2$, there exists the path $P'_1 = y_2 t_2 z_2$ avoiding $N[x_2]$, unless x_2 is adjacent to t_2. □

(k,ℓ)-CE Lemma. Suppose that we have the situation in Fig. 4.18. We can easily show that it yields to a (k,ℓ)-CE obstruction:

Lemma 4.6.5 ((k,ℓ)-CE). *Let Q be a Q-node with children $T_1, \dots, T_n$, and let a, b and c be three cliques of $T[Q]$ contained respectively in T_α, T_β and T_γ, for $\alpha < \beta < \gamma$. Let $x_k \in a$, $y_k \in c$ and $z_k \in P(b)$ be three non-adjacent vertices having a common pre-drawn neighbor u such that $\langle u \rangle'$ single overlaps $\langle z_k \rangle'$. Then $G[Q] \cup \{u\}$ and $\mathcal{R}'[\{z_k, u\}]$ contain a (k,ℓ)-CE obstruction, where either $\ell = 1$ or $k = \ell = 2$.*

Proof. By applying k-FAT Lemma 4.6.3 twice, once when $\langle x_k \rangle$ is on the left of $\langle y_k \rangle$ and once when it is on the right, we obtain the (k,ℓ)-CE obstruction. Further, by Lemma 4.6.4, we get that either $\ell = 1$, or $k = \ell = 2$. □

The proof of the following lemma reveals the structure of the minimal (k,ℓ)-CE obstructions in detail:

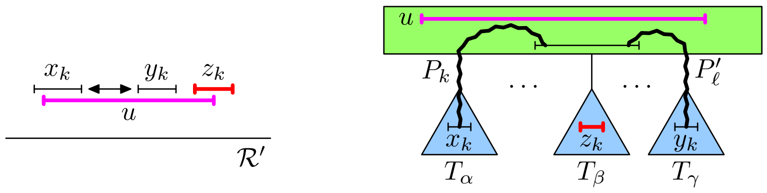


Figure 4.18: When $\langle u \rangle'$ single overlaps $\langle z_k \rangle'$, and the vertices x_k, y_k, and z_k are placed in the MPQ-tree as on the right, we get a (k,ℓ)-CE obstruction.

Lemma 4.6.6. *For $2 \geq k \geq \ell$, the list of minimal (k,ℓ)-CE obstructions is given in Fig. 4.6. For $k \geq 3$, the minimal (k,ℓ)-CE obstructions have $\ell = 1$ and consist of the graph H_k together with a vertex u, either adjacent to all vertices of H_k, or $u = t_k$.*

Proof. The simplest case is when there exist a path P_k from x_k to z_k avoiding $N[y_k]$, and a path P'_ℓ from y_k to z_k avoiding $N[x_k]$. Let P_k and P'_ℓ be shortest such paths as in Fig. 4.18, right. We get a $(1,1)$-CE obstruction. By Lemma 4.2.3, the paths P_k and P'_ℓ are monotone. Therefore, their inner vertices are non-adjacent to each other, with the possible exception of the last vertices before z_k, which can be adjacent or even identical. Concerning minimality, we can always find one of the three finite $(1,1)$-CE obstructions depicted in Fig. 4.6a. The reason is that when paths P_k and P'_ℓ are long, we can take as x_k and y_k one of their inner vertices, making them shorter.

Suppose next that there exists no path P_k from x_k to z_k avoiding $N[y_k]$. Assuming that $\langle x_k \rangle$ is placed on the left of $\langle y_k \rangle$, we apply k-FAT Lemma 4.6.3 and we get the subgraph H_k of a k-FAT obstruction (which is not the complete k-FAT obstruction because x_k and y_k are free). Let $C(x_k)$, W_k, and t_k be defined as in the proof of k-FAT Lemma 4.6.3.

Case 1: There exists some path P'_ℓ from y_k to z_k avoiding $N[x_k]$. Let P'_ℓ be a shortest such path (notice that $\ell = 1$). Together with the above subgraph H_k, we get a $(k,1)$-CE obstruction; see Fig. 4.19a. In particular, if some vertex $w \in W_k$ is non-adjacent to x_k (possibly $w = t_k$), we can use $P'_\ell = y_k w z_k$.

We note that when $k \geq 3$, such a path P'_ℓ necessarily exists, as we can use $P'_\ell = y_k t_{k-1} z_k$, as argued in Lemma 4.6.4. Therefore, the $(k,1)$-CE obstructions consist of the subgraph H_k together with u; assuming minimality, we have that either u is adjacent to all vertices of H_k, or $u = t_k$. If $k = 2$, then P'_ℓ might still exist but it might be longer and might use inner vertices not contained in H_k. Concerning minimality, we always find one of the three $(2,1)$-CE obstructions depicted in Fig. 4.6b. Indeed, P_2 can be assumed to be of length one or two, since otherwise we could use one of its inner vertices as x_2. For length two, we get $P'_1 = y_k t_k z_k$. For length one, we get a path P'_1 from z_2 to y_2, and we can assume that y_2 is adjacent to x_1 (otherwise we could use as y_2 the neighbor of x_1 on P_1).

Case 2: No such path P'_ℓ exists. By Lemma 4.6.4, necessarily $k = 2$. We want to show that there exists a $(2,2)$-CE obstruction which we describe in detail.

Notice that all vertices $w \in W_k$ are adjacent to x_k, y_k, and z_k, since otherwise

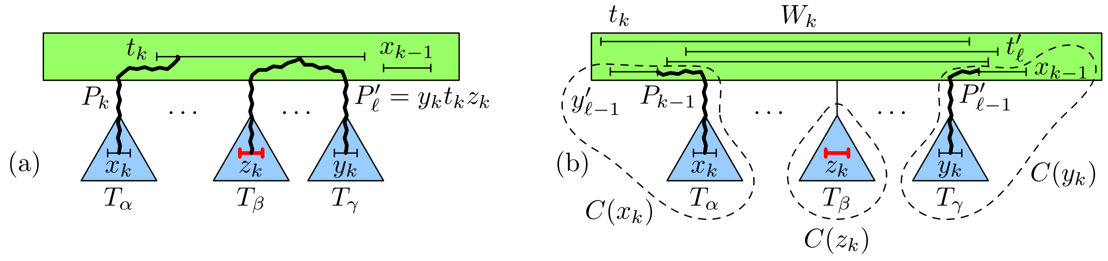


Figure 4.19: (a) Case 1: If there exists a path P'_ℓ from y_k to z_k avoiding $N[x_k]$, then we get a $(k,1)$-CE obstruction. (b) Case 2: We get a $(2,2)$-CE obstruction.

there would exist a path $P'_\ell = y_k w z_k$ avoiding $N[x_k]$. Hence the vertices of W_k belong to sections of Q, covering all subtrees between T_α and T_γ; see Fig. 4.19b. Let $C(y_k)$ and $C(z_k)$ be the components of $G[Q] \setminus W_k$ containing y_k and z_k, respectively. Since there exists no path P'_ℓ, we obtain that $C(x_k)$, $C(y_k)$, and $C(z_k)$ are pairwise different. To determine the structure of a $(2,2)$-CE obstruction, we apply the argument from Case 1 of the proof of k-FAT Lemma 4.6.3 symmetrically twice.

Let t_k be a vertex of W_k having leftmost section $s^{\rightarrow}_{t_k}(Q)$ and let t'_ℓ be a vertex of W_k having rightmost section $s^{\leftarrow}_{t'_\ell}(Q)$ (possibly $t_k = t'_\ell$). It is easy to see that $C(z_k)$ is small, otherwise we could flip it and obtain an ordering of the maximal cliques not compatible with the Q-node.

Similarly as in the proof of k-FAT Lemma 4.6.3, this implies that both $C(x_k)$ and $C(y_k)$ are big. Therefore, t_k is not universal for $C(y_k)$ and t'_ℓ is not universal for $C(x_k)$. As in the proof of k-FAT Lemma 4.6.3, we choose $x_{k-1} \in C(y_k)$ non-adjacent to t_k and $y'_{\ell-1} \in C(x_k)$ non-adjacent to t'_ℓ. There exist paths P_{k-1} from x_{k-1} to y_k and $P'_{\ell-1}$ from $y'_{\ell-1}$ to x_k. In consequence, we obtain a $(2,2)$-CE obstruction.

Regarding minimality, notice that we can assume that y_2 is adjacent to x_1, and x_2 is adjacent to y'_1; otherwise, we could choose as y_2 and x_2 the neighbors of x_1 and y'_1 on the paths P_1 and P'_1, respectively. We get the four minimal finite $(2,2)$-CE obstructions that are illustrated in Fig. 4.6c. $\square$

4.6.3 Cliques in Three Different Subtrees

When a Q-node Q is obstructed by three maximal cliques $a \in T_\alpha$, $b \in T_\beta$ and T_γ, where $\alpha < \beta < \gamma$, the situation is quite complex. Fig. 4.20 gives an overview of the cases and obstructions obtained in this case.

Lemma 4.6.7. *Without loss of generality, we can assume that $a \lhd b \rhd c$ and $\curvearrowright(a) \le \curvearrowright(c)$.*

Proof. Since a, b and c create an obstruction, b is either a minimal or a maximal element in $\lhd|_{\{a,b,c\}}$. Without loss of generality (using the flip operation), we can assume that b is maximal, so $a \lhd b \rhd c$. Since we can swap a and c by reversing the Q-node, we can assume that $\curvearrowright(a) \le \curvearrowright(c)$. $\square$

Since $a \lhd b \rhd c$, both I_a and I_c appear on the left of I_b. Since $\curvearrowright(a) \le \curvearrowright(c)$, either I_a contained in I_c, or I_a is on the left of I_c. The first case is easier:

Lemma 4.6.8. *If I_a is contained in I_c, then G and $\mathcal{R}'$ contain a (k,ℓ)-CE obstruction, where $\ell = 1$ or $2 \ge k \ge \ell$.*

Proof. The proof is illustrated in Fig. 4.21. By Lemma 3.2.4, $P(c) \subseteq P(a)$. Since b is placed between a and c in the Q-node Q, every vertex contained in both a and c is contained in b as well. Hence $P(c) \subsetneq P(b)$, and there exists $r \in P(b) \setminus P(c)$. Since $\langle r \rangle'$ is on the right of I_c, it is also on the right of I_a, and thus $r \notin P(a)$.

Let $u \in P^{\leftarrow\!\!\!\dashv}(c)$. We apply Sliding Lemma 4.2.5 to I_c, I_b, and $\langle r \rangle'$. We get a predrawn interval $\langle z_k \rangle'$ covering $r(u)$, and an induced path P_{r,z_k} from r to z_k consisting

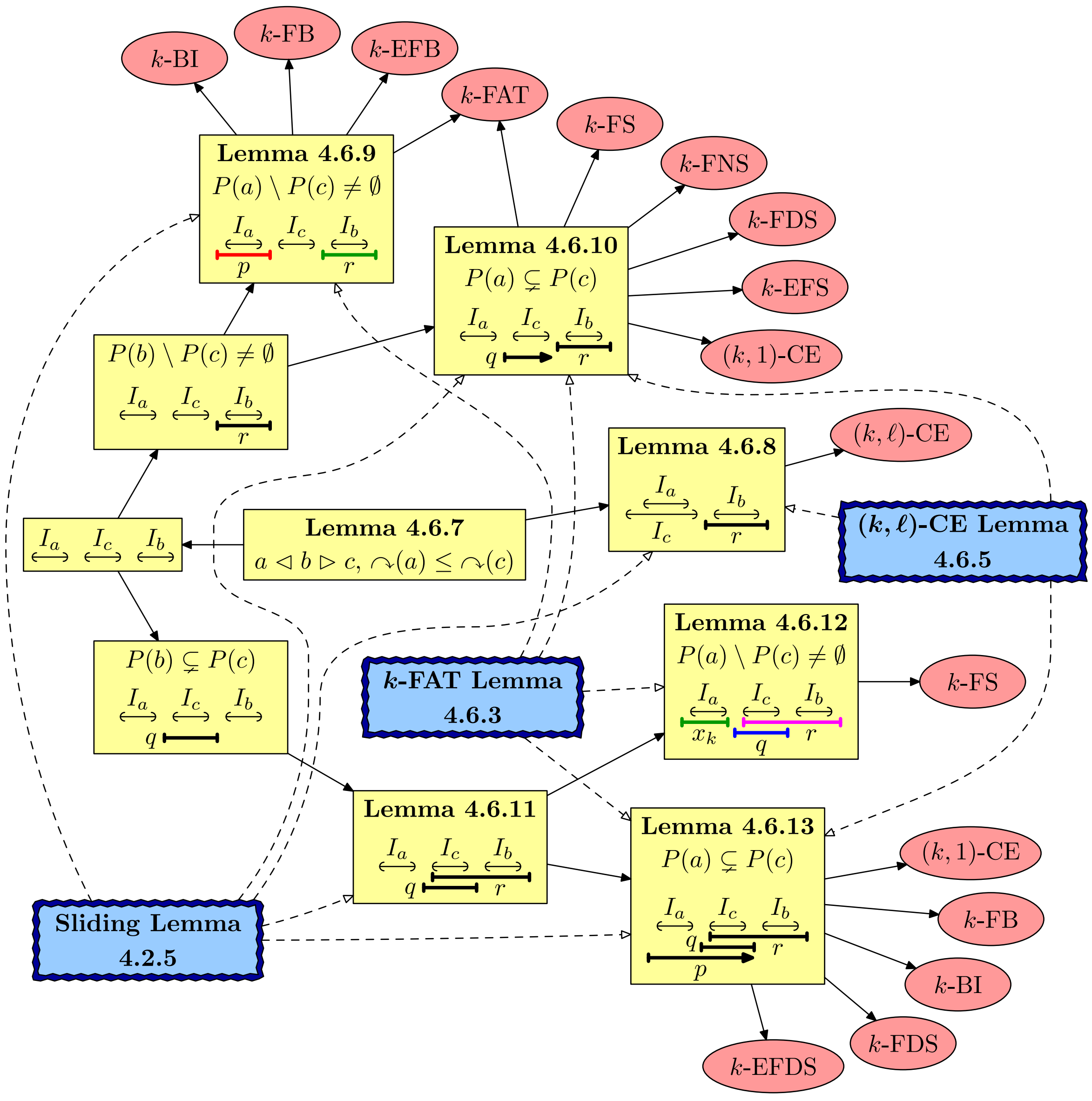


Figure 4.20: A summary of Section 4.6.3. The diagram starts in the middle with Lemma 4.6.7. Inside the cases, we draw the positions of I_a, I_b, I_c, and some pre-drawn intervals. An arrow at a pre-drawn interval means that it may be further stretched in the given direction. The obtained obstructions are highlighted in gray, the used tools have highlighted borders.

of pre-drawn intervals not in $P(c)$. Therefore z_k is on the left of c in Q. Since all pre-drawn intervals of P_{r,z_k} do not belong to $P(c)$, they are on the right of I_c. Thus they are also on the right of I_a, which implies that they do not belong to $P(a)$. Consequently, z_k is between a and c in Q.

Let $x_k \in a$ and $y_k \in c$ be vertices non-adjacent to z_k. By (k,ℓ)-CE Lemma 4.6.5, x_k, y_k, z_k, and u create a (k,ℓ)-CE obstruction, for $\ell = 1$ or $2 \geq k \geq \ell$. Notice that the clique associated to z_k is some $b' \neq b$. $\square$

The case where I_a is on the left of I_c is further divided into several subcases. In

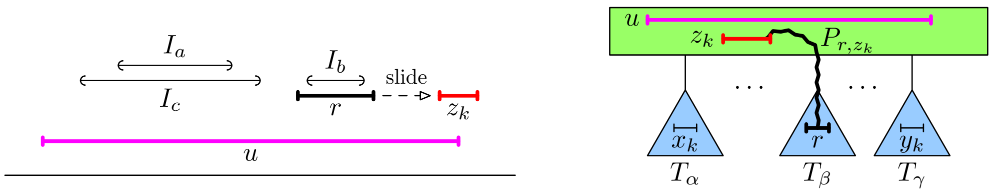


Figure 4.21: Proof of Lemma 4.6.8. On the left, the pre-drawn intervals. On the right, their positions in the MPQ-tree.

the next two lemmas, we focus on the situation where $P(b) \setminus P(c) \neq \emptyset$.

Lemma 4.6.9. *If I_a is on the left of I_c, $P(b) \setminus P(c) \neq \emptyset$ and $P(a) \setminus P(c) \neq \emptyset$, then G and $\mathcal{R}'$ contain a k-FAT, k-BI ($k \leq 2$), k-FB, or k-EFB obstruction.*

Proof. The proof is illustrated in Fig. 4.22. Let $p \in P(a) \setminus P(c)$ and $r \in P(b) \setminus P(c)$. Then $\langle p \rangle'$ is on the left of I_c, and $\langle r \rangle'$ is on the right of I_c. Clearly, $\langle p \rangle'$ and $\langle r \rangle'$ are disjoint, so p appears in the Q-node on the left of r. Let $u \in P^{\hookleftarrow}(c)$ and $v \in P^{\mapsto}(c)$ (possibly $u = v$).

If $r(u) \leq r(r)$, then $z_k = r$. Obviously, z_k is between p and c in the Q-node. Otherwise, $P(c) \subsetneq P(b)$, and we apply Sliding Lemma 4.2.5 to I_c, I_b, and r. We obtain a pre-drawn interval $\langle z_k \rangle'$ not contained in $P(c)$ covering $r(u)$, and a path P_{r,z_k} whose inner vertices are pre-drawn and not contained in $P(c)$. Notice that all of these pre-drawn vertices are on the right of I_c. Therefore, these vertices are not contained in $P(a)$. In this case, we also get that z_k is between p and c in the Q-node.

Similarly, if $\ell(p) \leq \ell(v)$, then $x_k = p$. Otherwise, we use the flipped version of Sliding Lemma 4.2.5 to I_c, I_a, and p, which gives a pre-drawn interval $\langle x_k \rangle'$ not contained in $P(c)$ covering $\ell(v)$. By a similar argument, in both cases, we show that x_k is on the left of z_k in the Q-node.

Let $y_k \in c$ be a vertex non-adjacent to z_k (possibly, $y_k = u$ or $y_k = v$). Such a vertex exists because z_k is on the left of c in the Q-node. Notice that y_k is also non-adjacent to x_k. Since y_k is adjacent to u and v, in every extending representation $\langle y_k \rangle$ is between $\langle x_k \rangle'$ and $\langle z_k \rangle'$. So we can assume that it is pre-drawn in this position and, by k-FAT Lemma 4.6.3, we get a k-FAT obstruction. Together with u and v (or possibly only one of them), we get one of the obstructions in Fig. 4.23. $\square$

Lemma 4.6.10. *If I_a is on the left of I_c, $P(b) \setminus P(c) \neq \emptyset$ and $P(a) \subsetneq P(c)$, then G and $\mathcal{R}'$ contain a k-FAT, $(k,1)$-CE, k-FS, k-FDS, k-FNS, or k-EFS obstruction.*

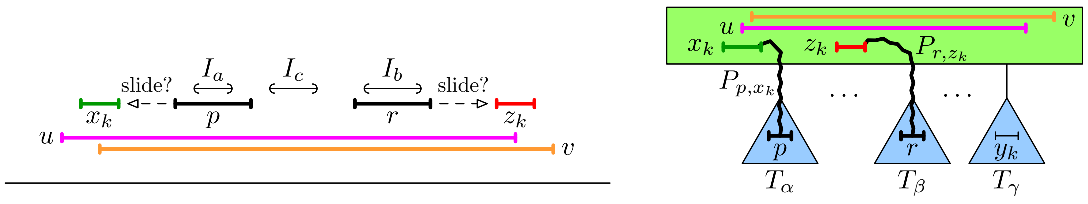


Figure 4.22: Proof of Lemma 4.6.9. On the left, the pre-drawn intervals, with possible sliding on each side. On the right, their positions in the MPQ-tree.

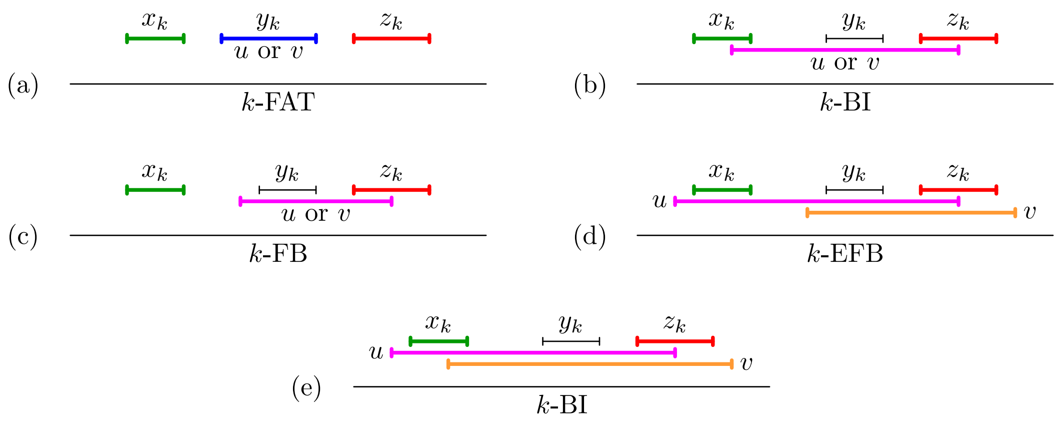


Figure 4.23: The different obstructions obtained in the proof of Lemma 4.6.9. If $\ell(x_k) \le \ell(u) \le r(u) \le r(z_k)$, we get one of the obstructions (a) to (c). Since z_k is in the Q-node between x_k and c, if u or v intersect x_k, then they also intersect z_k. In the cases (d) and (e), $\ell(u) < \ell(x_k)$ and $r(z_k) < r(v)$. Since u intersects z_k, there are only two possible obstructions.

Proof. We choose $r \in P(b) \setminus P(c)$ and $q \in P(c) \setminus P(a)$ with leftmost right endpoint. Then $\langle r \rangle'$ is on the right of I_c and $\langle q \rangle'$ is on the right of I_a. We note that q might be adjacent to r or not, and might belong to $P(b)$ or not. Since $P(a) \subsetneq P(c)$, we get from the structure of the Q-node that also $P(a) \subsetneq P(b)$. Let $u \in P^{\leftarrow\!\shortmid}(a)$. Notice that at least one of q and u belongs to $P^{\leftarrow\!\shortmid}(c)$.

Case 1: $u \in P^{\leftarrow\!\shortmid}(c)$. Then $r(u) \le r(q)$ and $P(c) \subsetneq P(b)$; the situation is depicted in Fig. 4.24a. We apply Sliding Lemma 4.2.5 to I_c, I_b, and r. We get a pre-drawn interval $z_k \notin P(c)$ covering $r(u)$, and a path P_{r,z_k} consisting of pre-drawn intervals not contained in $P(c)$. Therefore, z_k is on left of c in the Q-node. Since I_a is on the left

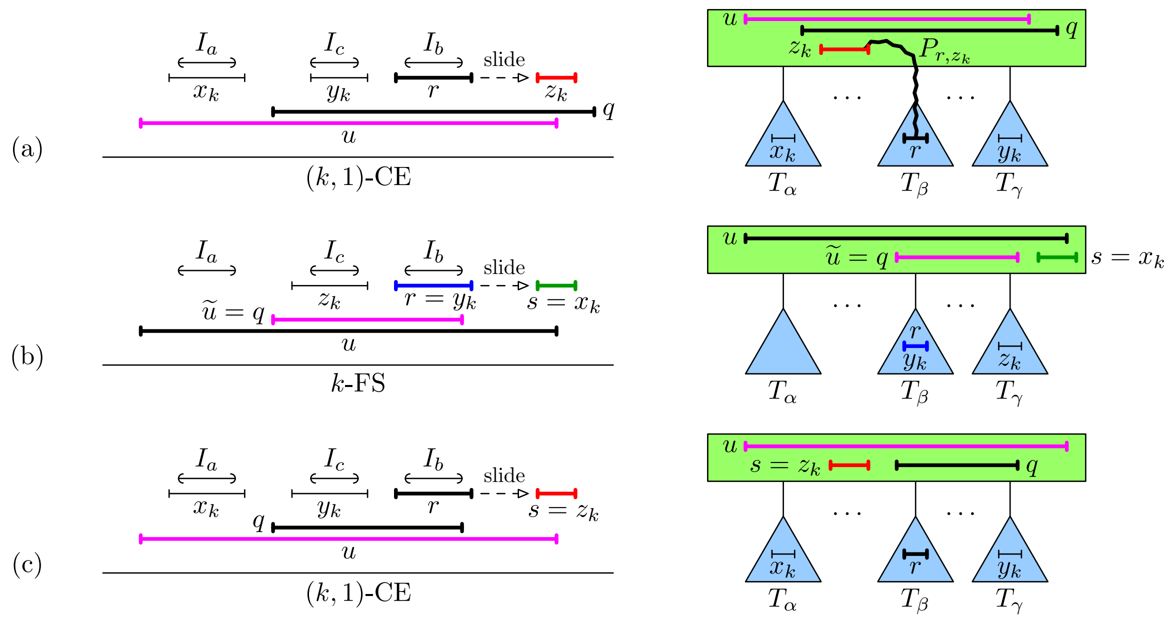


Figure 4.24: Proof of Lemma 4.6.10. On the left, the pre-drawn intervals. On the right, their positions in the MPQ-tree. (a) Case 1. (b) Subcase 2A. (c) Subcase 2B.

of I_c, all vertices of P_{r,z_k} are also not contained in $P(a)$. Thus z_k is on the right of a in the Q-node.

Choose $y_k \in c$ non-adjacent to z_k. By Lemma 4.2.4, there exists $x_k \in a$ non-adjacent to both z_k and q. Since z_k is between a and c in the Q-node, also x_k is non-adjacent to y_k. Since $y_k q z_k$ is a path avoiding $N[x_k]$, by (k,ℓ)-CE Lemma 4.6.5 we obtain a $(k,1)$-CE obstruction.

Case 2: $q \in P^{\leftarrow\!\shortmid}(c)$. Then $r(q) < r(u)$. First we argue that, without loss of generality, we can assume that either $\langle q\rangle'$ and $\langle r\rangle'$ are disjoint, or $\langle r\rangle'$ covers $r(q)$. Suppose that $\langle r\rangle'$ is contained in $\langle q\rangle'$. Since $q \in P^{\leftarrow\!\shortmid}(c)$, this implies that $P(c) \subsetneq P(b)$. By applying Sliding Lemma 4.2.5 to I_c, I_b and r, we obtain a pre-drawn interval $\widetilde{r}$ not in $P(c)$ which covers $r(q)$. We also get a path $P_{r,\widetilde{r}}$ whose vertices are pre-drawn and not contained in $P(c)$; since I_a is on the left of I_c, they are also not in $P(a)$. Therefore $\widetilde{r}$ is between a and c in the Q-node. Further, $\widetilde{r}$ belongs to some clique $\widetilde{b}$ for which $I_{\widetilde{b}}$ is on the right of I_c. From now on, we work with $\widetilde{r}$ as r, and with $\widetilde{b}$ as b. Hence the assumption on the relative positions of $\langle q\rangle'$ and $\langle r\rangle'$ holds.

We apply Sliding Lemma 4.2.5 to I_a, I_b and r, and we get a pre-drawn interval $s \notin P(a)$ covering $r(u)$. This sliding is weaker that in Case 1: we know that s is on the right of a, but we do not know its position with respect to c. We distinguish three subcases according to the relative positions of q and s in the Q-node.

Subcase 2A: s is on the right of q. The situation is depicted in Fig. 4.24b. Let $x_k = s$ and $y_k = r$. If $\langle q\rangle'$ is on the left of $\langle r\rangle'$, let $z_k = q$. Otherwise, let $z_k \in c$ be a vertex non-adjacent to r, but possibly adjacent to x_k. Since in every extending representation z_k is placed on the left of y_k, we can apply k-FAT Lemma 4.6.3 to x_k, y_k and z_k, and get a subgraph H_k. If $\langle q\rangle'$ is on the left of $\langle r\rangle'$, then H_k gives a k-FAT obstruction. If $\langle r\rangle'$ covers $r(q)$, then H_k together with $\widetilde{u} = q$ gives a k-FS obstruction.

Subcase 2B: s is on the left of q. We choose $x_k \in a$ and $y_k \in c$ non-adjacent to s; such vertices exist because s is between a and c in the Q-node. By (k,ℓ)-CE Lemma 4.6.5, we get a (k,ℓ)-CE obstruction for x_k, y_k, $z_k = s$ and u. Notice that we can construct a path P_{y_k,z_k} from y_k to z_k avoiding $N[x_k]$ by applying Sliding Lemma 4.2.5 to I_a, I_c, and q. Thus $\ell = 1$.

Subcase 2C: $\langle s\rangle'$ intersects $\langle q\rangle'$. Notice that $\langle s\rangle'$ also intersects $\langle r\rangle'$. Therefore, if $s \notin P(c)$, then it appears in the Q-node between a and c. Let $z_k = s$, we get a $(k,1)$-CE obstruction as follows. We choose $y_k \in c$ non-adjacent to z_k. By Lemma 4.2.4, there exists $x_k \in a$ non-adjacent to q, y_k, and z_k. By (k,ℓ)-CE Lemma 4.6.5, we get a $(k,1)$-CE obstruction for x_k, y_k, z_k and u as illustrated in Fig. 4.25a; notice that the path $y_k q z_k$ avoids $N[x_k]$.

It remains to deal with the situation when $s \in P(c)$. Let $z_k = r$. If $\langle q\rangle'$ intersects $\langle r\rangle'$, let $y_k \in c$ be a vertex non-adjacent to r; otherwise let $y_k = q$. By Lemma 4.2.4, there exists $x_k \in a$ non-adjacent to q, y_k, and z_k. In every extending representation, $\langle y_k\rangle$ is placed on the left of $\langle z_k\rangle'$, and $\langle x_k\rangle$ is placed on the left of $\langle y_k\rangle$. Therefore, by k-FAT Lemma 4.6.3, we get a subgraph H_k of a k-FAT obstruction. Together with u, $v = s$, $w = q$ (for $y_k \neq q$), or possibly some of them, we get a k-FDS, k-EFS, or k-FNS obstruction; see Fig. 4.25b, c, and d. $\square$

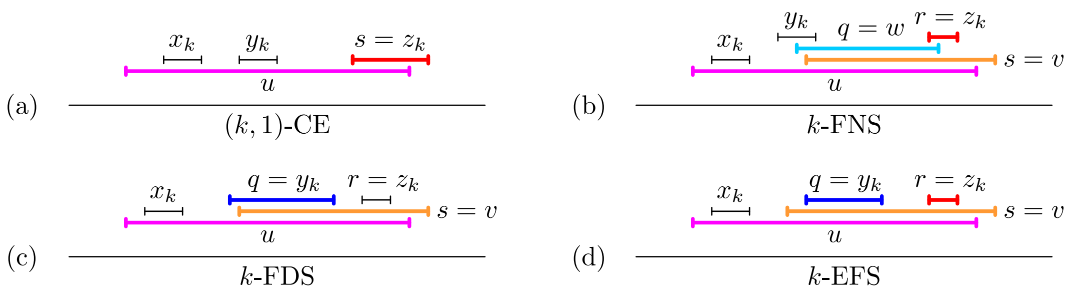


Figure 4.25: Four possible obstructions obtained in Subcase 2C of the proof of Lemma 4.6.10. (a) If $s \notin P(c)$, we get a $(k,1)$-CE obstruction. (b) If $\langle q\rangle'$ intersects $\langle r\rangle'$, we get a k-FNS obstruction. Recall that the relative order of $\ell(q)$ and $\ell(s)$ does not matter. (c) If $\langle q\rangle'$ is on the left of $\langle r\rangle'$ and $\ell(q) \le \ell(s)$, we get a k-FDS obstruction. (d) If $\langle q\rangle'$ is on the left of $\langle r\rangle'$ and $\ell(s) < \ell(q)$, we get a k-EFS obstruction.

The case where $P(b) \subsetneq P(c)$ is addressed in Lemmas 4.6.12 and 4.6.13. First, we need an auxiliary result.

Lemma 4.6.11. *If I_a is on the left of I_c and $P(b) \subsetneq P(c)$, there exist $q \in P(c) \setminus P(b)$ and $r \in P(b) \setminus P(a)$ such that $\langle q\rangle'$ is on the right of I_a and on the left of I_b, and $\langle r\rangle'$ is on the right of I_a, containing I_c and I_b. Without loss of generality, $\langle q\rangle'$ covers $\ell(r)$.*

Proof. The proof is depicted in Fig. 4.26. Clearly, there exists $q \in P(c) \setminus P(b)$. Due to the structure of the Q-node, we also have that $q \notin P(a)$. Therefore, $\langle q\rangle'$ is between I_a and I_b.

Next, we argue that there exists $r \in P(b) \setminus P(a)$. For contradiction, assume that $P(b) \subsetneq P(a)$. Let $v \in P^{\mapsto}(b)$; notice that v contains I_a and I_c. By the flipped version of Sliding Lemma 4.2.5 applied to I_c, I_b and q, there exists a path consisting of pre-drawn intervals not contained in $P(b)$ from q to z, where $\langle z\rangle'$ covers $\ell(v)$. At least one interval of this path intersects I_a, so it belongs to $P(a)$. This contradicts the fact that b is between a and c in the Q-node. Hence, there exists $r \in P(b) \setminus P(a)$.

We choose r having rightmost left endpoint. Clearly, $\langle r\rangle'$ is on the right of I_a, and contains I_b and I_c. Suppose that $\ell(r) < \ell(q)$. Since r has rightmost left endpoint among all intervals in $P(b) \setminus P(a)$, and every interval in $P(a)$ has its left endpoint more to the left, we obtain that $r \in P^{\mapsto}(b)$. Therefore, we can apply the flipped version of Sliding Lemma 4.2.5 to I_c, I_b and q. We get a pre-drawn interval $\widetilde{q} \notin P(b)$ covering $\ell(r)$, and a path $P_{q,\widetilde{q}}$ from q to $\widetilde{q}$ whose vertices are not in b. Therefore, $\widetilde{q}$ is on the right of b in the Q-node. Let $\widetilde{c}$ be a maximal clique containing $\widetilde{q}$. Since $I_{\widetilde{c}}$ is contained

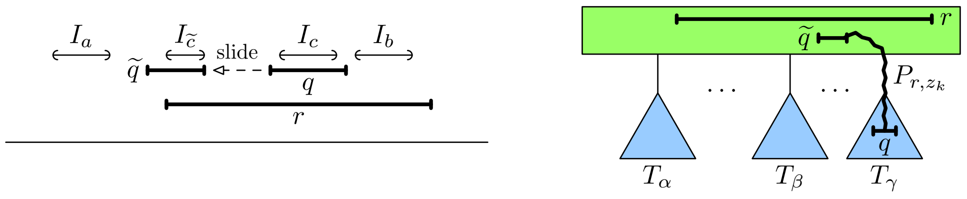


Figure 4.26: Proof of Lemma 4.6.11. On the left, the pre-drawn intervals. On the right, their positions in the MPQ-tree.

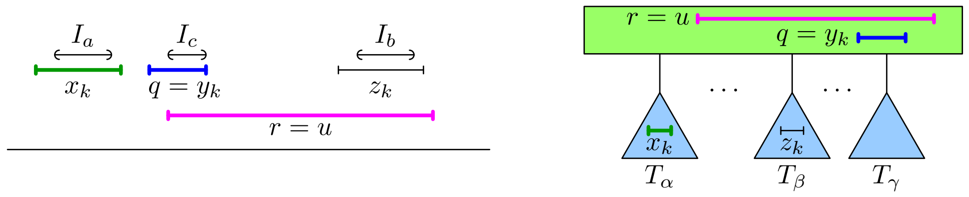


Figure 4.27: Proof of Lemma 4.6.12. We derive that $\langle x_k\rangle'$ is on the left of $\langle q\rangle'$, which gives a k-FS obstruction based on the positions in the Q-node.

in $\widetilde{q}$, it is between I_a and I_b. Therefore, we can work with $\widetilde{q}$ and $\widetilde{c}$ instead of q and c. Thus we can assume that $\langle q\rangle'$ covers $\ell(r)$. ☐

For $P(b) \subsetneq P(c)$, we distinguish two cases.

Lemma 4.6.12. *If I_a is on the left of I_c, $P(b) \subsetneq P(c)$, and $P(a) \setminus P(c) \neq \emptyset$, then G and $\mathcal{R}'$ contain a k-FS obstruction.*

Proof. The proof is illustrated in Fig. 4.27. By Lemma 4.6.11, there exist $q \in P(c) \setminus P(b)$ and $r \in P(b) \setminus P(a)$ such that $\langle q\rangle'$ covers $\ell(r)$. Let $x_k \in P(a) \setminus P(c)$ and $y_k = q$. Then $\langle x_k\rangle'$ is on the left of I_c, and therefore also on the left of I_b. Thus $x_k \notin P(b)$. We infer that x_k is on the left of b in the Q-node, so it is non-adjacent to y_k. In consequence, $\langle x_k\rangle'$ is on the left of $\langle y_k\rangle'$. Let $z_k \in b$ be a vertex non-adjacent to y_k.

If z_k is adjacent to x_k, we get a 1-FS obstruction. Otherwise, in every extending representation, $\langle z_k\rangle$ is to the right of $\langle y_k\rangle'$. By k-FAT Lemma 4.6.3, we get a subgraph H_k of a k-FAT obstruction. Together with $u = r$, this leads to a k-FS obstruction. ☐

Lemma 4.6.13. *If I_a is on the left of I_c, $P(b) \subsetneq P(c)$ and $P(a) \subsetneq P(c)$, then G and $\mathcal{R}'$ contain a $(k,1)$-CE, k-FB, k-BI, k-FDS, or k-EFDS obstruction.*

Proof. Let $p \in P^{\leftarrow\!\!\dashv}(a)$, and q be the vertex from Lemma 4.6.11. By applying Sliding Lemma 4.2.5 to I_a, I_c and q, we get a pre-drawn interval $s \notin P(a)$ covering $r(p)$, and path $P_{q,s}$ of intervals not in $P(a)$, so s appears on the right of a in the Q-node. Similarly, as in Case 2 of the proof of Lemma 4.6.10, we distinguish three cases according to the relative positions of s and q in the Q-node; see Fig 4.28.

Case 1: s is on the left of q. By Lemma 4.2.4, there exists $x_k \in a$ non-adjacent to all vertices of $P_{q,s}$, in particular non-adjacent to s and q. Let $y_k = q$, $z_k = s$, and $u = p$. Clearly, z_k is between x_k and y_k in the Q-node. By (k,ℓ)-CE Lemma 4.6.5 and the existence of P_{y_k,z_k}, we get a $(k,1)$-CE obstruction. Notice that $\langle y_k\rangle'$ can be made free; see Fig. 4.28a.

Case 2: s is on the right of q. Since q is between b and s in the Q-node, we get that $s \notin P(b)$. Let $x_k = s$, $z_k = q$, and $u = r$, where r is the vertex from Lemma 4.6.11. There exists $y_k \in b$ non-adjacent to z_k and, by the structure of the Q-node, also non-adjacent to x_k. Since y_k is adjacent to p and r, $\langle y_k\rangle$ is between $\langle x_k\rangle'$ and $\langle z_k\rangle'$ in every extending representation. By k-FAT Lemma 4.6.3, we get a subgraph H_k of a k-FAT obstruction. If $r(r) \le r(x_k)$, together with u, we obtain a k-FB or a k-BI obstruction. If $r(r) > r(x_k)$, together with u and $v = p$, we obtain a k-BI obstruction; see Fig. 4.28b.

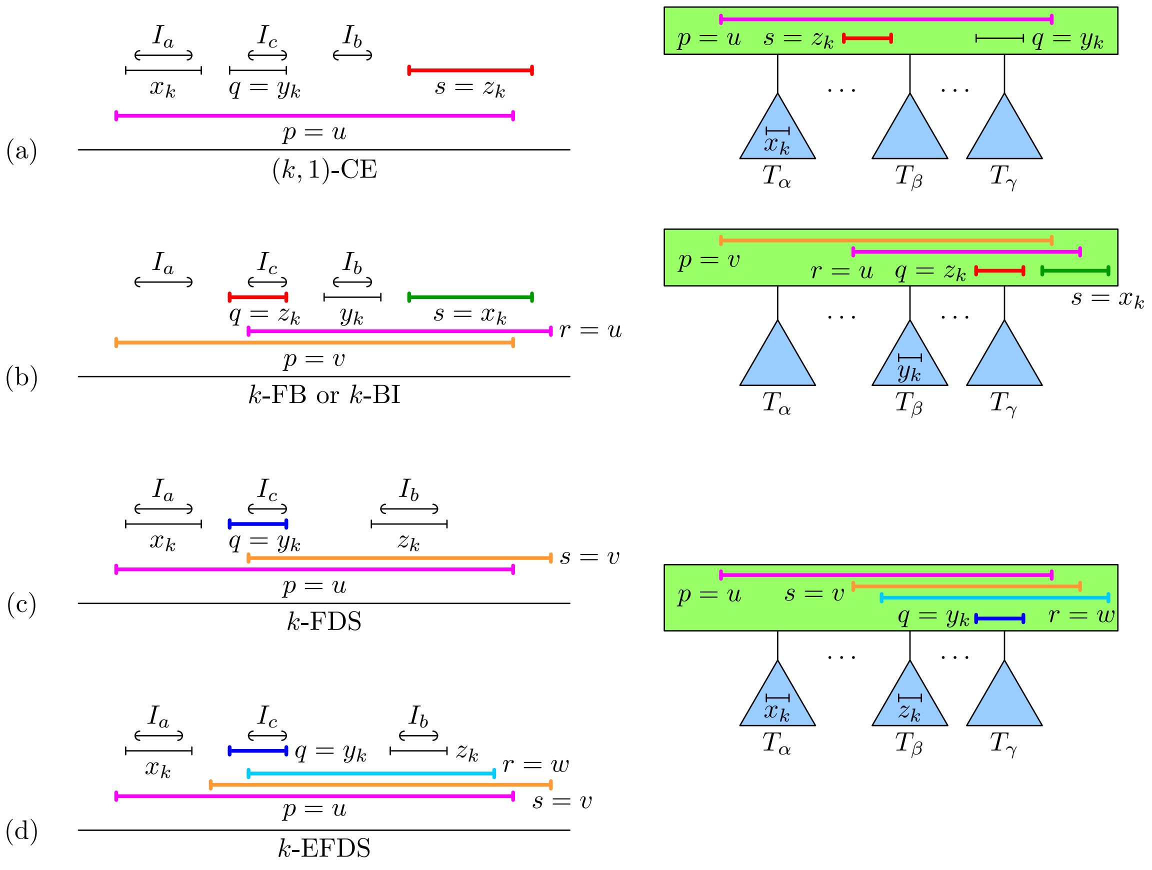


Figure 4.28: Proof of Lemma 4.6.13. On the left, the pre-drawn intervals. On the right, their positions in the MPQ-tree. (a) Case 1. (b) Case 2. (c) Case 3, if $\ell(y_k) \le \ell(s)$. (d) Case 3, if $\ell(y_k) > \ell(s)$.

Case 3: $\langle s \rangle'$ intersects $\langle q \rangle'$. Since s contains I_b, it belongs to $P(b)$. Let $y_k = q$, $u = p$, $v = s$, and $w = r$; we note that possibly $s = r$. By Lemma 4.2.4, there exists $x_k \in a$ non-adjacent to y_k, v, and w. Since x_k is adjacent to u, then $\langle x_k \rangle$ is on the left of $\langle y_k \rangle'$ in every extending representation. Finally, there exists $z_k \in b$ non-adjacent to y_k. Since z_k is adjacent to u, v, and w, we have that $\langle z_k \rangle$ is on the right of $\langle y_k \rangle'$ in every extending representation.

Since z_k is between x_k and y_k in the Q-node, we can apply k-FAT Lemma 4.6.3, which gives a subgraph H_k of a k-FAT obstruction. If $\ell(y_k) \le \ell(v)$, together with u and v, we obtain a k-FDS obstruction; see Fig. 4.28c. If $\ell(y_k) > \ell(v)$, together with u, v, and w, we get a k-EFDS obstruction; see Fig. 4.28d. □

In summary, we conclude:

Lemma 4.6.14 (The Q-node case, Three Subtrees)**.** *If the three cliques creating the obstruction belong to three different subtrees, then G and $\mathcal{R}'$ contain a k-FAT, k-BI ($k \le 2$), k-FS, k-EFS, k-FB, k-EFB, k-FDS, k-EFDS, k-FNS, or (k, ℓ)-CE obstruction (either $k = \ell = 2$, or $k \ge \ell = 1$).*

Proof. For an overview, see the diagram in Fig. 4.20. The proof follows from Lemmas 4.6.7, 4.6.8, 4.6.9, 4.6.10, 4.6.11, 4.6.12, and 4.6.13. □

4.7 Proofs of the Main Results

Now, we are ready to put all results together to prove two main results for partial representation extension of interval graphs.

Characterization of Minimal Obstructions. We are ready to prove Theorem 2.2.1 which states that a partial representation $\mathcal{R}'$ of G is extendible if and only if G and $\mathcal{R}'$ contain none of the obstructions described in Section 4.1.

Proof of Theorem 2.2.1. If G and $\mathcal{R}'$ contain one of the obstructions, they are non-extendible by Lemma 4.1.2. It remains to prove the converse. If G is not an interval graph, it contains an LB obstruction [249]. Otherwise, G is an interval graph and there exists an MPQ-tree T for it. By Corollary 3.4.2, we know that a partial representation $\mathcal{R}'$ is extendible if and only if T can be reordered according to $\vartriangleleft$. If it cannot be reordered, then the reordering algorithm fails in some node of T. If this reordering fails in a leaf, we get a 1-BI obstruction by Lemma 4.4.1. If it fails in a P-node, we get an SE, 1-BI, or 1-FAT obstruction by Lemma 4.5.3. And if it fails in a Q-node, we get one of the obstructions of Section 4.1 by Lemmas 4.6.1, 4.6.2, and 4.6.14. □

Next, we show that a partial representation $\mathcal{R}'$ is extendible if and only if every quadruple of pre-drawn intervals is extendible by itself.

Proof of Corollary 2.2.2. The result follows from the fact that all the obstructions of Theorem 2.2.1 contain at most four pre-drawn intervals. □

Linear-time Certifying Algorithm. Next, we modify the linear-time algorithm of Proposition 3.4.3 to also certify non-extendible partial representations by described minimal obstructions. We first show that k-FAT obstructions can be found in linear time:

Lemma 4.7.1. *Suppose that the assumptions of k-FAT Lemma 4.6.3 are satisfied. Then we can find a k-FAT obstruction in time $\mathcal{O}(n+m)$.*

Proof. Since the proof of k-FAT Lemma 4.6.3 is constructive, the algorithm follows it. Let Q be the Q-node. We search the graph $G[Q] \setminus N[y_k]$ from x_k to compute $C(x_k)$, and test whether z_k belongs to it. If it does, the algorithm stops and outputs 1-FAT. Otherwise, we compute W_k, choose t_k, and store it together with P_k. We choose x_{k-1} as in the the proof; if $s_i(Q) = s^{\rightarrow}_{t_k}(Q)$, then either $s^{\leftarrow}_{x_{k-1}}(Q) = s_{i+1}(Q)$, or x_{k-1} belongs to sections of T_{i+1}. Then we apply the rest of the algorithm recursively. It is important that then we can remove $C(x_k)$ and W_k from the graph because they are not used in the remainder of the obstruction.

Since the algorithm searches each vertex and edge of $G[Q] \setminus N[y_k]$ at most once when computing $C(x_j)$, we obtain that the algorithm runs in time $\mathcal{O}(n+m)$. □

Similarly, a (k,ℓ)-CE obstruction can be obtained from (k,ℓ)-CE Lemma 4.6.5 in time $\mathcal{O}(n+m)$. Since obstructions are built constructively, we can modify the algorithm of Proposition 3.4.3 to get a linear-time certifying algorithm for the partial representation extension problem:

Proof of Theorem 2.2.3. We can assume that G is an interval graph; otherwise we can find an LB obstruction in time $\mathcal{O}(n+m)$ using [255]. We recall the algorithm of Proposition 3.4.3. We compute the MPQ-tree T and the interval ordering $\lhd$, defined by the partial representation $\mathcal{R}'$. Using Proposition 3.3.4, we test whether T can be reordered compatible with $\lhd$ in time $\mathcal{O}(n+m)$. By Corollary 3.4.2, the partial representation $\mathcal{R}'$ is extendible if and only if the reordering is possible.

The algorithm of Proposition 3.3.4 uses Lemma 3.3.7 to reorder nodes of T. If the reordering is not possible, it fails to reorder some node and the negative answer is certified by subtrees which create an obstruction (1 clique for a leaf, 2 subtrees for a P-node, and 4 subtrees for a Q-node). We distinguish three cases according to the distinct types of obstructed nodes.

Case 1: A leaf cannot be reordered. We output a 1-BI obstruction in time $\mathcal{O}(n)$, by searching the partial representation.

Case 2: A P-node P cannot be reordered. From Lemma 3.3.7, we get directly a two-cycle $T_i \lhd T_j \lhd T_i$, ensured by Lemma 4.5.1, and four maximal cliques a, b, c, and d defining it. By Lemma 4.5.2, one of these maximal cliques can be omitted, and it can be clearly found in constant time. It remains to output an SE, 1-BI, or 1-FAT obstruction in time $\mathcal{O}(n+m)$, by following Lemma 4.5.3. For 1-BI and 1-FAT obstructions, we find a shortest path in $G[P] \setminus N[y_k]$ by searching the graph.

Case 3: A Q-node Q cannot be reordered. From Lemma 3.3.7, we get four subtrees with four maximal cliques defining the obstruction and, by following Lemma 4.6.1, we can reduce it to at most three maximal cliques. An SE obstruction can be computed in time $\mathcal{O}(n+m)$. If three maximal cliques are contained in two subtrees, we follow Lemma 4.6.2 and output one of the obstructions in time $\mathcal{O}(n+m)$.

If three maximal cliques belong to three different subtrees, we follow the structure of the proof of Lemma 4.6.14. In all cases, we derive some vertices somehow placed in the Q-node and some pre-drawn intervals, which can be easily done in time $\mathcal{O}(n+m)$. Next, we either apply k-FAT Lemma, or (k,ℓ)-CE Lemma to construct the obstruction, which can be done in time $\mathcal{O}(n+m)$ by Lemma 4.7.1. □

4.8 Conclusions

In this paper, we have described the minimal obstructions that make a partial interval representation non-extendible. There are three main points following from the proof:

1. Minimal obstructions for the partial representation extension problem are much more complicated than minimal forbidden induced subgraphs of interval graphs, characterized by Lekkerkerker and Boland [249].
2. Nevertheless, it is possible to describe these obstructions using structural results derived in [215] and in this paper. We show that almost all of these obstructions

consist of three intervals x_k, y_k and z_k that are forced by the partial representation to be drawn in an incorrect left-to-right order. This incorrect placement leads to the complex zig-zag structure of a k-FAT obstruction.

3. The structure of the sections of a Q-node Q can be very intricate. Suppose that we contract in $G[Q]$ the sections of each subtree T_i into one vertex. Then we get an interval graph which has a unique interval representation up to flipping the real line. Such interval graphs have been extensively studied, see for instance [174, 125, 293]. Therefore, our structural results needed to find minimal obstructions may be of independent interest.

Structural Open Problems. The first open problem we propose is a characterization of minimal obstructions for other graph classes. We select those classes for which polynomial-time algorithms are known [58, 59, 210, 211, 212, 20, 243].

Problem 4.8.1. *What are the minimal obstructions for the partial representation extension problems of the classes* CIRCLE*,* FUN*,* PERM*,* TRAPEZOID*,* PROPER INT*,* UNIT INT*, k-*NestedINT*, and* PROPER CIRCULAR-ARC*?*

The second open problem involves bounded representations, described in Section 2.9.2.

Problem 4.8.2. *What are minimal obstructions making bounds for interval graphs unsolvable?*

Algorithmic Open Problems. We have described a linear-time certifying algorithm that can find one of the minimal obstructions in a non-extendible partial representation. There are several related computational problems, suggested by Jan Kratochvíl, for which the complexity is open:

Problem 4.8.3. *What is the computational complexity of the problem of testing whether a given minimal obstruction is contained in G and $\mathcal{R}'$?*

Since a minimal obstruction contains at most four pre-drawn intervals, we can test over all subsets of at most four pre-drawn intervals whether they form an obstruction (say, by freeing the rest of them and testing whether the modified partial representation is extendible). If k is fixed, we can test whether the subgraph of a given obstruction is contained in G. Given a triple x_k, y_k and z_k forming a k-FAT obstruction, the proof of k-FAT Lemma 4.6.3 and the algorithm of Lemma 4.7.1 constructs it while minimizing k. The approach needs to be changed to check whether they also form an ℓ-FAT obstruction, for $\ell > k$.

The next problem generalizes the partial representation extension problem.

Problem 4.8.4. *What is the computational complexity of testing whether at most ℓ pre-drawn intervals can be freed to make a partial representation extendible $\mathcal{R}'$?*

Similar problems are usually NP-complete. On the other hand, we propose the following reformulation which might lead to a polynomial-time algorithm. Every minimal obstruction contains at most four pre-drawn intervals. Let P be the set of pre-drawn intervals, and let $\mathcal{S}$ consist of all subsets of P of size at most four which form an obstruction. We can clearly compute $\mathcal{S}$ in polynomial time. Then the problem above is equivalent to finding a minimal hitting set of P and $\mathcal{S}$. This problem is in general NP-complete, but the extra structure given by the MPQ-tree might make it tractable.

Problem 4.8.5. *What is the complexity of testing whether it is possible to remove at most ℓ vertices from an interval graph G to make a partial representation extendible $\mathcal{R}'$?*

This problem is fundamentally different from Problem 4.8.4, in which the partial representation $\mathcal{R}'$ is modified. In this problem, we modify the graph G itself, changing its structure. When we remove a pre-drawn vertex, we also remove its pre-drawn interval from the partial representation. We note that the assumption that G is an interval graph is important. For general graphs G, the problem is known to be NP-complete even when $\mathcal{R}' = \emptyset$ [251].

5 Interval Graphs of Limited Nesting and Count of Lengths

This chapter contains:

- *5.1 Extending Partial Representations with Two Lengths.* We show an NP-hardness reduction for two lengths, which works even when they are known and each interval has its lenght prescribed by the input.
- *5.2 Basic Properties of k-nested Interval Graphs.* We introduce some definitions and describe an efficient encoding of k-nested interval graphs using $2n\lceil \log k + 1 \rceil$ bits. We present cleaned representations minimizing nesting for a consecutive ordering of maximal cliques and determine which nestings are forced in every interval representation.
- *5.3 Recognizing k-nested Interval Graphs.* We describe a linear-time algorithm for computing an interval representation of minimal nesting based on dynamic programming on MPQ-trees. We compute triples (α, β, γ) for each subtree of the MPQ-tree, from bottom to top, linked to minimal interval representations. Formulas for P-nodes and for Q-nodes are described, computing its triple from the triples of their subtrees.

`http://pavel.klavik.cz/orgpad/nest_len_int.html`

5.1 Extending Partial Representations with Two Lengths

The complexity of recognizing k-LengthINT is a long-standing open problem, even for $k = 2$. In this section, we show that RepExt(k-LengthINT) is NP-hard even when $k = 2$. We adapt the reduction from 3-Partition used in [213, 214, 211, 212] which is the following computational problem:

Problem:	3-Partition
Input:	Integers $A_1, \dots, A_{3s}$ and M such that $\frac{M}{2} < A_i < \frac{M}{4}$ and $\sum A_i = Ms$.
Question:	Can A_i's be split into s triples, each summing to exactly M.

This problem is strongly NP-complete [140], which means that it is NP-complete even when the input is coded in unary, i.e., all integers are of polynomial sizes.

Theorem 2.4.1. Assume (i) and (ii). For an instance of 3-Partition, the reduction constructs an interval graph G and a partial representation $\mathcal{R}'$ as depicted in Fig. 5.1. We claim that $\mathcal{R}'$ can be extended using two lengths of intervals if and only if the instance of 3-Partition is solvable. We set $a = 1$ and $b = s \cdot (M + 2) - 1$. The partial representation $\mathcal{R}'$ consists of $s + 1$ disjoint pre-drawn intervals $\langle v_0 \rangle', \dots, \langle v_s \rangle'$ of length a such that $\langle v_i \rangle' = [i \cdot (M + 2), i \cdot (M + 2) + 1]$. So they split the real line into s equal gaps of size $M + 1$ and two infinite regions.

Aside $v_0, \dots, v_s$, the graph G contains a vertex w represented by an interval of length b, adjacent to every vertex in G. Further, for each A_i, the graph $G \setminus w$ contains P_{2A_i} (a path with $2A_i$ vertices) as one component, with each vertex represented by an interval of length a.

The described reduction clearly runs in polynomial time. It remains to show that $\mathcal{R}'$ is extendible if and only if the instance of 3-Partition is solvable. The length of b implies that every extending representation has $\langle w \rangle = [1, s \cdot (M + 2)]$ to intersect both $\langle v_0 \rangle'$ and $\langle v_s \rangle'$. Therefore, each of the paths P_{2A_i} has to be placed in exactly one of the s gaps. In every representation of P_{2A_i}, it requires the space at least $A_i + \varepsilon$ for

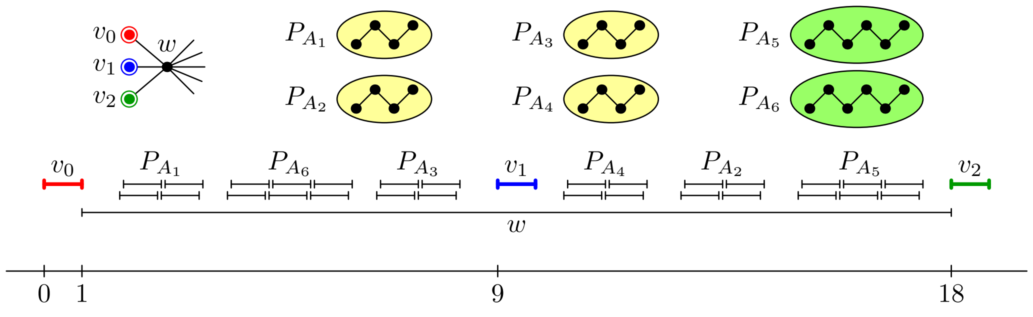


Figure 5.1: Suppose that we have the following input for 3-Partition: $s = 2$, $M = 7$, $A_1 = A_2 = A_3 = A_4 = 2$ and $A_5 = A_6 = 3$. The associated graph G is depicted on top, and at the bottom we find one of its extending representations, giving the 3-partitioning $\{A_1, A_3, A_6\}$ and $\{A_2, A_4, A_5\}$.

some $\varepsilon > 0$. Three paths can be packed into the same gap if and only if their three integers sum to at most M. Therefore, an extending representation $\mathcal{R}'$ gives a solution to 3-Partition, and vice versa. A similar reduction from BinPacking implies W[1]-hardness when parameterized by the number of pre-drawn intervals; see [213, 214] for details.

This reduction can be easily modified when (i) and (ii) are avoided. We add two extra vertices: w_0 adjacent to v_0 and w_s adjacent to v_s, both non-adjacent to w. It forces the length of w to be in $[s \cdot (M+2) - 1, s \cdot (M+2) + 1)$, so the length b does not have to be prescribed. Also, this reduction works even when non-predrawn intervals do not have lengths assigned. □

5.2 Basic Properties of k-Nested Interval Graphs

In this section, we describe basic definitions and properties about nesting in interval representations and about k-NestedINT.

Definitions. For an interval representation $\mathcal{R}$, the nesting defines a partial ordering $\subsetneq$ of intervals. Intervals $\langle u_1 \rangle, \dots, \langle u_k \rangle$ form a chain of nested intervals of length k if $\langle u_1 \rangle \subsetneq \langle u_2 \rangle \subsetneq \cdots \subsetneq \langle u_k \rangle$. By $\nu(u)$, we denote the length of a longest chain of nested intervals ending with $\langle u \rangle$. We denote $\nu(\mathcal{R})$ the length of a longest chain of nested intervals in $\mathcal{R}$, i.e.,

$$\nu(\mathcal{R}) = \max_{u \in \boldsymbol{V}(G)} \nu(u) \qquad \text{and} \qquad \nu(G) = \min_{\mathcal{R}} \nu(\mathcal{R}) = \min_{\mathcal{R}} \max_{u \in \boldsymbol{V}(G)} \nu(u).$$

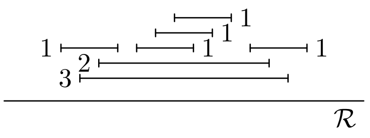
For $A \subseteq \boldsymbol{V}(G)$, we denote by $G[A]$ the subgraph of G induced by A. For a representation $\mathcal{R}$ of G, let $\mathcal{R}[A]$ be the representation of $G[A]$ created by restricting $\mathcal{R}$ to the intervals of A. And for an induced subgraph H of G, let $\mathcal{R}[H] = \mathcal{R}[\boldsymbol{V}(H)]$.

Pruning Twins. Two vertices x and y are twins if and only if $N[x] = N[y]$. The standard observation is that twins can be ignored since they can be represented by identical intervals, and notice that this does not increase nesting and the number of lengths. We can locate all twins in time $\mathcal{O}(n+m)$ [311] and we can *prune* the graph by keeping one vertex per equivalence class of twins. An interval graph belongs to k-NestedINT if and only if the pruned graph belongs to k-NestedINT.

Decomposition into Proper Interval Representations. The following equivalent definition of k-NestedINT is used by Gajarský et al. [136]:

Lemma 5.2.1. *An interval graph belongs to k-NestedINT if and only if it has an interval representation which can be partitioned into k proper interval representations.*

Proof. Let $\mathcal{R}$ be an interval representation partitioned into proper interval representations $\mathcal{R}_1, \dots, \mathcal{R}_k$. No chain of nested intervals contains two intervals from some $\mathcal{R}_i$, so the nesting is at most k. On the other hand, let $\mathcal{R}$ be a k-nested interval representation. We label each interval $\langle u \rangle$ by $\nu(u)$; see Fig. 5.2. Notice that the intervals of each label $i \in \{1, \dots, k\}$ form a proper interval representation $\mathcal{R}_i$. □



Code:
$1_\ell 3_\ell 2_\ell 1_r 1_\ell 1_\ell 1_\ell 1_r 1_r 1_r 1_\ell 2_r 3_r 1_r$

Figure 5.2: We label each interval by the length of a maximal chain of nested intervals ending in it. We code the graph by the left-to-right sequence of left endpoints ℓ and right endpoints r together with their labels.

Efficient encoding. An interval graph can be encoded by $2n\lceil\log n\rceil$ bits by labeling the vertices $1,\dots,n$ and listing the left-to-right ordering of labels of the endpoints. Proper interval graphs can be encoded more efficiently using only $2n$ bits: the sequence of endpoints (ℓ for left one, r for right one), as they appear from left to right. We generalize it for k-NestedINT.

Lemma 5.2.2. *A graph in k-NestedINT can be encoded by $2n\lceil\log k+1\rceil$ bits where n is the number of vertices.*

Proof. See Fig. 1.16b for an example. Let $\mathcal{R}_1,\dots,\mathcal{R}_k$ be the labeling from the proof of Lemma 5.2.1. From left to right, we output ℓ or r for each endpoint together with its labels. This encoding requires $\lceil\log k+1\rceil$ bits per endpoint. □

Minimal Forbidden Induced Subgraphs. Interval graphs and the subclasses k-NestedINT and k-LengthINT are closed under induced subgraphs, so they can be characterized by minimal forbidden induced subgraphs. Lekkerkerker and Boland [249] describe them for interval graphs, and Roberts [304] proved that 1-NestedINT = 1-LengthINT are claw-free interval graphs. On the other hand, 2-LengthINT have infinitely many minimal forbidden induced subgraphs [126] which are interval graphs. In [179], our results in Section 5.3 are used to describe minimal forbidden induced subgraphs for k-NestedINT.

Cleaned Representations. We assume that the reader is familiar with consecutive orderings from Section 3.1. For a given consecutive ordering of maximal cliques, it is easy to construct a representation the number of all nestings called a *cleaned representation.*

Lemma 5.2.3. *For a given consecutive ordering $<$ of maximal cliques, there exists a cleaned representation such that if $\langle u\rangle \subsetneq \langle v\rangle$, then $\langle u\rangle$ is nested in $\langle v\rangle$ in every interval representation with this consecutive ordering $<$. We can construct it in time $\mathcal{O}(n+m)$.*

Proof. We place maximal cliques on the real line according to $<$. For each $v \in \boldsymbol{V}(G)$, we place $\langle v\rangle$ on top of the maximal cliques containing v. Let $v^\leftarrow$ be the left-most clique containing v and $v^\rightarrow$ be the right-most clique containing v. We place $\langle v\rangle$ on the left of $v^\leftarrow$ and on the right of $v^\rightarrow$.

For a maximal clique a, let $u_1,\dots,u_\ell$ be all vertices having $u_i^\leftarrow = a$, i.e., all intervals $\langle u_i\rangle$ start at a. Since there are no twins, we have $u_i^\rightarrow \neq u_j^\rightarrow$ for all $i \neq j$. We order the left endpoints of $\langle u_1\rangle, \dots, \langle u_\ell\rangle$ exactly as the maximal cliques $u_1^{\rightarrow}, \dots, u_\ell^{\rightarrow}$ are ordered in $<$. Similarly, let $v_1, \dots, v_{\ell'}$ be all vertices having $v_i^{\rightarrow} = a$. We order the right endpoins of $\langle v_1\rangle, \dots, \langle v_{\ell'}\rangle$ exactly as the maximal cliques $v_1^{\leftarrow}, \dots, v_{\ell'}^{\leftarrow}$ are ordered in $<$.

The constructed interval representation avoids all unnecessary nesting. We get that $\langle u\rangle \subsetneq \langle v\rangle$ if and only if $v^{\leftarrow} < u^{\leftarrow} \le u^{\rightarrow} < v^{\rightarrow}$ in which case the nesting is clearly forced by the consecutive ordering $<$. The construction clearly runs in time $\mathcal{O}(n+m)$. $\square$

Forced Nestings. We use MPQ-trees, see Sections 3.1 and 4.2 for definitions. We study under which conditions is $\langle u\rangle$ forced to be nested in $\langle v\rangle$ in every interval representation, and we represent this by a partial ordering $\subsetneq_F$. We have $u \subsetneq_F v$ if and only if there there exists a Q-node Q such that $s_v^{\leftarrow}(Q)$ is on the left of $s_u^{\leftarrow}(Q)$ and $s_v^{\rightarrow}(Q)$ is on the right of $s_u^{\rightarrow}(Q)$.

Lemma 5.2.4. *We have $\langle u\rangle \subsetneq \langle v\rangle$ for every interval representation if and only if $u \subsetneq_F v$.*

Proof. If $u \subsetneq_F v$, then every consecutive ordering contains at least one maximal clique containing v on the left of all maximal cliques containing u and at least one on the right, so necessarily $\langle u\rangle \subsetneq \langle v\rangle$.

Suppose that there exists a cleaned representation with $\langle u\rangle \subsetneq \langle v\rangle$. Therefore, every maximal clique contaning u also contains v, so u and v appear in sections of a path from a leaf to the root of the MPQ-tree, and v appears at least as high as u. Suppose that $u \not\subsetneq_F v$. Both u and v do not belong to a same Q-node, otherwise they could not be nested in a cleaned representation. There is no Q-node on the path between u and v, above u; possibly u belongs to all sections of one Q-node. Therefore, we can reorder all these P-nodes to place the subtrees containing u on the side, and the obtained cleaned representation has $\langle u\rangle \not\subsetneq \langle v\rangle$. $\square$

5.3 Recognizing k-nested Interval Graphs

In this section, we describe a linear-time algorithm for computing minimal nesting of interval graphs. By Lemma 5.2.3, the problem reduces to finding a consecutive ordering of maximal cliques which minimizes the nesting of a cleaned representation. So we want to reorder the MPQ-tree to minimize the nesting, which is done by dynamic programming from the bottom to the top.

Intuition. We process the MPQ-tree from the bottom to the top, and we optimize the nesting. Let N be a node of the MPQ-tree and let $T_1, \dots, T_\ell$ be its subtrees. Suppose that we know $\nu(G[T_1]), \dots, \nu(G[T_\ell])$ from the dynamic programming. Is $\nu(G[N])$ determined? The answer is that almost. Let $\mathcal{R}_1, \dots, \mathcal{R}_\ell$ be interval representations of $G[T_1], \dots, G[T_\ell]$ minimizing the nesting. We consider two model situations, depicted in Fig. 5.3:

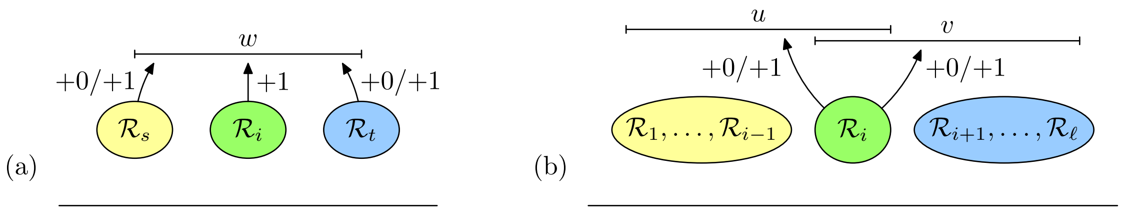


Figure 5.3: (a) The nesting $\nu(G[T_i])$, for $i \neq s, t$, is always increased by one with $\langle w \rangle$, but the nestings $\nu(G[T_s])$ and $\nu(G[T_t])$ may or may not be increased by one. (b) The nesting $\nu(G[T_i])$ may be increased by one with $\langle u \rangle$ or $\langle v \rangle$. It might not be possible to preserve nesting on both sides, for instance when $G[T_i]$ is the disjoint union of $K_{1,3}$ and K_1.

(a) Suppose that N is a P-node with $s(N) = \{w\}$. Then $G[N]$ is the disjoint union of $G[T_1], \dots, G[T_\ell]$ together with the universal vertex w. Every interval representation of $G[N]$ looks as depicted in Fig. 5.3a. We have two representations $\mathcal{R}_s$ and $\mathcal{R}_t$ placed on the left and right sides of $\langle w \rangle$, respectively, while the remaining $\mathcal{R}_i$, for $i \neq s, t$, are placed inside $\langle w \rangle$. Therefore, their nestings $\nu(G[T_i])$ are increased by one with $\langle w \rangle$. On the other hand, some intervals of $\mathcal{R}_s$ and $\mathcal{R}_t$ may stretch out of $\langle w \rangle$, so the nestings $\nu(G[T_s])$ and $\nu(G[T_t])$ is not necessarily increased by one. More precisely, the intervals of $\mathcal{R}_t$ contained in the left-most clique and the intervals of $\mathcal{R}_s$ are not nested in $\langle w \rangle$ in a cleaned representation.

(b) Suppose that N is a Q-node and we consider the following simplified situation depicted in Fig. 5.3b. The graph $G[N]$ consists of $G[T_i]$ together with two universal vertices u and v, each attached some other part of $G[N]$ non-adjacent to all vertices of $G[T_i]$. Then $\mathcal{R}_i$ is covered from, say, left by $\langle u \rangle$ and from right by $\langle v \rangle$. The nesting of $\nu(G[T_i])$ is not necessarily increased by one with $\langle u \rangle$ or $\langle v \rangle$. More precisely, in a cleaned representation, the intervals of $\mathcal{R}_i$ contained in the left-most clique are not nested in $\langle v \rangle$ and those contained in the right-most clique are not nested in $\langle u \rangle$. It is possible that both sides cannot be optimized simultaneously.

Therefore, the dynamic programming computes three values for each subtree T, denoted as a triple (α, β, γ), which we define formally in the next subsection. We have $\alpha = \nu(G[T])$. The value β is the increase in the nesting when T is placed on the side, as in (a); so either $\beta = \alpha$, or $\beta = \alpha - 1$. The value γ is the increase in the nesting of one side, subject to the other side being optimized according to β, as in (b). So always $\beta \leq \gamma$ and either $\gamma = \alpha$ or $\gamma = \alpha - 1$.

5.3.1 Triples (α, β, γ)

For an interval graph G, we define the triple (α, β, γ) as follows. Let G_α, G_β and G_γ be the graphs constructed from G as in Fig. 5.4. Let

$$\alpha = \nu(G_\alpha) - 1, \qquad \beta = \nu(G_\beta) - 1, \qquad \text{and} \qquad \gamma = \nu(G_\gamma) - 1.$$

Similarly, for a subtree T of the MPQ-tree, we define its triple as the triple of $G[T]$. The dynamic algorithm computes triples of all subtrees from the leaves to the root, and outputs a of the root as $\nu(G)$.

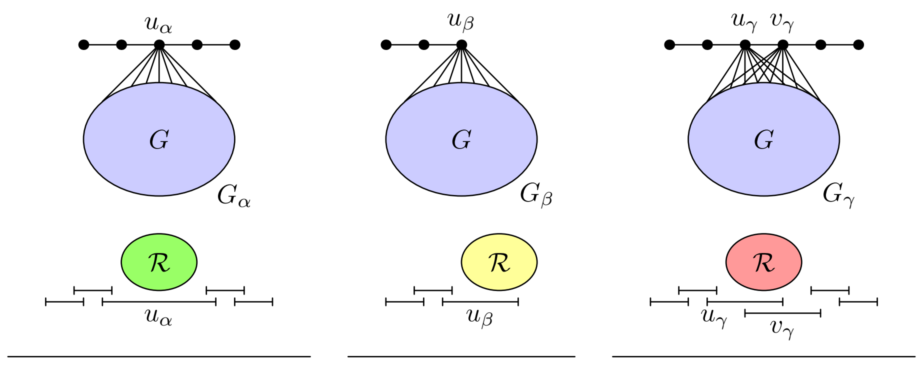


Figure 5.4: The graphs G_α, G_β and G_γ with representations, defining the triple (α, β, γ) of T. The vertices of G are adjacent to the added vertices u_α, u_β, u_γ, and v_γ, and not to the others. In bottom, we depict the structure of their representations with $\mathcal{R}$ being a representation of G.

Lemma 5.3.1. *For every interval graph G, its triple (α, β, γ) satisfies $\alpha - 1 \le \beta \le \gamma \le \alpha$.*

Proof. We prove equivalently that $\nu(G_\alpha) - 1 \le \nu(G_\beta) \le \nu(G_\gamma) \le \nu(G_\alpha)$. We trivially know that $\nu(G_\beta) \le \nu(G_\gamma)$ since G_β is an induced subgraph of G_γ.

The definition of G_α implies that $\nu(G_\alpha) = \nu(G) + 1$, since in every interval representation of G_α, both endpoints of $\langle u_\alpha \rangle$ are covered by attached paths, and therefore a representation $\mathcal{R}$ of G is nested in $\langle u_\alpha \rangle$. Since G is an induced subgraph of G_β, we have $\nu(G) \le \nu(G_\beta)$, so the inequality $\nu(G_\alpha) - 1 \le \nu(G_\beta)$ follows. For an alternative proof, consider a representation of G_β minimizing nesting. We modify it to a representation of G_α by stretching $\langle u_\beta \rangle$ into $\langle u_\alpha \rangle$, which increases nesting by at most one, and by adding the second path attached to $\langle u_\alpha \rangle$. So $\nu(G_\alpha) \le \nu(G_\beta) + 1$.

It remains to show the last inequality that $\nu(G_\gamma) \le \nu(G_\alpha)$. Consider a representation of G_α with minimal nesting, we have G strictly contained inside $\langle u_\alpha \rangle$. By shifting $r(u_\alpha)$ to the left, we get $\langle u_\gamma \rangle$. By adding $\langle v_\gamma \rangle$, we do not increase the nesting and we get a representation of G_γ. So $\nu(G_\gamma) \le \nu(G_\alpha)$. $\square$

Therefore, the triples classify interval graphs into three types; see Fig. 5.5 for examples.

Corollary 5.3.2. *Interval graphs G with $\nu(G) = k$ have triples of three types: $(k, k-1, k-1)$, $(k, k-1, k)$ and (k, k, k).*

Interpreting Triples. Let (α, β, γ) be the triple for G. We want to argue how the formal definition relates to the description in the last paragraph of Intuition. We can interpret the triple of G as increase in the nesting, depending how G is represented with respect to the rest of the graph. Since $\alpha = \nu(G)$, it is easy to understand. Next, we describe an interpretation for the value β.

Lemma 5.3.3. *For every representation of G_β, we have $\nu(u_\beta) \ge \beta + 1$.*

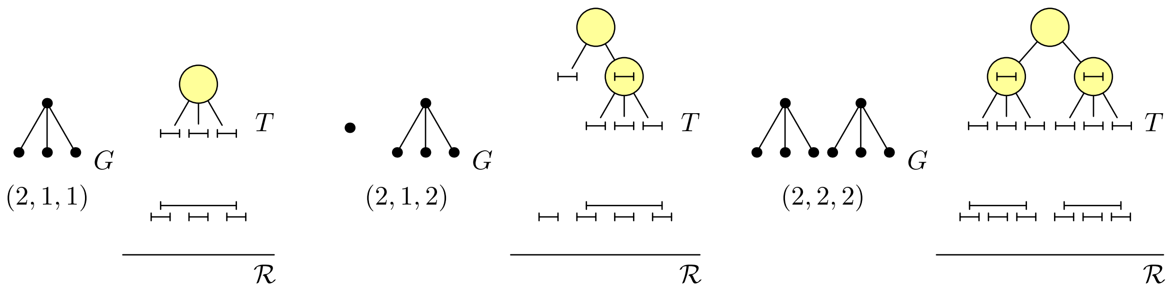


Figure 5.5: Three interval graphs G with $\nu(G) = 2$, together with MPQ-trees T and representations $\mathcal{R}$ minimizing the nesting.

Proof. We assume that a representation $\mathcal{R}_\beta$ of G_β is cleaned; it only decreases nesting. By the definition of β, there exists a maximal chain of nested intervals of length at least $\beta + 1$. Suppose that its length is at least two. Let $\mathcal{R} = \mathcal{R}_\beta[G]$, and we assume that $\langle u_\beta \rangle$ covers $\mathcal{R}$ from the left. If this chain does not end with $\langle u_\beta \rangle$, it ends with an interval of $\mathcal{R}$ placed in the right-most maximal clique. Since every other interval of the chain is nested in $\langle u_\beta \rangle$, we replace this end with $\langle u_\beta \rangle$, and obtain a chain of nested intervals of length at least $\beta + 1$ ending with $\langle u_\beta \rangle$. □

In other words, in every representation of G, there exists a chain of length at least β which is nested in any interval in the rest of the graph which plays the role of $\langle u_\beta \rangle$. In Lemma 5.3.5, we show that there exists a representation for which the length of a longest such chain is exactly β. This links the value β to Fig. 5.3a.

Last, we describe an interpretation for the value γ.

Lemma 5.3.4. *For every representation of G_γ, we have*

$$\min\Big\{\nu(u_\gamma), \nu(v_\gamma)\Big\} \geq \beta + 1 \qquad \textit{and} \qquad \max\Big\{\nu(u_\gamma), \nu(v_\gamma)\Big\} \geq \gamma + 1.$$

Proof. We prove this similar as in Lemma 5.3.3. Consider a cleaned representation $\mathcal{R}_\gamma$ of G_γ. It contains a maximal chain of length at least $\gamma + 1$ ending with $\langle x \rangle$. If $x \neq u_\gamma$ and $x \neq v_\gamma$, we can replace $\langle x \rangle$ with both $\langle u_\gamma \rangle$ and $\langle v_\gamma \rangle$, so both $\nu(u_\gamma) \geq \gamma + 1$ and $\nu(v_\gamma) \geq \gamma + 1$. Otherwise, suppose that, say, $x = v_\gamma$. Then $\nu(v_\gamma) \geq \gamma + 1$ and by removing $\langle v_\gamma \rangle$ and the added intervals, we obtain a representation of $\mathcal{R}_\beta$ with $u_\beta = u_\gamma$. By Lemma 5.3.3, $\nu(u_\beta) \geq \beta + 1$. □

Therefore, in every representation of G, there exists a chain of length at least γ which is nested in any interval in the rest of the graph which plays the role of either $\langle u_\gamma \rangle$ or $\langle v_\gamma \rangle$, while there is a chain of length at least β which in nested in any interval playing the role of the other one. In Lemma 5.3.5, we show that there exists a representation for which the lengths of longest such chains are exactly β and γ, respectively. This links the value γ to Fig. 5.3b.

Minimal Representations. Let (α, β, γ) be a triple of an interval graph G and let $\mathcal{R}$ be a cleaned representation of G with $a^\leftarrow$ and $a^\rightarrow$ being the leftmost and the rightmost maximal cliques in its consecutive ordering of maximal cliques. We define:

$$\nu^\rightarrow(\mathcal{R}) = \max_{x \in \boldsymbol{V}(G) \setminus a^\leftarrow} \nu(x), \qquad \text{and} \qquad \nu^\leftarrow(\mathcal{R}) = \max_{x \in \boldsymbol{V}(G) \setminus a^\rightarrow} \nu(x).$$

The representation $\mathcal{R}$ of G is *minimal* if $\nu(\mathcal{R}) = \alpha$, $\nu^{\rightarrow}(\mathcal{R}) = \beta$ and $\nu^{\leftarrow}(\mathcal{R}) = \gamma$. So a minimal representation $\mathcal{R}$ can be used simultaneously in representations of G_α, G_β and G_γ to get nestings $\alpha + 1$, $\beta + 1$ and $\gamma + 1$, respectively. For instance, all representations in Fig. 5.5 are minimal.

Lemma 5.3.5. *For every interval graph G, there exists a minimal representation $\mathcal{R}$.*

Proof. We argue according to the type of the triple of G.

The triple $(k, k-1, k-1)$. Let $\mathcal{R}_\gamma$ be a cleaned representation of G_γ minimizing the nesting, we have $\nu(\mathcal{R}_\gamma) = k$. Since $\nu(G) = k$, we have $\nu(\mathcal{R}_\gamma[G]) = k$ as well. By Lemma 5.3.4, $\nu(u_\gamma) \geq k$ and $\nu(v_\gamma) \geq k$, so we get equalities. The representation $\mathcal{R} = \mathcal{R}_\gamma[G]$ has $\nu^{\leftarrow}(\mathcal{R}) = \nu^{\rightarrow}(\mathcal{R}) = k - 1$ and $\mathcal{R}$ is minimal.

The triple $(k, k-1, k)$. Let $\mathcal{R}_\beta$ be a cleaned representation of G_β minimizing the nesting such that $\langle u_\beta \rangle$ intersects $\mathcal{R}_\beta[G]$ from left, we have $\nu(\mathcal{R}_\beta) = k$. Let $\mathcal{R} = \mathcal{R}_\beta[G]$. Since $\nu(G) = k$, we have $\nu(\mathcal{R}) = k$ as well. Similarly as in the proof of Lemma 5.3.3, we get $\nu^{\rightarrow}(\mathcal{R}) = k - 1$. If $u_\gamma = u_\beta$ and we add $\langle v_\gamma \rangle$ with the attached path, we obtain a representation of G_γ with nesting at least $k + 1$. Therefore, $\nu^{\leftarrow}(\mathcal{R}) = k$ and $\mathcal{R}$ is minimal.

The triple (k, k, k). Let $\mathcal{R}$ be a cleaned representation of G minimizing the nesting, so $\nu(\mathcal{R}) = k$. If $\nu^{\rightarrow}(\mathcal{R}) < k$ or $\nu^{\leftarrow}(\mathcal{R}) < k$, we can add $\langle u_b \rangle$ and the attached intervals to obtain a representation of G_b of nesting k, so $b = k - 1$; a contradiction. So $\nu^{\rightarrow}(\mathcal{R}) = \nu^{\leftarrow}(\mathcal{R}) = k$ and $\mathcal{R}$ is minimal. □

For a representation $\mathcal{R}$, the *flipped* representation $\mathcal{R}^{\leftrightarrow}$ is created by reversing the left-to-right order of endpoints of intervals. Notice that $\nu(\mathcal{R}) = \nu(\mathcal{R}^{\leftrightarrow})$, $\nu^{\rightarrow}(\mathcal{R}) = \nu^{\leftarrow}(\mathcal{R}^{\leftrightarrow})$ and $\nu^{\leftarrow}(\mathcal{R}) = \nu^{\rightarrow}(\mathcal{R}^{\leftrightarrow})$.

Lemma 5.3.6. *For every interval graph G, there exists a cleaned representation $\mathcal{R}$ of G minimizing the nesting such that for every subtree T of its MPQ-tree, $\mathcal{R}[T]$ is minimal or $\mathcal{R}^{\leftrightarrow}[T]$ is minimal.*

Proof. Let $\mathcal{R}$ be a cleaned representation of G minimizing the nesting, and consider a maximal subtree T for which $\mathcal{R}[T]$ is not minimal. By Lemma 5.3.5, there exists a minimal representation $\mathcal{R}_T$ of $G[T]$. If $\nu^{\rightarrow}(\mathcal{R}[T]) \leq \nu^{\leftarrow}(\mathcal{R}[T])$, let $\mathcal{R}_T^* = \mathcal{R}_T$, otherwise let $\mathcal{R}_T^* = \mathcal{R}_T^{\leftrightarrow}$. Since $\mathcal{R}[T]$ appears consecutively in $\mathcal{R}$, we replace it by $\mathcal{R}_T^*$, and construct a modified cleaned representation $\hat{\mathcal{R}}$ of G. It remains to argue that for every subtree T' containing T, the representation $\hat{\mathcal{R}}[T']$ remains minimal; the lemmas then follows by induction.

We know that $\mathcal{R}[T']$ is minimal. The modification only changed chains which start in $\mathcal{R}_T^*$. By Lemmas 5.3.4, 5.3.5, we get that $\nu(\mathcal{R}_T^*) \leq \nu(\mathcal{R}[T])$, $\nu^{\rightarrow}(\mathcal{R}_T^*) \leq \nu^{\rightarrow}(\mathcal{R}[T])$ and $\nu^{\leftarrow}(\mathcal{R}_T^*) \leq \nu^{\leftarrow}(\mathcal{R}[T])$. Therefore, every chain above $\mathcal{R}[T]$ extends only chains with lengths equal or shorter, so $\hat{\mathcal{R}}[T']$ remains minimal. □

Triples for Leaves. Recall that we have no twins. For a leaf L of the MPQ-tree, we have either $G[L]$ having no vertices (when $s(L) = \emptyset$), or $G[L] \cong K_1$ (when $s(L) = \{w\}$). In the former case, the triple of L is equal $(0, 0, 0)$. In the latter case, it is equal $(1, 0, 0)$.

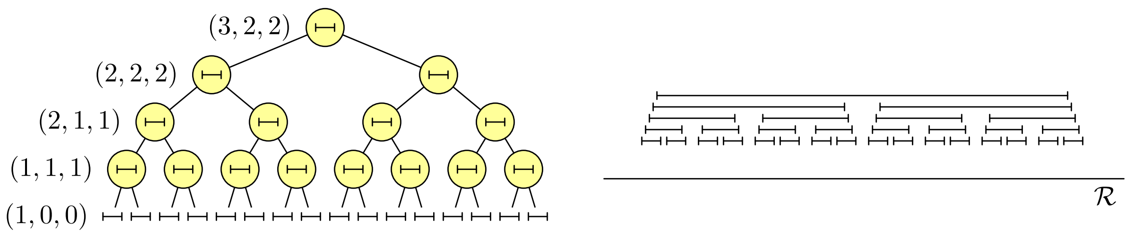


Figure 5.6: An MPQ-tree representing G with the computed triples (α, β, γ) (equal on each level) and a cleaned representation $\mathcal{R}$ minimizing nesting. We have $\nu(G) = 3$.

5.3.2 Triples for P-nodes

Let $T_1, \dots, T_p$ be the children of a P-node P, with $p \geq 2$, with the computed triple $(\alpha_i, \beta_i, \gamma_i)$ for each subtree T_i. We compute the triple (α, β, γ) of the subtree $T = T[P]$ using the following formulas; see Fig. 5.6 for an example:

$$\begin{aligned}
\alpha &= \begin{cases} \max\{\alpha_1, \dots, \alpha_p\}, & \text{if } s(P) = \emptyset, \\ \min_{s \neq t} \max\{\beta_s + 1, \beta_t + 1, \alpha_i + 1 : i \neq s, t\}, & \text{if } s(P) = \{w\}. \end{cases} \\
\beta &= \min_s \max\{\beta_s, \alpha_i : i \neq s\}, \\
\gamma &= \max\{\alpha_1, \dots, \alpha_p\}.
\end{aligned}$$

Lemma 5.3.7. *The formulas compute the triple (α, β, γ) of $T[P]$ correctly.*

Proof. This proof also explains how these formulas are formed.

The value α is computed correctly. We know that $\alpha = \nu(G[T])$. If $s(P) = \emptyset$, then $G[T]$ is the disjoint union of $G[T_1], \dots, G[T_p]$ with $\alpha_i = \nu(G[T_i])$, so $\alpha = \max\{\alpha_1, \dots, \alpha_p\}$. Otherwise, we get the situation from Fig. 5.3a.

First, we argue that $G[T]$ has a representation $\mathcal{R}$ of nesting α from the formula. Intervals of all subtrees except for the leftmost subtree T_s and the rightmost subtree T_t are completely nested inside $\langle w \rangle$; and we minimize over all possible choices of $s \neq t$. Let $\mathcal{R}_i$ be a minimal representation for $G[T_i]$ from Lemma 5.3.5, and we use $\mathcal{R}_s^{\leftrightarrow}$ for $G[T_s]$. For every $i \neq s, t$, we get that $\nu(\mathcal{R}_i) = \alpha_i$ is increased by one with $\langle w \rangle$. For $\mathcal{R}_s$ and $\mathcal{R}_t$, only $\nu^{\leftarrow}(\mathcal{R}_s^{\leftrightarrow}) = \beta_s$ and $\nu^{\rightarrow}(\mathcal{R}_t) = \beta_t$ are increased by one with $\langle w \rangle$. We get

$$\nu(\mathcal{R}) = \nu(w) = \min_{s \neq t} \max\{\beta_s + 1, \beta_t + 1, \alpha_i + 1 : i \neq s, t\} = \alpha.$$

On the other hand, consider a representation $\mathcal{R}$ of $G[T]$. There is no chain of nested intervals containing intervals from two different subtrees T_i and T_j. Let $\mathcal{R}_i = \mathcal{R}[T_i]$ and let $\mathcal{R}_s$ and $\mathcal{R}_t$ be the leftmost and the rightmost of these representations, respectively. For every $i \neq s, j$, the representation $\mathcal{R}_i$ has the nesting at least α_i, so $\nu(\mathcal{R}) \geq \alpha_i + 1$. By Lemma 5.3.4, we know that $\nu^{\leftarrow}(\mathcal{R}_s) \geq \beta_s$ and $\nu^{\rightarrow}(\mathcal{R}_t) \geq \beta_t$ and these chains are nested in $\langle w \rangle$, so $\nu(\mathcal{R}) \geq \max\{\beta_s + 1, \beta_t + 1\}$. So $\nu(\mathcal{R}) \geq \alpha$ from the formula.

The value β is computed correctly. First, we construct a representation $\mathcal{R}_\beta$ of G_β with nesting $\beta + 1$. If $s(P) = \{w\}$, in every cleaned representation, $\langle w \rangle \not\subseteq \langle u_\beta \rangle$, so that every other interval is either nested in both, or in neither. So we can assume that $s(P) = \emptyset$.

When the added intervals are placed on the right of $\langle u_\beta \rangle$, intervals of all subtrees except for a left-most one T_s are completely nested inside $\langle u_\beta \rangle$; and we again minimize over all possible choices of s. Let $\mathcal{R}_i$ be a minimal representation of $G[T_i]$, we use $\mathcal{R}_s^{\leftrightarrow}$ for $G[T_s]$. For every $i \neq s$, we get that the nesting $\nu(\mathcal{R}_i) = \alpha_i$ is increased by one by $\langle u_\beta \rangle$. For T_s, the nesting $\nu^{\leftarrow}(\mathcal{R}_s^{\leftrightarrow}) = \beta_s$ is increased by one with $\langle u_\beta \rangle$. We get

$$\nu(\mathcal{R}_\beta) = \nu(u_\beta) = \min_s \max\{\beta_s + 1, \alpha_i + 1 : i \neq s\} = \beta + 1.$$

For the other implication, consider a cleaned representation of G_β. Similarly, as above, we get that the nesting is at least $\beta + 1$.

The value γ is computed correctly. We just sketch the argument, it is similar as above. Let $\mathcal{R}_i$ be a minimal representation of $G[T_i]$. We may choose T_s and T_t, and use $\mathcal{R}_s^{\leftrightarrow}$ for T_s. Then only the nesting $\nu^{\leftarrow}(\mathcal{R}_s^{\leftrightarrow}) = \beta_s$ is increased by one with $\langle v_\gamma \rangle$ and only the nesting $\nu^{\rightarrow}(\mathcal{R}_t) = \beta_t$ is increased by one with $\langle u_\gamma \rangle$. But since $\mathcal{R}_s^{\leftrightarrow}$ is nested inside $\langle u_\gamma \rangle$ and $\mathcal{R}_t$ is nested inside $\langle v_\gamma \rangle$, it does not matter and the nestings $\nu(\mathcal{R}_s^{\leftrightarrow})$ and $\nu(\mathcal{R}_t)$ are both increased by one anyway. Therefore, this choice of T_s and T_t is useless and a constructed representation of G_γ has the nesting $\max\{\alpha_1, \dots, \alpha_p\} + 1$. The other implication is proved similarly as before. □

Lemma 5.3.8. *For a P-node with p children, the triple (α, β, γ) can be computed in $\mathcal{O}(p)$.*

Proof. By Lemma 5.3.1, we always have either $\beta_i = \alpha_i - 1$, or $\beta_i = \alpha_i$. Only in the former case, we may improve the nesting by choosing $s = i$ or $t = i$. We call subtrees T_i with $\beta_i = \alpha_i - 1$ as *savable.*

For γ and for α with $s(P) = \emptyset$, we just find the maximum α_i which can be done in time $\mathcal{O}(p)$. For α with $s(P) \neq \emptyset$ and β, we first locate all T_i which maximize α_i. If at least one of them is not savable, say T_j, then $\alpha = \alpha_j + 1$ and $\beta = \alpha_j$. Otherwise if all are savable, then the values α and β depend on the number of these subtrees. If there are at most two, then $\alpha = \alpha_i$, otherwise $\alpha = \alpha_i + 1$. If there is exactly one, then $\beta = \beta_i$, otherwise $\beta = \beta_i + 1$. □

5.3.3 Triples for Q-nodes

The situation is more complex and the values γ are also required. Let Q be a Q-node with subtrees $T_1, \dots, T_q$, where $q \geq 3$, each with a triple $(\alpha_i, \beta_i, \gamma_i)$. We want to compute the triple (α, β, γ) of the subtree $T = T[Q]$. Since lengths of chains are not changed by flipping Q, we can fix the left-to-right order of its subtrees as $T_1, \dots, T_q$. See Fig. 5.7 for an example.

Structure of Chains. Suppose that some cleaned representations $\mathcal{R}_1, \dots, \mathcal{R}_q$ of $G[T_1], \dots, G[T_q]$ are chosen. Then the corresponding cleaned representation of $G[T]$ is uniquely determined. What is the structure of chains of nested intervals? Each chain starts in some subtree T_i and then continues with intervals in $s(Q)$ as follows. If it contains $\langle x \rangle \subsetneq \langle y \rangle$ for $x, y \in s(Q)$, then $x \subsetneq_F y$, so y starts more to the left and ends

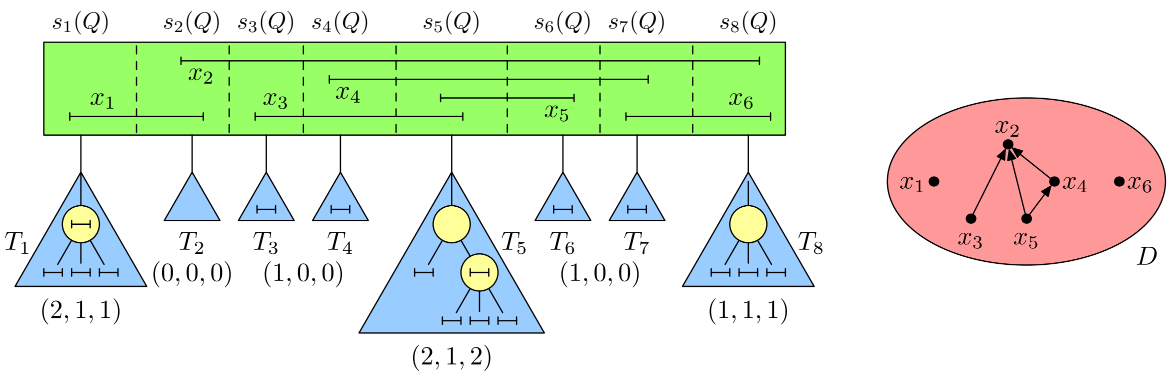


Figure 5.7: On the left, a Q-node Q with eight subtrees. On the right, the DAG D of forced nestings in $s(Q)$.
- For $\mathcal{R}_5^* = \mathcal{R}_5$ (depicted), we get $\nu(x_3) = 2$, $\nu(x_5) = 3$, $\nu(x_4) = 4$, and $\nu(x_2) = 5$.
- For $\mathcal{R}_5^* = \mathcal{R}_5^{\leftrightarrow}$ (by flipping T_5), we get $\nu(x_3) = 3$, $\nu(x_5) = 2$, $\nu(x_4) = 3$, and $\nu(x_2) = 4$.

The second option minimizes the nesting and it gives the triple $(4,3,4)$ for $T[Q]$.

more to the right than x in sections of Q. We represent the relation $\subsetneq_F$ on $s(Q)$ by a DAG D, having an edge (x,y) if and only if $x \subsetneq_F y$; see Fig. 5.7 on the right.

Suppose that a chain of nested intervals of $s(Q)$ of lenght ℓ starts with $\langle x\rangle$. Let $s_x^{\leftarrow}(Q) = s_s(Q)$ and $s_x^{\rightarrow}(Q) = s_t(Q)$, for some $s < t$. Then every chain of every $\mathcal{R}_i$ such that $s < i < t$ is nested in $\langle x\rangle$, so for each $\mathcal{R}_i$, there exists a chain of nested intervals of length $\nu(\mathcal{R}_i) + \ell$. But there might not be chains of lengths $\nu(\mathcal{R}_s) + \ell$ and $\nu(\mathcal{R}_t) + \ell$. The reason is that only chains in $\mathcal{R}_s$ not ending with an interval contained in the leftmost maximal clique of $\mathcal{R}_s$ are nested in $\langle x\rangle$, and only those of $\mathcal{R}_t$ avoiding the rightmost maximal clique of $\mathcal{R}_t$. So there only exist chains of lengths $\nu^{\rightarrow}(\mathcal{R}_s) + \ell$ and $\nu^{\leftarrow}(\mathcal{R}_t) + \ell$.

By Lemma 5.3.5, there exists a minimal representation $\mathcal{R}_i$ of $G[T_i]$ with $\nu(\mathcal{R}_i) = \alpha_i$, $\nu^{\rightarrow}(\mathcal{R}_i) = \beta_i$ and $\nu^{\leftarrow}(\mathcal{R}_i) = \gamma_i$; and we can swap the last two nestings with $\mathcal{R}_i^{\leftrightarrow}$. Let $\mathcal{R}_i^* \in \{\mathcal{R}_i, \mathcal{R}_i^{\leftrightarrow}\}$. We denote $\bigcirc_i^{\rightarrow} = \nu^{\rightarrow}(\mathcal{R}_i^*)$ and $\bigcirc_i^{\leftarrow} = \nu^{\leftarrow}(\mathcal{R}_i^*)$.

$$\begin{array}{rll} \textit{For each } T_i, \textit{ we choose either} \quad \mathcal{R}_i^* = \mathcal{R}_i : & \bigcirc_i^{\rightarrow} = \beta_i, \\ & \bigcirc_i^{\leftarrow} = \gamma_i, \\ \textit{or} \quad \mathcal{R}_i^* = \mathcal{R}_i^{\leftrightarrow} : & \bigcirc_i^{\rightarrow} = \gamma_i, \\ & \bigcirc_i^{\leftarrow} = \beta_i. \end{array} \tag{5.1}$$

By combining these chosen representations $\mathcal{R}_i^*$ for all subtree T_i, we get one of 2^q possible representations $\mathcal{R}[T]$. For each of them, we compute $\nu(x)$ for all $x \in s(Q)$ using the following formulas:

$$\begin{aligned} \nu(x) = \max\{&\bigcirc_s^{\leftarrow} + 1, \bigcirc_t^{\rightarrow} + 1, \alpha_i + 1, \nu(y) + 1 : \\ & s_x^{\leftarrow}(Q) = s_s(Q),\ s_x^{\rightarrow}(Q) = s_t(Q),\ s < i < t \text{ and } y \in \mathrm{Pred}_D(x)\Big\}, \end{aligned} \tag{5.2}$$

where $\mathrm{Pred}_D(x)$ denotes the set of all direct predecessors of x in D. These values can be computed according to a topological sort of D.

Formulas for Triples. The triple (α,β,γ) is determined by minimal nestings of G_α, G_β, and G_γ. We study how chains in $s(Q)$ are extended by the added intervals

$\langle u_\alpha \rangle$, $\langle u_\beta \rangle$, $\langle u_\gamma \rangle$ and $\langle v_\gamma \rangle$. Further, we consider two copies $\langle u_{\beta\leftarrow} \rangle$ and $\langle u_{\beta\rightarrow} \rangle$ of $\langle u_\beta \rangle$. Recall that the left-to-right ordering of the subtrees of Q is fixed. Therefore, $\langle u_\beta \rangle$ can intersect $\mathcal{R}$ either from left (represented by $\langle u_{\beta\rightarrow} \rangle$), or from right (represented by $\langle u_{\beta\leftarrow} \rangle$). Similarly, we assume that $\langle u_\gamma \rangle$ intersects $\mathcal{R}$ from left while $\langle v_\gamma \rangle$ from right.

We add auxiliary vertices u_α, $u_\beta^{\leftarrow}$, $u_\beta^{\rightarrow}$, u_γ and v_γ into D and get the following extended DAG D':

$$\begin{aligned}
\boldsymbol{V}(D') &= \boldsymbol{V}(D) \cup \{u_\alpha, u_{\beta\leftarrow}, u_{\beta\rightarrow}, u_\gamma, v_\gamma\}\\
\boldsymbol{E}(D') &= \boldsymbol{E}(D) \cup \Big\{(x, u_\alpha), (y, u_{\beta\leftarrow}), (y, v_\gamma), (z, u_{\beta\rightarrow}), (z, u_\gamma) :\\
&\qquad\qquad x \in s(Q), y \in s(Q) \setminus s_1(Q), z \in s(Q) \setminus s_q(Q)\Big\}.
\end{aligned}$$

In other words, u_α extends every chain in $s(Q)$, but $u_{\beta\leftarrow}$ and v_γ extend only those not ending with an interval in $s_1(Q)$, and $u_{\beta\rightarrow}$ and u_γ only those not ending with an interval in $s_q(Q)$.

We compute (α, β, γ) of $T[Q]$ using the following formulas:

$$\begin{aligned}
\alpha &= \min_{\forall \mathcal{R}_i^*} \max\Big\{\alpha_1, \dots, \alpha_q, \nu(y) : y \in \mathrm{Pred}_{D'}(u_\alpha)\Big\},\\
\beta^{\leftarrow} &= \min_{\forall \mathcal{R}_i^*} \max\Big\{\beta_1, \alpha_2, \dots, \alpha_q, \nu(y) : y \in \mathrm{Pred}_{D'}(u_{\beta\leftarrow})\Big\},\\
\beta^{\rightarrow} &= \min_{\forall \mathcal{R}_i^*} \max\Big\{\alpha_1, \dots, \alpha_{q-1}, \beta_q, \nu(y) : y \in \mathrm{Pred}_{D'}(u_{\beta\rightarrow})\Big\},\\
\beta &= \min\{\beta^{\leftarrow}, \beta^{\rightarrow}\},\\
\gamma &= \min_{\forall \mathcal{R}_i^*} \max\Big\{\alpha_1, \dots, \alpha_q, \nu(y) : y \in \mathrm{Pred}_{D'}(u_\gamma) \cup \mathrm{Pred}_{D'}(v_\gamma)\Big\}.
\end{aligned}$$

Lemma 5.3.9. *The formulas compute the triple (α, β, γ) of $T[Q]$ correctly.*

Proof. We assume that the left-to-right order of subtrees of Q is fixed, it does not change nesting. Recall that in a cleaned representation $\mathcal{R}$ of $G[T]$, the nesting $\nu(\mathcal{R})$ is determined by representations $\mathcal{R}[T_1], \dots, \mathcal{R}[T_q]$.

The value α is computed correctly. First, we construct a representation $\mathcal{R}_\alpha$ of G_α with $\nu(\mathcal{R}_\alpha) = \nu(u_\alpha) = \alpha + 1$ for α given by the above formula. We construct 2^q representations for all choices of $\mathcal{R}_i^*$ using (5.1), and we use a representation minimizing the nesting, corresponding to the minimum $\min_{\forall \mathcal{R}_i^*}$ in the formula. The choices $\mathcal{R}_i^*$ determine a cleaned representation $\mathcal{R}_\alpha$ of G_α. Nesting of the intervals of $s(Q)$ is computed using (5.2) and $\nu(u_\alpha)$ is equal the length of the longest chain in $\mathcal{R}_\alpha[G[Q]]$ increased by one. The formula for α maximizes over lengths of all chains in $\mathcal{R}_\alpha[G[Q]]$.

On the other hand, consider a cleaned representation $\mathcal{R}$ of $G[T]$. We argue that $\nu(\mathcal{R}) \geq \alpha$ for α given by the above formula. By Lemma 5.3.4, the representation $\mathcal{R}[T_i]$ has $\nu(\mathcal{R}[T_i]) \geq \alpha_i$ and either $\nu^{\rightarrow}(\mathcal{R}[T_i]) \geq \beta_i$ and $\nu^{\leftarrow}(\mathcal{R}[T_i]) \geq \gamma_i$, or $\nu^{\rightarrow}(\mathcal{R}[T_i]) \geq \gamma_i$ and $\nu^{\leftarrow}(\mathcal{R}[T_i]) \geq \beta_i$. As in the proof of Lemma 5.3.6, by replacing $\mathcal{R}[T_i]$ with $\mathcal{R}_i$ in the former case and with $\mathcal{R}_i^{\leftrightarrow}$ in the latter case, we do not increase the nesting. We obtain a representation of $G[T]$ of nesting α, so $\nu(\mathcal{R}) \geq \alpha$.

The value β is computed correctly. Concerning β, in every cleaned representation $\mathcal{R}_\beta$ of G_β, either intervals of $s_q(Q)$ are not nested in $\langle u_\beta \rangle$ (represented by $\langle u_{\beta\leftarrow} \rangle$), or intervals of $s_1(Q)$ are not nested in $\langle u_\beta \rangle$ (represented by $\langle u_{\beta^\rightarrow} \rangle$). We compute both possibilities in $\beta^\leftarrow$ and $\beta^\rightarrow$, and use the minimum. The rest of the arguments is similar as above.

The value γ is computed correctly. Again, the arguments are similar as for α above, the only difference is that $\langle u_\alpha \rangle$ is replaced by both $\langle u_\gamma \rangle$ and $\langle v_\gamma \rangle$. □

Unfortunately, formulas do not directly lead to a polynomial-time algorithm since they minimize over 2^q possible choices of $\mathcal{R}_i^*$. Next, we prove that these choices can be done greedily.

Lemma 5.3.10. *For each of α, $\beta^\leftarrow$, $\beta^\rightarrow$ and γ, we can locally choose $\mathcal{R}_i^*$ minimizing the value.*

Proof. Notice that the choices of $\mathcal{R}_i^*$ are independent of each other since each $\mathcal{R}_i^*$ influences only lenghts of chains starting in $\mathcal{R}[T_i]$. We give a description for α, and it works similarly for the others.

For each $x \in s(Q)$, we compute the length $\ell(x)$ of a longest chain in $s(Q) \cup \{u_\alpha\}$ starting with $\langle x \rangle$. Let

$$\ell_i^\leftarrow = \max_{\substack{x \in s(Q) \\ s_x^\leftarrow(Q) = s_i(Q)}} \ell(x), \qquad \text{and} \qquad \ell_i^\rightarrow = \max_{\substack{x \in s(Q) \\ s_x^\rightarrow(Q) = s_i(Q)}} \ell(x),$$

and let $\ell_i^* = 0$ if no such $x \in s(Q)$ exists. We choose $\mathcal{R}_i^* = \mathcal{R}_i$ if and only if $\ell_i^\rightarrow \geq \ell_i^\leftarrow$. These choices minimize lengths of all chains in a representation of $G[T]$. For instance, in Fig. 5.7, we get $\ell_5^\leftarrow = 3$ and $\ell_5^\rightarrow = 2$, so we choose $\mathcal{R}_5^* = \mathcal{R}_5^\leftrightarrow$. □

Lemma 5.3.11. *For a Q-node Q with q children, the triple (α, β, γ) of $T[Q]$ can be computed in time $\mathcal{O}(q + m_Q)$, where m_Q is the number of edges of $G[s(Q)]$.*

Proof. Since $G[s(Q)]$ is connected, it contains at most m_Q vertices. For every $x \in s(Q)$, we know $s_x^\leftarrow(Q)$ and $s_x^\rightarrow(Q)$ which we use to compute the DAG D. This can be done by considering all m_Q edges, and testing for each whether the pair is nested. Then, we construct the extended DAG D'.

For each $x \in \boldsymbol{V}(D')$ and each $y \in \{u_\alpha, u_\beta^\leftarrow, u_\beta^\rightarrow, u_\gamma, v_\gamma\}$, we compute the length of a longest path from x to y. This can be done in linear time for all vertices x by processing D' from the top to the bottom. For each T_i, we choose greedily $\mathcal{R}_i^*$ as described in the proof of Lemma 5.3.10. We compute the triple (α, β, γ) using the above formulas. The total running time is $\mathcal{O}(q + m_Q)$. □

5.3.4 Construction of Linear-time Algorithm

We use the above results to prove that $\nu(G)$ can be computed in linear time:

Theorem 2.4.2. For an interval graph G, we compute its MPQ-tree in time $\mathcal{O}(n + m)$ [235]. Then we process the tree from the leaves to the root and compute triples (α, β, γ) for every node, as described above. We output α of the root which is the minimal nesting number $\nu(G)$. By Lemmas 5.3.7 and 5.3.9, this value is computed correctly. By Lemmas 5.3.8 and 5.3.11, the running time of the algorithm is $\mathcal{O}(n + m)$. □

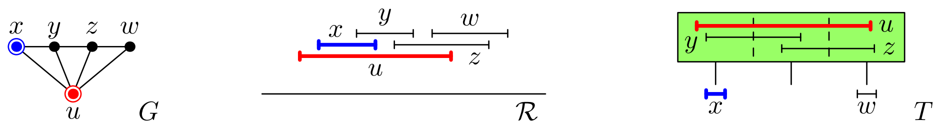


Figure 5.8: On the left, a proper interval graph G, i.e., $\nu(G) = 1$. In the middle, an example of an extending representation $\mathcal{R}$ with $\nu(\mathcal{R}) = 2$, and its nesting is optimal since $\langle x \rangle' \subsetneq \langle u \rangle'$. Further, in every extending representation, $\langle x \rangle' \subsetneq \langle y \rangle$ or $\langle y \rangle \subsetneq \langle u \rangle'$. On the right, the corresponding MPQ-tree T.

5.4 Conclusions

In this thesis, we have introduced k-nested interval graphs which is a new hierarchy of graph classes between proper interval graphs and interval graphs. The presented understanding is already much greater than understanding of k-length interval graphs reached after more than 35 years of their research. We have presented a relatively simple recognition algorithm based on dynamic programming and minimal representations. Other research directions immediately open. In [179], our results are used to derive minimal forbidden induced subgraphs of k-NestedINT.

Problem 5.4.1. *Which structural properties and characterizations of proper interval graphs generalize to k-nested interval graphs?*

Problem 5.4.2. *Which computational problems solvable efficiently for proper interval graphs can be solved efficiently for k-nested interval graphs as well?*

The second problem is interesting for computational problems which are harder for general interval graphs. One example is deciding first-order logic properties which is W[2]-hard for interval graphs [138], but can be solved in FPT for k-nested interval graphs [136].

Partial Representation Extension. While this part deals with the partial representation extension problems, they were not discussed in Chapter 5 at all. The presented results can be used as a prerequisite to attack the problems RepExt(k-NestedINT). In [220], they are used to find an extending interval representation of minimal nesting in polynomial time, by a much more involved dynamic programming than in the recognition algorithm of Section 5.3. Can the structural results of Chapters 4 and 5 and [179] be joined to get a structural characterization of extendible partial representations of k-nested interval graphs?

Problem 5.4.3. *Is it possible to characterize minimal obstructions for partial representation extension of k-nested interval graphs?*

A partial representation $\mathcal{R}'$ poses three restrictions:

(i) Some pre-drawn intervals can be nested in each other which increases the nesting.
(ii) The consecutive ordering has to extend $\vartriangleleft$ which restricts the possible shuffling of subtrees.

(iii) Some subtrees can be optimized differently depending on the side they are attached.

The problem is difficult since we have to deal with them simultaneously. For an example, see Fig. 5.8.

RepExt for *k*-length Interval Graphs. In Theorem 2.4.1, we prove that the problem RepExt(k-LengthINT) is NP-hard for $k \geq 2$. Are these problems NP-complete? The natural idea would be to apply a similar shifting procedure as in the proof of Lemma 2.3.3. But to use such an approach, we first need to restrict k different lengths in every extending representation.

Problem 5.4.4. *Does* RepExt(k-LengthINT) *belong to* NP, *for every* $k \geq 2$*?*

PART II

Extending Algebraic Properties of Graphs

6 Overview of Algebraic Properties of Graphs

This chapter contains:

- *6.1: Outline.* We describe which algebraic properties are studied.
- *6.2: 3-connected Reduction.* We introduce the 3-connected reduction which generalizes connected components and block trees, and decomposes graphs into 3-connected components. All described results are proved in Chapter 7.
- *6.3: Automorphism Groups of Planar Graphs.* We describe results about automorphism groups of restricted classes of graphs and in particular of planar graphs. The inductive Jordan-like characterization of the automorphism groups of planar graphs is proved in Chapter 8.
- *6.4: Graph Isomorphism Problem.* Known complexity results are discussed.
- *6.5: Graph Isomorphism Problem Restricted by Lists.* Lubiw [260] proved that this generalization of the graph isomorphism problem is NP-complete. We describe results for variety of graph classes and graph parameters proved in Chapter 9.
- *6.6: Regular Graph Covers.* We describe motivations for regular graph covers, survey several related problems and describe structural results proved in Chapter 10 and algorithmic results proved in Chapter 11.

`http://pavel.klavik.cz/orgpad/algebraic_properties.html`

6.1 Outline

In Part II, we study algebraic properties related to symmetries of graphs described by automorphism groups, and we study them from both the structural and computational points of view. For a graph G, a bijection $\pi : G \to G$ is called an *automorphism* if $uv \in \boldsymbol{E}(G) \iff \pi(u)\pi(v) \in \boldsymbol{E}(H)$. The automorphism group of G, denoted $\mathrm{Aut}(G)$, consists of all automorphisms of G.

In mathematics and computer science, study of symmetries is important because many objects are highly regular and can be simplified and understood using symmetries. For instance, groups help in designing large computer networks [106, 365]. The well-studied degree-diameter problem asks, given integers d and k, to find a maximal graph X with diameter d and degree k. Such graphs are desirable networks having small degrees and short distances. Currently, the best constructions are highly symmetrical graphs made using groups [281].

Below, we give an outline of this chapter and of Part II:

Section 6.2. We describe the 3-connected reduction which decomposes a graph into its 3-connected components. It is based on our papers [118, 119]. It is the main structural tool of Part II: we study algebraic properties of 3-connected graphs and then combine these findings for general graphs. The results are presented in Chapter 7.

Section 6.3. We describe the automorphism groups of trees, interval graphs, permutation graphs and circle graphs, based on [202, 224, 225]. Then, we give an overview of the first inductive Jordan-like characterizations of the automorphism groups of planar graphs proved in our papers [217, 218], which we present in Chapter 8.

Section 6.4. We describe the known results for the famous graph isomorphism problem, denoted GraphIso. For graph G and H, it asks whether there exists an isomorphism $\pi : G \to H$ satisfying $uv \in \boldsymbol{E}(G) \iff \pi(u)\pi(v) \in \boldsymbol{E}(H)$.

Section 6.5. We present a generalization called list restricted graph isomorphism, first introduced by Lubiw [260]. For each vertex $v \in \boldsymbol{V}(G)$, we are given a list $\mathfrak{L}(v) \subseteq \boldsymbol{V}(H)$ of possible images. In Chapter 9, we prove many results concerning the complexity of list restricted graph isomorphism for variety of graph classes and graph parameters, based on our paper [209].

Section 6.6. Graph covering is a topological notion of local similarity between two graphs G and H. It is given by a covering projection p from the big graph G to the small graph H. In Chapters 10 and 11, we study regular graph covering in which the covering projection is given by a semiregular subgroup Γ of $\mathrm{Aut}(G)$ such that $H \cong G/\Gamma$, and H is called a quotient of G. In Chapter 10, we fully describe the behaviour of regular graph covering with respect to 1-cuts and 2-cuts in G, i.e., with respect to the 3-connected reduction. Based on these structural resuls, in Chapter 11, we construct an FPT algorithm for regular covering testing for planar inputs G. The presented results are based on our papers [118, 119, 120]. Also, in Section 6.6.6, we define the properties (P1), (P2), (P3), and (P3*) for a graph class $\mathcal{C}$, used in many places of Part II.

6.2 3-connected Reduction

It is well known that a disconnected graph can be decomposed into its connected components. Similarly, every connected graph forms a tree of 2-connected *blocks* joined by articulations called *block tree* (see Section 7.3 for details). Therefore, many problems for general graphs may be solved separately for each block of each connected components, and then the results are joined together.

6.2.1 Atoms, Reduction Series and Reduction Tree

In Chapter 7, we describe a generalization called the *3-connected reduction* which decomposes a graph into 3-connected components. Below, we describe only simplified definitions, see Chapter 7 for precise ones.

Atoms. In Section 7.4, we introduce the important definition of an *atom.* Atoms are inclusion-minimal subgraphs with respect to 1-cuts and 2-cuts which cannot be further simplified. Atoms are essentially paths, cycles, stars, or 3-connected graphs. We distinguish three types of atoms:

- *Proper atoms* are inclusion-minimal subgraphs separated by a 2-cut inside a block.
- *Dipoles* are formed by the sets of all parallel edges joining two vertices.
- *Block atoms* are blocks which are leaves of the block tree, or stars of all pendant edges attached to a vertex. The central block is never a block atom.

Our definition of atoms is quite technical, so that it works nicely with respect to regular graph covering studied in Chapters 10 and 11.

Reduction. The 3-connected reduction, described in Section 7.5, forms a *reduction series* of graphs $G = G_0, G_1, \dots, G_r$. The graph G_{i+1} is constructed from G_i by replacing all atoms of G_i by colored edges of two types: directed and undirected. The colors encode the isomorphism classes of atoms while the edge types encode symmetry types of atoms. In this process, we remove details from the graph but preserve its overall structure. The terminal graph in the series, denoted by G_r, contains no atoms and it is called a *primitive graph.* We prove that it is either very simple (essentially K_2 or a cycle), or essentially a 3-connected graph.

Reduction Tree. The 3-connected reduction decomposes a graph G into atoms and a primitive graph. The decomposition process is captured by a *reduction tree.* It is a rooted tree having the primitive graph as a root. The remaining nodes correspond to atoms encountered in the reduction. Each node corresponding to some atom A is attached to the edge e in another node such that e replaces A in the reduction.

6.2.2 Change in Automorphism Groups

It is well-known that the automorphism group $\mathrm{Aut}(G)$ preserves the central block or central articulation of G (see Section 7.3). Similarly, the central block plays a key role in every regular covering projection $p : G \to H$. The reason is that p is non-trivial only on this central block; the remaining blocks are isomorphically preserved in H. Therefore, the atoms and the 3-connected reduction are defined with respect to the central block or articulation, so that it is preserved in the primitive graph. The reduction captures important information about symmetries and regular quotients of G.

More precisely, the reduction from G_i to G_{i+1} is defined in such a way that an induced *reduction epimorphism* $\mathbf{\Phi}_i : \mathrm{Aut}(G_i) \to \mathrm{Aut}(G_{i+1})$ possesses nice properties; see Proposition 7.6.1. We can describe the change of the automorphism group explicitly using $\mathbf{\Phi}_i$:

Proposition 6.2.1. *If G_i is reduced to G_{i+1}, then*

$$\mathrm{Aut}(G_{i+1}) \cong \mathrm{Aut}(G_i)/\mathrm{Ker}(\mathbf{\Phi}_i).$$

The 3-connected reduction behaves nicely with respect to planar graphs. As described in Sections 1.4.1 and 8.1, the automorphism groups of 3-connected planar graphs are spherical groups. Similarly, the regular quotients of 3-connected planar graphs can be easily understood using classical results from geometry (see Section 10.4). In Chapters 8 and 10, we use the 3-connected reduction to describe the automorphism group and the regular quotients of planar graphs, respectively.

6.2.3 Relation to Previous Works

Seminal papers by Mac Lane [265] and Trakhtenbrot [340] introduced this idea of decomposition into 3-connected components. It was further extended in [344, 188, 191, 78, 352, 27]. This decomposition can be represented by a tree whose nodes are 3-connected graphs, and this tree is known in the literature mostly under the name *SPQR tree* [95, 96, 97, 167].

The novelty of our reduction is in the following three key differences with respect to the aforementioned results:

1. The previous results apply the reduction only to 2-connected graphs. Our reduction is applied to all connected graphs, so we also reduce parts separated by 1-cuts. One advantage is that it captures the entire decomposition of a connected graph by one unified tree, making many results more transparent. Other reasons are mentioned in Section 7.3. A disadvantage is that the definition of atoms is more involved.
2. The reduction is augmented by colored edges (encoding different isomorphism classes of the corresponding atoms) which are undirected or directed (encoding different symmetry types of atoms). This allows to capture the changes in the automorphism group as described in Proposition 6.2.1. Some of these ideas also appear in [13]. Further, a third type of halvable edges in introduced in Chapters 10 and 11 to capture semiregular subgroups and regular quotients.
3. Unlike SPQR trees, the reduction tree is rooted, having a primitive graph as the root. Without this, the automorphism group cannot be captured by the reduction tree. The reason is that every automorphism fixes the primitive graph.

We give a more detailed comparison in Section 7.8. Also, the 3-connected reduction is completely described in Chapter 7, without referencing results from the mentioned papers (aside algorithmic results of [191, 167]). Therefore, the reader can get the full understanding just by reading Chapter 7.

6.3 Automorphism Groups of Planar Graphs

Automorphism groups of graphs can be studied from the structural and complexity points of view. Frucht's Theorem [131] states that for every finite abstract group Ψ, there exists a graph G such that $\mathrm{Aut}(G) \cong \Psi$; so automorphism groups of graphs are universal. It is interesting to study which automorphism groups can be realized by restricted graph classes, and we give an overview in Section 6.3.1.

The following definitions are described in [224, 225]. For a graph class $\mathcal{C}$, let $\mathrm{Aut}(\mathcal{C})$ be the class of all abstract groups which can be realized as automorphism groups of $\mathcal{C}$, i.e.,

$$\mathrm{Aut}(\mathcal{C}) = \Big\{\Psi : \exists G \in \mathcal{C}, \mathrm{Aut}(G) \cong \Psi\Big\}.$$

We call $\mathcal{C}$ *universal* if every finite abstract group is in $\mathrm{Aut}(\mathcal{C})$, and *non-universal* otherwise.

The complexity point of view is related to the following computational problem:

Problem:	Computing automorphism group – AutGroup
Input:	A graph G.
Output:	Permutations $\pi_1, \dots, \pi_k$ of $\boldsymbol{V}(G)$ generating $\mathrm{Aut}(G)$.

This problem is closely related to the graph isomorphism problem, as discussed in Section 6.4.

6.3.1 Restricted Graph Classes and Jordan-like Characterizations

Figure 6.1 depicts graph classes with understood automorphism groups studied in this thesis. Below, we sketch descriptions of these automorphism groups using group products, defined in Section 7.2.

Trees. The oldest characterization of automorphism groups of restricted graphs is from 1869 by Jordan [202] who characterized the automorphism groups of trees:

Theorem 6.3.1 (Jordan [202]). *The class* $\mathrm{Aut}(\mathsf{TREE})$ *is defined inductively as follows:*

(a) $\{1\} \in \mathrm{Aut}(\mathsf{TREE})$.
(b) If $\Psi_1, \Psi_2 \in \mathrm{Aut}(\mathsf{TREE})$, *then* $\Psi_1 \times \Psi_2 \in \mathrm{Aut}(\mathsf{TREE})$.
(c) If $\Psi \in \mathrm{Aut}(\mathsf{TREE})$, *then* $\Psi \wr \mathbb{S}_n \in \mathrm{Aut}(\mathsf{TREE})$.

Since every tree T has a center fixed by $\mathrm{Aut}(T)$, we may work with rooted trees T. We construct $\mathrm{Aut}(T)$ from the bottom to the root. For an arbitrary vertex $v \in \boldsymbol{V}(T)$, the automorphism group of T restricted to v and its descendants is build as follows. The root v is fixed while isomorphic classes of subtrees can be arbitrarily permuted. The direct product in (b) constructs the automorphisms that act independently on non-isomorphic subtrees, while the wreath product in (c) constructs the automorphisms that permute n isomorphic subtrees.

Intersection Graphs. Recently, we have characterized automorphism groups of several classes of intersection graphs in the papers [224, 225] with Zeman. These papers study algebraic properties of intersection graphs and form a link between Parts I and II of this thesis. In the rest of this section, we state the results and describe the main ideas.

The paper [225, Section 3] describes a geometric interpretation of automorphisms. Let G be a graph with all representations $\mathfrak{Rep}(G)$. The automorphism group $\mathrm{Aut}(G)$ induces an action on $\mathfrak{Rep}(G)$. Each automorphism $\pi \in \mathrm{Aut}(G)$ transforms an intersection representation $\mathcal{R} \in \mathfrak{Rep}(G)$ into another intersection representation $\pi(\mathcal{R}) \in \mathfrak{Rep}(G)$ by permuting the sets in $\mathcal{R}$ according to π. There are two types of automorphism in $\mathrm{Aut}(G)$.

- When $\mathcal{R} = \pi(\mathcal{R})$, then π belong to the stabilizer of $\mathcal{R}$ and we call it an *automorphism of* $\mathcal{R}$. Often, for a suitable definition of $\mathfrak{Rep}(G)$, stabilizers of all representaions are the same and they form a (normal) subgroup of $\mathrm{Aut}(G)$ denoted $\mathrm{Aut}(\mathcal{R})$. Usually, $\mathrm{Aut}(\mathcal{R})$ is easy to understand.
- When π transforms $\mathcal{R}$ into another representation $\pi(\mathcal{R})$, it is called a *morphism of* $\mathcal{R}$. Understanding the structure of all representations may help with describing possible morphisms. When $\mathrm{Aut}(\mathcal{R})$ is a normal subgroup of $\mathrm{Aut}(G)$, then the quotient $\mathrm{Aut}(G)/\mathrm{Aut}(\mathcal{R})$ is the group of all morphisms.

For well-behaved classes of intersection graphs such as interval graphs, we can use the structure of all representations to characterize the automorphism groups. All representations are described by trees whose nodes correspond to parts of the graph G, e.g., MPQ-trees, split trees, modular trees. It is proved in [224, 225] that these trees capture the automorphism groups. Then we can apply Jordan's bottom-up approach to describe the groups inductively, and we call such inductive characterizations *Jordan-like.* The difference is that the root node N is some graph with some $\mathrm{Aut}(N)$.

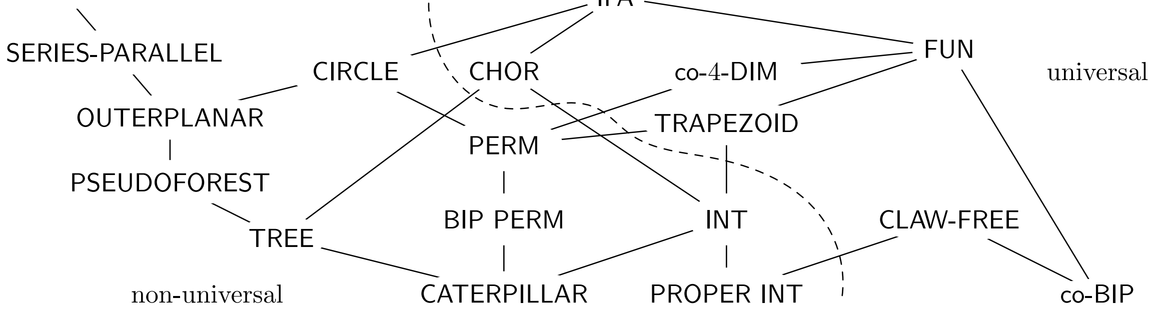


Figure 6.1: Hasse diagram of graph classes with understood automorphism groups.

Therefore, possible permutations of isomorphic subtrees are restricted by $\mathrm{Aut}(N)$, which limits possible group products.

Interval Graphs. Recall MPQ-trees from Sections 3.1 and 4.2. It is proved in [225, Lemma 4.3] that MPQ-trees capture automorphism groups of interval graphs. If an interval graph G has an MPQ-tree T, then $\mathrm{Aut}(T) \cong \mathrm{Aut}(G)/\mathrm{Aut}(\mathcal{R})$. So the automorphism group of T captures all morphisms and it can be constructed inductively from the bottom to the top, similarly as in Theorem 6.3.1. For a P-node P, we may arbitrarily permute vertices of $s(P)$ and arbitrarily permute isomorphic subtrees, so the automorphism group can be constructed using operations (b) and (c) from Theorem 6.3.1. For a symmetric Q-node Q, we may reverse it which corresponds to swapping some intervals in $s(Q)$ and some subtrees, so it can be described by a semidirect product with $\mathbb{C}_2$ and again can be realized by (b) and (c). Therefore, the following is proved:

Theorem 6.3.2 (Klavík and Zeman [224, 225]). *The following equalities hold:*

(i) $\mathrm{Aut}(\textsf{INT}) = \mathrm{Aut}(\textsf{TREE})$,
(ii) $\mathrm{Aut}(\textsf{connected PROPER INT}) = \mathrm{Aut}(\textsf{CATERPILLAR})$,

Concerning (i), this equality is not well known. It was stated by Hanlon [174] without a proof in the conclusion of his paper from 1982 on enumeration of interval graphs. The structural analysis based on MPQ-trees explains this equality and further solves an open problem of Hanlon: for a given interval graph, to construct a tree with the same automorphism group. Without PQ-trees, this equality is surprising since these classes are very different. Caterpillars which form their intersection have very restricted automorphism groups. The result (ii) follows from the known properties of proper interval graphs (see Section 2.3) and (i).

As in Section 3.1, two interval representations are the same if and only if they have the same left-to-right consecutive ordering of maximal cliques. Geometrically,

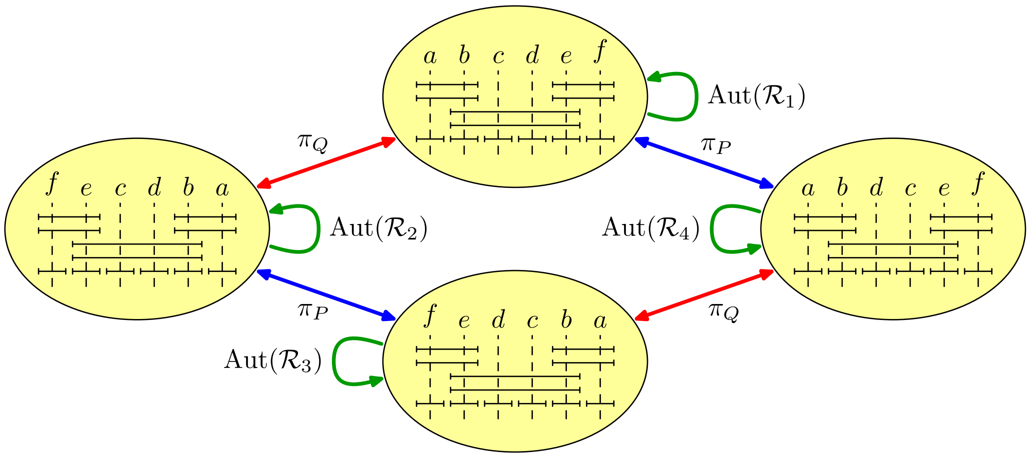


Figure 6.2: An interval graph with four different representations $\mathcal{R}_1, \dots, \mathcal{R}_4$. The MPQ-tree consists of a symmetric Q-node and a P-node, similarly as in Fig. 4.7. The action of two morphisms π_Q and π_P corresponding to the reversal of the Q-node and swapping two children of the P-node is depicted. The stabilizer $\mathrm{Aut}(\mathcal{R}) \cong \mathbb{S}_2^3$.

the automorphism group $\mathrm{Aut}(\mathcal{R})$ of an interval representation $\mathcal{R}$ consists of automorphisms which permute twins (represented by identical intervals), so it is a direct product of symmetric groups. It is a normal subgroup of $\mathrm{Aut}(G)$. Figure 6.2 shows that morphisms $\mathrm{Aut}(G)/\mathrm{Aut}(\mathcal{R})$ of interval representations can be understood from the MPQ-tree. If a symmetric Q-node is reversed, it corresponds to a reflection of a symmetric part of a representation. For P-nodes, we can permute arbitrarily isomorphic parts of a representation.

Permutation Graphs. Recall the modular decomposition and the modular trees from Section 2.7.1. For a graph G with a modular tree T, it is again proved in [224, 225] that $\mathrm{Aut}(G) \cong \mathrm{Aut}(T)$, so the modular decomposition captures all automorphisms. But for general graphs or even comparability graphs G, this is not helpful. The reason is that for a prime node N, the group $\mathrm{Aut}(N)$ may be arbitrary. We cannot hope for any inductive characterization since comparability graphs and even posets are universal [30, 338]. In other words, the stabilizer of a transitive orientation of a comparability graph may be isomorphic to an arbitrary group.

The situation is very different for permutations graphs. Since $\mathsf{PERM} = \mathsf{COMP} \cap \mathsf{co\text{-}COMP}$, it is proved in [225, Lemma 6.6] that for a prime permutation graph G, $\mathrm{Aut}(G)$ is a subgroup of $\mathbb{C}_2^2$. The following Jordan-like characterization is proved:

Theorem 6.3.3 (Klavík and Zeman [225])**.** *The class* $\mathrm{Aut}(\mathsf{PERM})$ *is described inductively as follows:*

(a) $\{1\} \in \mathrm{Aut}(\mathsf{PERM})$,
(b) *If* $\Psi_1, \Psi_2 \in \mathrm{Aut}(\mathsf{PERM})$, *then* $\Psi_1 \times \Psi_2 \in \mathrm{Aut}(\mathsf{PERM})$.
(c) *If* $\Psi \in \mathrm{Aut}(\mathsf{PERM})$, *then* $\Psi \wr \mathbb{S}_n \in \mathrm{Aut}(\mathsf{PERM})$.
(d) *If* $\Psi_1, \Psi_2, \Psi_3 \in \mathrm{Aut}(\mathsf{PERM})$, *then* $(\Psi_1^4 \times \Psi_2^2 \times \Psi_3^2) \rtimes \mathbb{C}_2^2 \in \mathrm{Aut}(\mathsf{PERM})$.

In comparison to Theorem 6.3.1, there is the additional operation (d) which shows that $\mathrm{Aut}(\mathsf{TREE}) \subsetneq \mathrm{Aut}(\mathsf{PERM})$. Geometrically, the group $\mathbb{C}_2^2$ in (d) is generated by the horizontal and vertical reflections of a symmetric permutation representation;

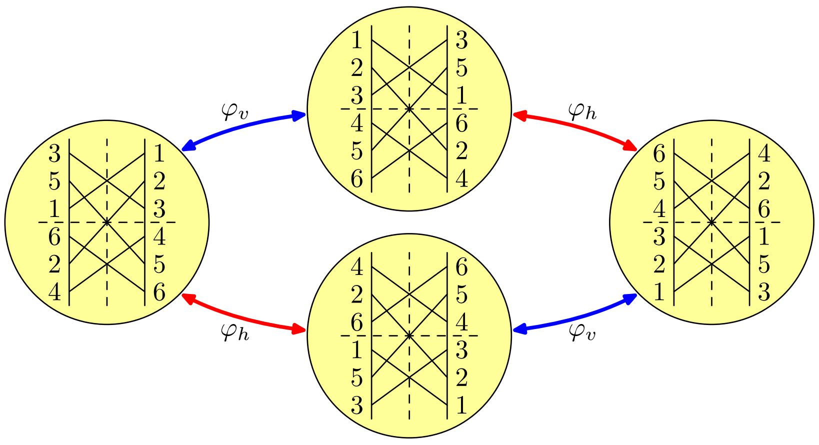


Figure 6.3: A symmetric prime permutation graph G with $\mathrm{Aut}(G) \cong \mathbb{C}_2^2$, generated by two reflections.

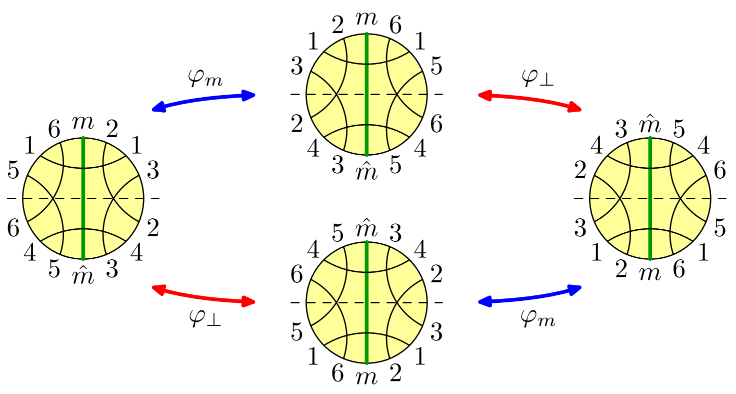


Figure 6.4: For a symmetric prime circle graph, the stabilizer of m is generated by two depicted reflections.

see Fig. 6.3. Also, the automorphism groups of *bipartite permutation graphs* (denoted BIP PERM) are characterized in [225].

Circle Graphs. Recall the split decomposition and the split trees from Section 2.6. The characterization of the automorphism groups of circle graphs is most involved. Again, automorphism groups are captured by the minimal split decomposition. For the split tree T corresponding to a connected graph G, it is proved in [224, 225] that $\mathrm{Aut}(G) \cong \mathrm{Aut}(T)$. Every automorphism of T preserves the center T so we may assume that T is rooted and has a root node.

We apply the bottom-up Jordan's approach on T. Let N be a node with attached subtrees. If N is not a root node, we may only permute the attached subtrees according to the stabilizer of the marker vertex m of N which attaches N towards the root. But when N is the root node, nothing has to be stabilized, so we are more free to permute subtrees. Degenerate graphs are easy to deal with. Using geometry, it is proved in [225, Lemmas 5.5 and 5.6] that for a prime circle graph G, $\mathrm{Aut}(G)$ is a subgroup of $\mathbb{D}_n$ and the stabilizer of a vertex is a subgroup of $\mathbb{C}_2^2$ as depicted in Fig. 6.4.

We get the following characterization of the automorphism groups of connected circle graphs where Σ is the class of all stabilizers of connected circle graphs:

Theorem 6.3.4 (Klavík and Zeman [224, 225])**.** *Let Σ be the class of groups defined inductively as follows:*

(a) $\{1\} \in \Sigma$.
(b) *If* $\Psi_1, \Psi_2 \in \Sigma$, *then* $\Psi_1 \times \Psi_2 \in \Sigma$.
(c) *If* $\Psi \in \Sigma$, *then* $\Psi \wr \mathbb{S}_n \in \Sigma$.
(d) *If* $\Psi_1, \Psi_2, \Psi_3, \Psi_4 \in \Sigma$, *then* $(\Psi_1^4 \times \Psi_2^2 \times \Psi_3^2 \times \Psi_4^2) \rtimes \mathbb{C}_2^2 \in \Sigma$.

Then Aut(connected CIRCLE) *consists of the following groups:*

- *If* $\Psi \in \Sigma$, *then* $\Psi \wr \mathbb{C}_n \in$ Aut(connected CIRCLE).
- *If* $\Psi_1, \Psi_2 \in \Sigma$, *then*

$$(\Psi_1^{2n} \times \Psi_2^n) \rtimes \mathbb{D}_n \in \mathrm{Aut}(\textsf{connected CIRCLE}), \qquad \forall n \text{ odd.}$$

- *If* $\Psi_1, \Psi_2, \Psi_3 \in \Sigma$, *then*

$$(\Psi_1^{2n} \times \Psi_2^n \times \Psi_3^n) \rtimes \mathbb{D}_n \in \mathrm{Aut}(\textsf{connected CIRCLE}), \qquad \forall n \text{ even.}$$

The automorphism group of general (disconnected) circle graphs are easily described by Theorem 7.2.1. The inductive Jordan-like characterization of the automorphism groups of planar graphs, derived in Chapter 8, works in similar two steps. First, we characterize stabilizers corresponding to automorphism groups of subtrees. Then, we combine them with the automorphism group of a primitive graphs in the root node.

Universal Graph Classes. Figure 6.1 depicts several graph classes for which the automorphism groups are known to be universal. Proofs mostly directly follow from GI-hardness reductions. To show that GraphIso is equally hard for a graph class $\mathcal{C}$ as for all graphs, the reduction encodes all graphs into graphs from $\mathcal{C}$. These reductions usually preserve the automorphism groups. Therefore, Frucht's Theorem [131] implies universality of $\mathrm{Aut}(\mathcal{C})$. In Section 9.2, we define these reductions more formally and give an overview of reductions for many graph classes.

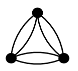
Open Problems. In [225, 56, 345], it is proved that 4-DIM is universal and its GraphIso is GI-complete. The reduction in [225] is fairly simple, to construct G', it replaces every edge of G by a path of length 8. As usual, the non-trivial part is to prove that $G' \in$ 4-DIM for every graph G. On the other hand, since the automorphism groups of permutation graphs are non-universal by Theorem 6.3.3 and the graph isomorphism problem can be solved in linear time by combining [67, 326, 272], we have the following natural open problem:

Problem 6.3.5. *What is the complexity of* GraphIso *for* 3-DIM*? Is* Aut(3-DIM) *universal?*

The second open problem involved the automorphism groups of circular-arc graphs; see also 2.10. We note that the complexity of GraphIso is open for circular-arc graphs.

Problem 6.3.6. *What is* Aut(CIRCULAR-ARC)*?*

Recall H-GRAPH from Section 2.5. Chaplick et al. [61] prove that H-GRAPH are universal if H contains as a minor the multigraph

$M =$  .

We note that these H-GRAPH generalize CIRCULAR-ARC.

6.3.2 Babai's Characterization

In 1960's, there were several generalizations of Frucht's Theorem [132, 313] proving universality of the automorphism groups for graphs of many restricted properties: for a fixed chromatic number, for k connected graphs, for k-regular graphs, etc. The notable exceptions were trees, characterized in Theorem 6.3.1, and 3-connected planar graphs, characterized by Mani [268] (see Section 8.1). In 1969, Turán asked for a characterization of the automorphism groups of planar graphs.

Babai was a student of Turán and worked on this problem. In 1972, Babai [10] proved that no planar graph has the automorphism group isomorphic to the group of quaternions, proving non-universality of Aut(PLANAR). In 1973, Babai [11, Corollary 8.12] described a characterization of the automorphism groups of planar graphs; see Section 8.4 for the exact statement.

As we describe in Section 8.1, the automorphism groups of 3-connected planar graphs are spherical groups. Babai's characterization [11] describes the automorphism groups of k-connected planar graphs, where $k < 3$, are constructed by wreath products of the automorphism groups of $(k+1)$-connected planar graphs and of stabilizers of k-connected graphs. Babai points out in [11, p. 69] that his characterization is not inductive:

> "For the case of planar graphs, we determine the groups occuring in the Main Theorem, as abstract groups (up to isomorphism). [. . .] It cannot be, however, considered as a characterization by recursion of the automorphism groups of the planar graphs, since the group construction refers to the action of the constituents of the wreath products."

Babai's characterization has several further disadvantages. The statement is very long, separated into multiple cases and subcases. More importantly, is not clear precisely which abstract groups belong to Aut(PLANAR). The used language is complicated, very difficult for non-experts in permutation group theory and in graph symmetries.

In [13, p. 1457–1459], Babai gives a more understandable overview of two key ideas (automorphism groups of 3-connected planar graphs, and 3-connected reduction), with his characterization only sketched as "a description of the automorphism groups of planar graphs in terms of generalized wreath products of symmetric groups and polyhedral groups." Babai also states an easy consequence:

Theorem 6.3.7 (Babai [13])**.** *If G is planar, then the group* $\mathrm{Aut}(G)$ *has a subnormal chain*

$$\mathrm{Aut}(G) = \Psi_0 \triangleright \Psi_1 \triangleright \cdots \triangleright \Psi_m = \{1\}$$

such that each quotient group Ψ_{i-1}/Ψ_i is either cyclic or symmetric or $\mathbb{A}_5$.

This theorem describes $\mathrm{Aut}(G)$ only very roughly, by stating its building blocks. For comparison, an easy consequence of our characterization (Theorem 6.3.8) describes these building blocks more precisely and also states how they are "put together" in $\mathrm{Aut}(G)$. In particular, the composition of the action of a spherical group Σ with a stabilizer of a vertex in a planar graph depends on the action of Σ on vertices and edges of the associated 3-connected planar graph. To describe it in detail is not an easy task, and we do it in a standalone paper [218]; see Tables 8.1, 8.2, and 8.3. Our characterization also describes "geometry" of the automorphism groups in terms of actions on planar graphs around every 1-cut and 2-cut.

The class of planar graphs is of great importance, and thus we are convinced that a more detailed and transparent description of their symmetries is of an interest. These reasons led us to write the papers [217, 218] which describes Aut(PLANAR) more understandably, in more detail.

6.3.3 The Jordan-like Characterization

Let G be a planar graph. If it is disconnected, then $\mathrm{Aut}(G)$ can be constructed from the automorphism groups of its connected components (Theorem 7.2.1). Therefore, in the rest of the paper, we assume that G is connected.

In Chapter 8, we describe an involved inductive Jordan-like characterization of Aut(connected PLANAR). We apply the reduction series on G and obtain the reduction tree T which captures $\mathrm{Aut}(G)$. If G is planar, then all nodes of T, corresponding to atoms and the primitive graph, are either very simple or 3-connected. From geometry, their automorphism groups are either spherical groups, or direct products of symmetric groups (see Section 8.1). Our characterization combines these automorphism groups and describes the automorphism groups of planar graphs without referring to planarity, as a simple recursive process which builds them from a few standard groups. A short version of our main result reads as follows:

Theorem 6.3.8. *Let G be a connected planar graph with the reduction series $G = G_0, \dots, G_r$. Then $\mathrm{Aut}(G_r)$ is a spherical group and $\mathrm{Aut}(G_i) \cong \Psi_i \rtimes \mathrm{Aut}(G_{i+1})$, where Ψ_i is a direct product of symmetric, cyclic and dihedral groups.*

We characterize Aut(connected PLANAR) in two steps. First, similarly as in Theorem 6.3.1, we describe in Theorem 8.2.1 an inductive characterization of stabilizers of vertices of planar graphs, denoted Fix(PLANAR). It is the class of groups closed under the direct product, the wreath product with symmetric and cyclic groups and semidirect products with dihedral groups. In Theorem 8.2.10, we give Aut(connected PLANAR) precisely as the class of groups

$$(\Psi_1^{m_1} \times \cdots \times \Psi_\ell^{m_\ell}) \rtimes \mathrm{Aut}(H),$$

where $\Psi_i \in$ Fix(PLANAR) and H is a 3-connected planar graph with colored vertices and colored, possibly oriented, edges. The group $\mathrm{Aut}(H)$ acts on the factors of the direct product $\Psi_1^{m_1} \times \cdots \times \Psi_\ell^{m_\ell}$ in the natural way, permuting the isomorphic factors, following the action of H on the vertices and edges of H; for more details, see Section 8.2.

In Section 8.3, we apply this Jordan-like characterization to describe the automorphism groups of 2-connected planar graphs, outerplanar graphs and series-parallel graphs. and of the following subclasses of planar graphs.

6.3.4 Quadratic-time Algorithm

Explicit algorithms for computing automorphism groups of restricted graph classes are quite rare. Colbourn and Booth [68] describe linear-time algorithms computing permutation generators of trees, interval graphs and outerplanar graphs. The papers [224, 225] describe a linear-time algorithm for permutation graphs and a polynomial-time algorithm for circle graphs. Luks's algorithm [263] and other algorithms for graph isomorphism based on group theory compute automorphism groups; see Section 6.4.

The graph isomorphism problem of planar graphs was attacked in papers [188, 192]. Finally, linear-time algorithms were described by Hopcroft and Wong [194], and by Fontet [130]. As we explain in Section 8.5, the fundamental difficulty is deciding isomorphism of 3-connected (colored) planar graphs in linear time. The idea in [194] is to modify both graphs by a series of reductions ending with colored platonic solids, cycles, or K_2. This is a seminal paper used by many other computer science algorithms as a black box; e.g., [252, 279, 207, 206]. Unfortunately, full versions of [130, 194] were never published.

Colbourn and Booth [68] propose the idea to modify the algorithm of [194] for computing the automorphism groups of planar graphs in linear time. The following is stated in [68, p. 223]:

> "Necessarily we will only be able to sketch our procedure. A more complete description and a proof of correctness would require a more thorough analysis of the Hopcroft-Wong algorithm than has yet appeared in the literature."

To the best of our knowledge, no such algorithm was ever described in detail. We note that it is not possible to use the result of [194] as a black box for computing generators of the automorphism group, since one has to check carefully that the applied reductions preserve the automorphism group. (Or that the change of the automorphism group is under control, similarly as in Proposition 7.6.4.)

By combining the results of [130, 194] and [270], the best previously known polynomial-time algorithm computing generators the automorphism group of a planar graph runs in time $\mathcal{O}(n^4)$. In Section 8.5, we describe a quadratic-time algorithm based on our structural description of the automorphism groups of planar graphs.

Theorem 6.3.9. *There exists a quadratic-time algorithm which computes generators of* $\mathrm{Aut}(G)$ *of an input planar graph* G *in terms of group products of symmetric and spherical groups and of permutation generators.*

Visualization of Symmetries. As one of the applications of graph symmetries, drawing of planar graphs maximizing the symmetries of the picture were studied in [103, 186, 184, 185]. Disadvantage of this approach is that even though the automorphism group $\mathrm{Aut}(G)$ of a planar graph G might be huge, it is possible to highlight only a small fraction of its symmetries; usually just a dihedral or cyclic subgroup. Even if we would consider drawing on the sphere, one can only visualize a spherical subgroup of $\mathrm{Aut}(G)$. Based on our structural decomposition of $\mathrm{Aut}(G)$, we propose in Section 8.6 a different spatial visualization which allows to capture the entire automorphism group, and thus visualize our characterization.

6.4 Graph Isomorphism Problem

For graphs G and H, a bijection $\pi : G \to H$ is called an *isomorphism* if $uv \in \boldsymbol{E}(G) \iff \pi(u)\pi(v) \in \boldsymbol{E}(H)$; so an automorphism of G is an isomorphism $G \to G$. Graphs G and H are *isomorphic*, denoted $G \cong H$, if there exists an isomorphism from G to H.

Problem:	Graph isomorphism – GraphIso
Input:	Graphs G and H.
Question:	Is there an isomorphism $\pi : G \to H$?

The problem GraphIso obviously belongs to NP, and no polynomial-time algorithm is known. It is a prime candidate for an intermediate problem with complexity between P and NP-complete. There are threefold evidences that GraphIso is unlikely to be NP-complete: equivalence of existence and counting [12, 270], GraphIso belongs to coAM, so the polynomial-hierarchy collapses if GraphIso is NP-complete [152, 318], and GraphIso can be solved in quasipolynomial time [14]. For a survey, see [13]. Let GI be the class of all decision problem which have a polynomial-time reduction to GraphIso.

The graph isomorphism problem is solved efficiently for various restricted graph classes and parameters, see Fig. 6.5.

Combinatorial Algorithms. A prime example is the linear-time algorithm for testing graph isomorphism of (rooted) trees. It is a bottom-up procedure comparing subtrees. This algorithm is very robust and captures all possible isomorphisms. For many other graph classes, graph isomorphism reduces to graph isomorphism of labeled trees: for planar graphs [191, 189, 194], interval graphs [262], circle graphs [195], and permutation graphs [67, 326]. Involved combinatorial arguments are used to solve graph

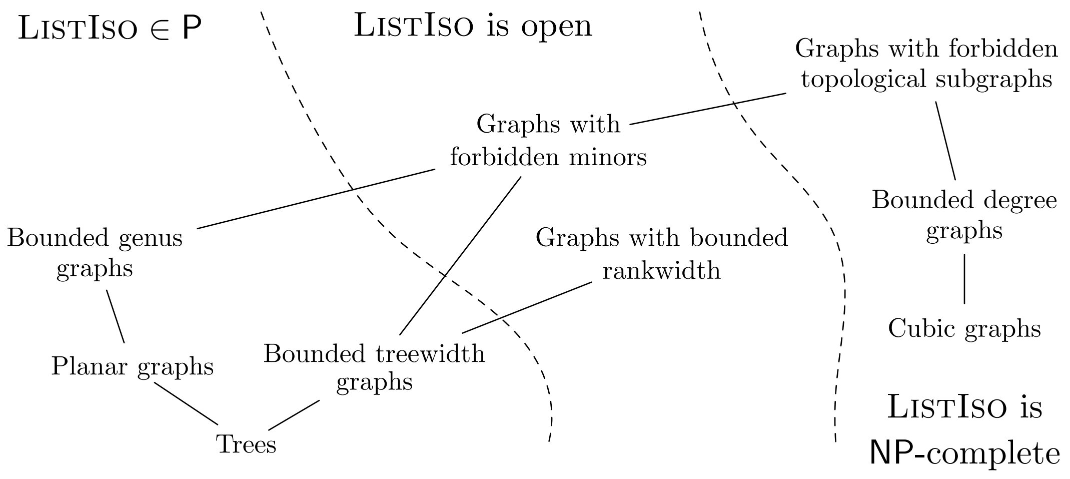


Figure 6.5: Important graph classes for which the graph isomorphism problem can be solved in polynomial time. Our complexity results for the list restricted graph isomorphism problem proved in Chapter 9 are depicted.

isomorphism for bounded genus graphs [252, 122, 279, 206] and bounded treewidth graphs [37, 256].

Algorithms Based on Group Theory. The graph isomorphism problem is closely related to group theory, in particular to computing generators of automorphism groups of graphs. Assuming that G and H are connected, we can test $G \cong H$ by computing generators of $\mathrm{Aut}(G \dot\cup H)$ and checking whether there exists a generator which swaps G and H. For the converse relation, Mathon [270] proved that generators of the automorphism group can be computed using $\mathcal{O}(n^3)$ instances of graph isomorphism.

Therefore, GraphIso can be attacked by techniques of group theory. A prime example is the seminal result of Luks [263] which uses group theory to solve GraphIso for graphs of bounded degree in polynomial time. If G has bounded degree, its automorphism group $\mathrm{Aut}(G)$ may be arbitrary, but the stabilizer $\mathrm{Aut}_e(G)$ of an edge e is restricted. Luks' algorithm tests GraphIso by an iterative process which determines $\mathrm{Aut}_e(G)$ in steps, by adding layers around e.

Group theory can be used to solve GraphIso of colored graphs with bounded sizes of color classes [134] and of graphs with bounded eigenvalue multiplicity [15, 108]. Miller [280] solved GraphIso of k-contractible graphs (which generalize both bounded degree and bounded genus graphs), and his results are used by Ponomarenko [298] to show that GraphIso can be decided in polynomial time for graphs with excluded minors. Luks' algorithm [263] for bounded degree graphs is also used by Grohe and Marx [163] as a subroutine to solve GraphIso on graphs with excluded topological subgraphs. The recent breakthrough of Babai [14] heavily uses group theory to solve the graph isomorphism problem in quasipolynomial time.

Is Group Theory Needed? One of the fundamental problems for understanding the graph isomorphism problem is to understand in which cases group theory is really needed, and in which cases it can be avoided.[1] For instance, for which graph classes can GraphIso be decided by the classical algorithm called k-dimensional Weisfieler-Leman refinement (k-WL)? (Described in Section 9.9.)

Ponomarenko [298] used group theory to solve GraphIso in polynomial time on graphs with excluded minors. Robertson and Seymour [307] proved that a graph G with an excluded minor can be decomposed into pieces which are "almost embeddable" to a surface of genus g, where g depends on this minor. Recently, Grohe [162] generalized this to show that for G, there exists a treelike decomposition into almost embeddable pieces which is *automorphism-invariant* (every automorphism of G induces an automorphism of the treelike decomposition). Using this decomposition, it is possible to solve graph isomorphism in polynomial time and to avoid group theory techniques. In particular, k-WL can decide graph isomorphism on graphs with excluded minors where k depends on the minor.

It is a long-standing open problem whether the graph isomorphism problem for bounded degree graphs, and in particular for cubic graphs, can be solved in polynomial time without group theory. It is known that k-WL, for any fixed k, cannot decide graph isomorphism on cubic graphs [46]. Very recently, fixed parameter tractable algorithms

[1]Ilya Ponomarenko in personal communication.

for graphs of bounded treewidth [256] and for graphs of bounded genus [206] were constructed. On the other hand, the best known parameterized algorithm for graphs of bounded degree is the XP algorithm of Luks [263], and it is a major open problem whether an FPT algorithm exists.

6.5 Graph Isomorphism Problem Restricted by Lists

In this thesis, we propose a different approach to show limitations of techniques used to attack the graph isomorphism problem described in Section 6.4. We study its generalization called *list restricted graph isomorphism* (ListIso) defined below which is NP-complete for general graphs.

Implications for GraphIso. The study for ListIso allows to classify the results for the graph isomorphism problem. An algorithm for GraphIso is called *robust* if it can be modified to solve ListIso while preserving the complexity. (Say, it remains a polynomial-time algorithm, fixed parameter tractable algorithm, etc.)

We understand that the notions of modified algorithms and of robustness are vague. For instance, if an algorithm B is created from A by completely replacing A with B, is B still a modification of A? At this moment, a precise definition of robustness is unclear, but the reader may understand it intuitively, similarly as a statement: "The proof of the result X is created by a modification of a proof of a result Y." Robustness is not used in any formal statement mentioned in this thesis. The purpose of this section and of Chapter 9 is to give more insight into this notion in the context of the graph isomorphism problem.

We show that many combinatorial algorithms for graph isomorphism are robust. On the other hand, hardness results for ListIso imply non-existence of robust algorithms for GraphIso. In particular, we show that ListIso is NP-complete for cubic graphs, so no robust algorithm for cubic graph isomorphism exists, unless P = NP. Similarly, no robust FPT algorithm for graph isomorphism of graphs of bounded degree exists.

Known Results. In 1981, Lubiw [260] introduced the computational problems ListIso and ListAut. Let G and H be graphs, and the vertices of G be equipped with lists: each vertex $u \in \boldsymbol{V}(G)$ has a list $\mathfrak{L}(u) \subseteq \boldsymbol{V}(H)$. We say that an isomorphism $\pi : G \to H$ is *list-compatible* if, for all vertices $u \in \boldsymbol{V}(G)$, we have $\pi(u) \in \mathfrak{L}(u)$; see Fig. 6.6a. A list-compatible isomorphism $\pi : G \to G$ is called a *list-compatible automorphism.* The existence of a list-compatible isomorphism is denoted by $G \xrightarrow{\mathfrak{L}} H$.

Problem:	List restricted graph isomorphism – ListIso
Input:	Graphs G and H, and the vertices of G are equipped by lists $\mathfrak{L}(u) \subseteq \boldsymbol{V}(H)$.
Question:	Is there a list-compatible isomorphism $\pi : G \to H$?

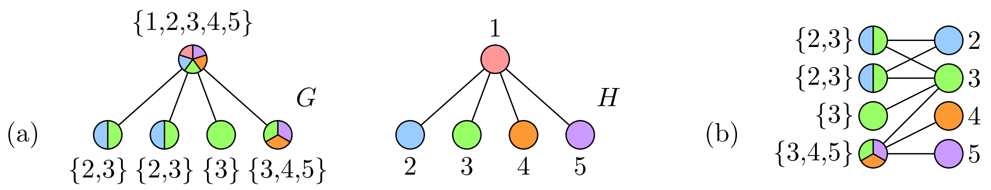


Figure 6.6: (a) Two isomorphic graphs G and H with no list-compatible isomorphism. (b) It does not exist because there is no perfect matching between the lists of the leaves of G and the leaves of H.

Problem:	List restricted graph automorphism – ListAut
Input:	A graph G with vertices equipped with lists $\mathfrak{L}(u) \subseteq \boldsymbol{V}(G)$.
Question:	Is there a list-compatible automorphism $\pi : G \to G$?

These two problems are polynomially equivalent (see Lemma 9.1.3). Lubiw [260] proved the following surprising result:

Theorem 6.5.1 (Lubiw [260])**.** *The problems* ListIso *and* ListAut *are* NP*-complete.*

Moreover, she even proved even following stronger result:

Theorem 6.5.2 (Lubiw [260])**.** *Deciding whether a graph has a fixed-point free involutory automorphism is* NP*-complete.*

Unfortunately, even though the paper [260] has over 50 citations in Web of Science and about 150 citations in Google Scholar, the problem ListIso was not further studied. We believe that one of the reasons is that the definition of ListIso and Theorem 6.5.1 was overshadowed by a simple statement of Theorem 6.5.2. For instance, Babai [13] only describes Theorem 6.5.2. This line of research was further followed by Lalonde [246] who showed that it is NP-complete to decide whether a bipartite graph has an involutory automorphism exchanging the parts. Also, see [129] for an alternative description using star systems.

We read the paper [260] already in 2012. We have independently rediscovered ListIso while studying the computational complexity of regular graph covers, described in Chapters 10 and 11. As one subroutine of the algorithm of Theorem 6.6.3, we solve ListIso on 3-connected planar and projectively planar graphs. We noticed that [260] contains the definition of ListIso only in 2016 when we started studying the computational complexity of ListIso in general.

We note that other computational problems restricted by lists are frequently studied. *List coloring*, introduced by Vizing [348], is NP-complete even for planar graphs [241] and interval graphs [31]. *List H-homomorphisms*, having a similar setting as ListIso, were also considered; see [181, 83, 64].

It was suggested by an anonymous reviewer that unlike for GraphIso, for which the role of graphs G and H is symmetric, the role of G and H in ListIso is asymmetric, and thus the ListIso problem is closer to the H-homomorphism problem. We argue that this is not the case, ListIso is symmetric as well. Given lists $\mathfrak{L}(u) \subseteq \boldsymbol{V}(H)$ for each $u \in \boldsymbol{V}(G)$, we derive the corresponding lists $\mathfrak{L}^{-1}(w) \subseteq \boldsymbol{V}(G)$ for each $w \in \boldsymbol{V}(H)$:

$$\mathfrak{L}^{-1}(w) = \Big\{u : u \in \boldsymbol{V}(G), w \in \mathfrak{L}(u)\Big\}.$$

An isomorphism $\pi : G \to H$ is list-compatible with $\mathfrak{L}$, if and only if the isomorphism $\pi^{-1} : H \to G$ is list-compatible with $\mathfrak{L}^{-1}$. Actually, we could work with both lists simultaneously. The similarity of ListIso and GraphIso also follows from the fact that many combinatorial algorithms for GraphIso can be modified for ListIso without any difficulty.

In Chapter 9, we study the complexity of ListIso for various restricted graph classes, an overview of our results in given in Fig. 6.5. We also consider special instances called ColoredGraphIso in which both graphs G and H are colored and we ask for existence of a color-preserving isomorphism, denoted $G \xrightarrow{c} H$. Unlike ListIso, the ColoredGraphIso problem is a well-known problem which is polynomial-time equivalent to GraphIso.

6.5.1 Our Results

We revive the study of list restricted graph isomorphism. The goal is to determine which techniques for GraphIso translate to ListIso. We believe that ListIso is a very natural computational problem, as evidenced by its application in [118, 120]. Further, its hardness results prove non-existence of robust algorithms for the graph isomorphism problem itself. For instance, it is believed that no NP-complete problem can be solved in quasipolynomial time. Therefore, Babai's algorithm [14] cannot be robust, i.e., it cannot be modified to solve ListIso in quasipolynomial time. To solve GraphIso efficiently (say, in polynomial time), one necessarily has to apply some non-robust techniques which does not generalize to ListIso.

The described algorithms for ListIso are a straightforward modification of previously known algorithms for the graph isomorphism problem. The main point of Chapter 9 is *not to develop new algorithmic techniques*, but to *classify known techniques for* GraphIso *from a different viewpoint.* This viewpoint is the robustness of algorithms with respect to ListIso, and our results give an insight into its meaning.

We prove the following three *informal results* in Chapter 9; see Fig. 6.5 for an overview:

Result 6.5.1. GI-*completeness results for* GraphIso *with polynomial-time reductions using vertex-gadgets imply* NP-*completeness for* ListIso.

For many classes $\mathcal{C}$ of graphs, it is known that GraphIso is equally hard for them as for general graphs, i.e., it is GI-complete. For instance, GraphIso is GI-complete for bipartite graphs, split and chordal graphs [262], chordal bipartite and strongly chordal graphs [346], trapezoid graphs [335], comparability graphs of dimension 4 [224], grid intersection graphs [345], line graphs [358], and self-complementary graphs [69].

The polynomial-time reductions are often done in a way that all graphs are encoded into $\mathcal{C}$, by replacing each vertex with a small vertex-gadget. (The constructions are quite simple, and the non-trivial part is to prove that the constructed graph belongs to $\mathcal{C}$.) As we prove in Theorem 9.2.1, such reductions using vertex-gadgets also translate to ListIso: they imply NP-completeness of ListIso for $\mathcal{C}$. For instance, ListIso is NP-complete for all graph classes mentioned above (Corollary 9.2.3).

Result 6.5.2. *The problem* ListIso *can be solved in polynomial-time for trees, planar graphs, interval graphs, circle graphs, permutation graphs, bounded genus graphs and bounded treewidth graphs.*

As a by-product, our paper gives an overview of the main combinatorial techniques involved in attacking the graph isomorphism problem. These combinatorial techniques for GraphIso are often robust and translate to ListIso in a straightforward way. Moreover, we can describe them more naturally with lists.

For example, the bottom-up linear-time algorithm for testing graph isomorphism of (rooted) trees translates to ListIso in Theorem 9.4.1, since it captures all possible isomorphisms. The key difference is that the algorithm for ListIso finds perfect matchings in bipartite graphs, in order to decide whether lists of several subtrees are simultaneously compatible; see Fig. 6.6b. We use the algorithm of Hopcroft and Karp [187], running in time $\mathcal{O}(\sqrt{n}m)$.

The algorithms for graph isomorphism of planar, interval, permutation and circle graphs based on tree decompositions and translate to ListIso, as we show in Theorems 9.5.3, 9.6.1, 9.6.2, and 9.6.3. Even more involved algorithms for graphs isomorphism of bounded genus and bounded treewidth graphs translate to ListIso in Theorems 9.7.1 and 9.8.5. The complexity for graphs with bounded rankwidth and graphs with excluded minors remains open, see Conclusions for details.

See Section 9.9 for discussions of k-WL and why it does not help in solving ListIso.

Result 6.5.3. *The problem* ListIso *is* NP*-complete for 3-regular colored graphs with all color classes of size at most 8 and with all lists of size at most 3.*

This result contrasts with two fundamental results using group theory techniques to solve graph isomorphism in polynomial time for graphs of bounded degrees [263] and bounded color classes [134]. Therefore, no robust algorithm solving graph isomorphism for these graph classes exists. In general, our impression is that group theory techniques do not seem to translate to ListIso since list-compatible automorphisms of a graph G do not form a subgroup of $\mathrm{Aut}(G)$. In Theorem 9.3.3, we prove Result 6.5.3 by describing a non-trivial modification of the original NP-hardness reduction of Lubiw [260].

6.6 Regular Graph Covers

The notion of *covering* originates in topology as a notion of local similarity of two topological spaces. For instance, consider the unit circle and the real line. Globally, these two spaces are not the same, they have different properties, different fundamental groups, etc. But when we restrict ourselves to a small part of the circle, it looks the

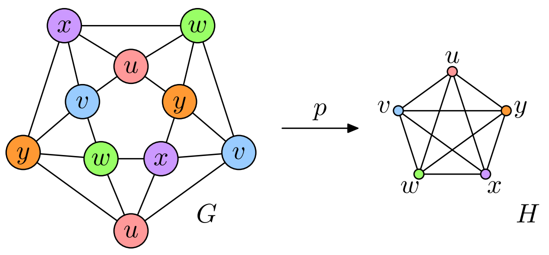


Figure 6.7: A regular covering projection p from a graph G to one of its quotients H. For every vertex $v \in \boldsymbol{V}(G)$, the image $p(v)$ is written in the circle.

same as a small part of the real line; more precisely the two spaces are locally homeomorphic, and thus they share the local properties. The notion of covering formalizes this property of two spaces being *locally the same*.

Suppose that we have two topological spaces: a big one G and a small one H. We say that G *covers* H if there exists an epimorphism called a *covering projection* $p : G \to H$ which locally preserves the structure of G. For instance, the mapping $p(x) = (\cos x, \sin x)$ from the real line to the unit circle is a covering projection. The existence of a covering projection ensures that G looks locally the same as H; see Fig. 6.7.

In Chapters 10 and 11, we study coverings of graphs in a more restricting version called *regular covering*, for which the covering projection is described by a semiregular action of a group. We say that G *regularly covers* H and H is a *(regular) quotient* of G, if there exists a semiregular subgroup Γ of $\mathrm{Aut}(G)$ such that $G/\Gamma \cong H$; see Section 10.1 for the formal definition.

6.6.1 Motivations for Regular Graph Covering

Suppose that G covers H and we have some information about one of the objects. How much knowledge does translate to the other object? It turns out that quite a lot, and this makes covering a powerful technique with many diverse applications. The big advantage of regular coverings is that they can be efficiently described and many properties easily translate between the objects. We sketch some applications now. See also [151].

Powerful Constructions. The reverse of covering called *lifting* can be applied to small objects in order to construct large objects of desired properties. For instance, the well-known Cayley graphs are large objects which can be described easily by a few elements of a group. Let G be a Cayley graph generated by elements $g_1, \dots, g_e$ of a group Γ. The vertices of G correspond to the elements of Γ and the edges are described by actions of $g_1, \dots, g_e$ on Γ by left multiplication; each g_i defines a permutation on Γ and we put edges along the cycles of this permutation. See Fig. 6.8 for examples. Cayley graphs were originally invented to study the structure of groups [52].

In the language of regular coverings, every Cayley graph G can be described as a lift of a one vertex graph H with e loops and half-edges attached labeled $g_1, \dots, g_e$. Regular covers can be viewed as a generalization of Cayley graphs where the small graph H may contain more than one vertex. For example, the famous Petersen graph

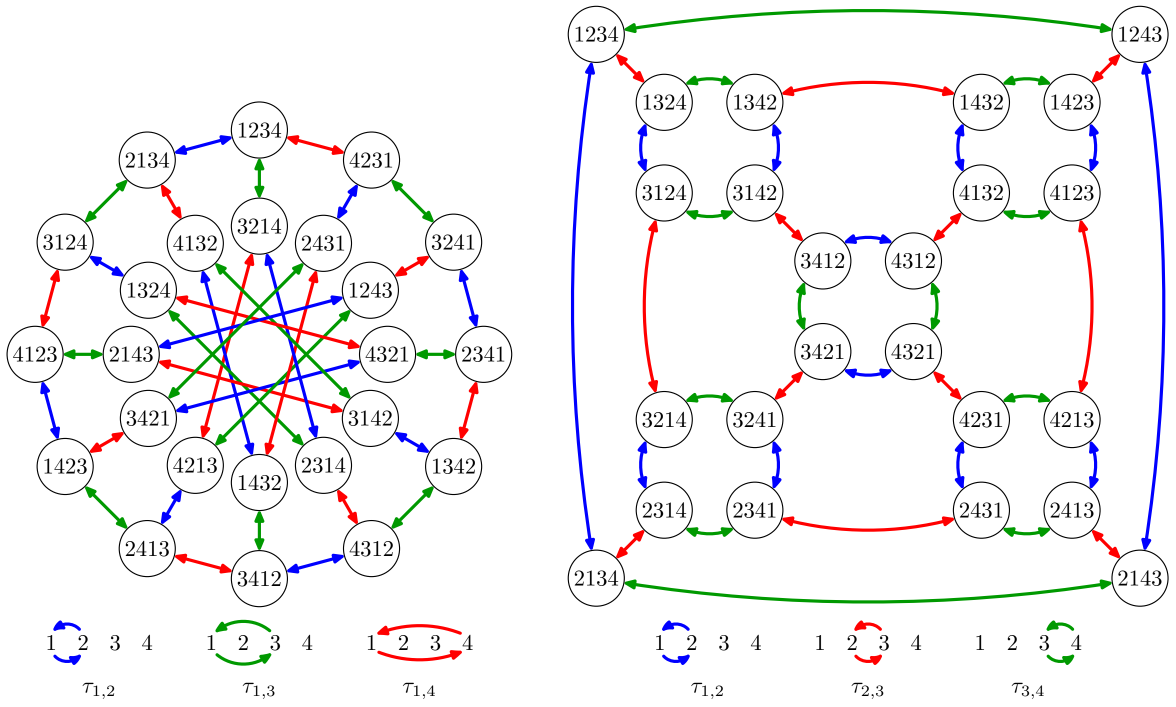


Figure 6.8: Two Cayley graphs created by different generators of the symmetric group $\mathbb{S}_4$ of all four-element permutations. By $\tau_{i,j}$, we denote the transposition of i and j.

can be constructed as a lift of a two-vertex graph H in Fig. 6.9a. These two vertices are necessary as it is known that Petersen graph is not a Cayley graph. Figure 6.9b shows a simple construction [274, 350] of the Hoffman-Singleton graph [182] which is a 7-regular graph with 50 vertices and diameter 2; see Fig. 6.10.

The Petersen and the Hoffman-Singleton graphs are extremal graphs for the *degree-diameter problem*: given integers d and k, find a maximal graph G with diameter d and degree k. In general, the size of G is not known. Many currently best constructions were obtained using the covering techniques [281].

Further applications employ the fact that nowhere-zero flows, vertex and edge colorings, eigenvalues and other graph invariants lift along a covering projection. Two main applications of constructions of lifts are the solution of the Heawood map col-

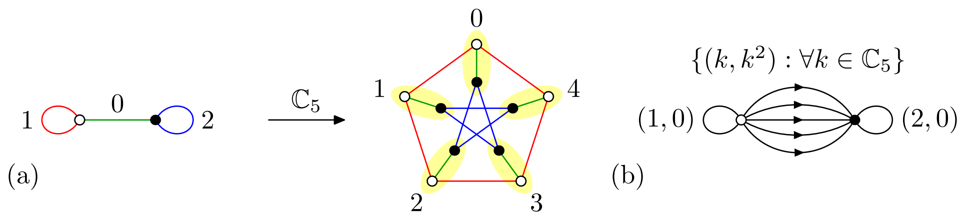


Figure 6.9: (a) A construction of the Petersen graph by lifting with the group $\mathbb{C}_5$. (b) By lifting the described graph with the group $\mathbb{C}_5^2$, we get the Hoffman-Singleton graph depicted in Fig. 6.10. The five parallel edges are labeled $(0,0)$, $(1,1)$, $(2,4)$, $(3,4)$ and $(4,1)$.

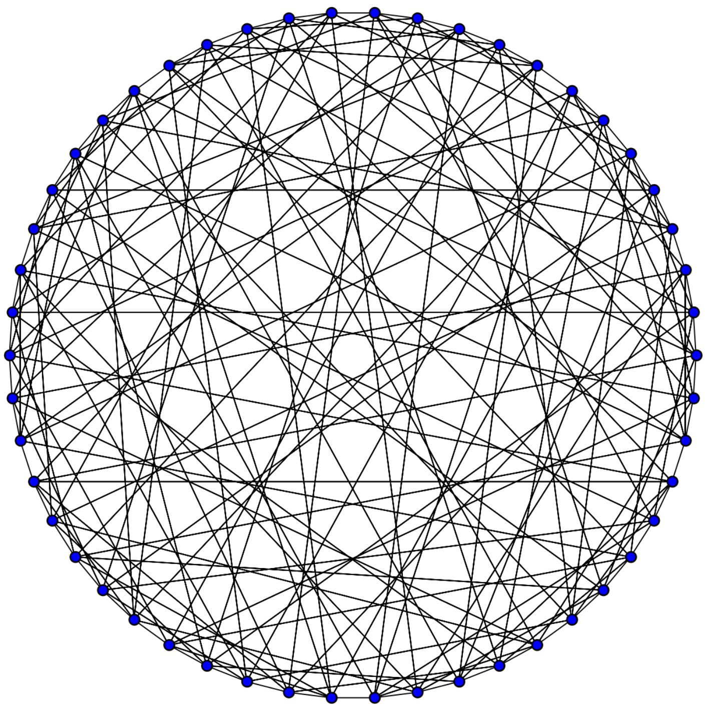


Figure 6.10: The figure is from Wikipedia. The Hoffman-Singleton graph G which is a 7-regular graph with 50 vertices and diameter 2. A surprisingly simple description of G is given in Fig. 6.9b which can be used to derive properties of G.

oring problem [303, 165] and constructions of arbitrarily large highly symmetrical graphs [28].

Models of Local Computation. These and similar constructions have many practical applications in designing highly efficient computer networks [106, 2, 21, 47, 48, 50, 166, 365], since these networks can be efficiently described/constructed and have many strong properties. In particular, networks based on covers of simple graphs allow fast parallelization of computation as described e.g. in [36, 5, 6].

Simplifying Objects. Regular coverings can be also applied in the opposite way, to project big objects onto smaller ones while preserving some properties. One way is to represent a class of objects satisfying some properties as quotients of the universal object of this property. For instance, this was used in the study of arc-transitive cubic graphs [153], and the key point is that universal objects are much easier to work with. This idea is commonly used in fields such as the theory of Riemann surfaces [113] and theoretical physics [205].

6.6.2 Structural Results

In Chapter 10, we fully describe the behaviour of regular covering with respect to 1-cuts and 2-cuts in G. It is closely related to the behaviour of 1- and 2-cuts under a semiregular action of a subgroup of the automorphism group $\mathrm{Aut}(G)$ of G. For 1-cuts, it is quite simple since $\mathrm{Aut}(G)$ fixes the central block. But the behaviour of regular covering on 2-cuts is complex. The main result Theorem 6.6.1 describes all possible quotients of some graph class if we understand quotients of 3-connected graphs in this class. It can be applied to planar graphs and it is used in Section 10.4 and in Chapter 11.

Expansions. Let G be a graph with a reduction series $G = G_0, \dots, G_r$ from Chapter 7. We aim to investigate how the knowledge of regular quotients of G_{i+1} can be used to construct all regular quotients of G_i. To do so, we introduce the reversal of the reduction called the *expansion*. If $H_{i+1} = G_{i+1}/\Gamma_{i+1}$, then the expansion produces H_i by replacing colored edges back by atoms. To do this, we have to understand how regular covering behaves with respect to atoms. Inspired by Negami [288], we show that each proper atom/dipole has three possible types of quotients that we call an *edge-quotient*, a *loop-quotient* and a *half-quotient*. The edge-quotient and the loop-quotient are uniquely determined but an atom may have many non-isomorphic half-quotients.

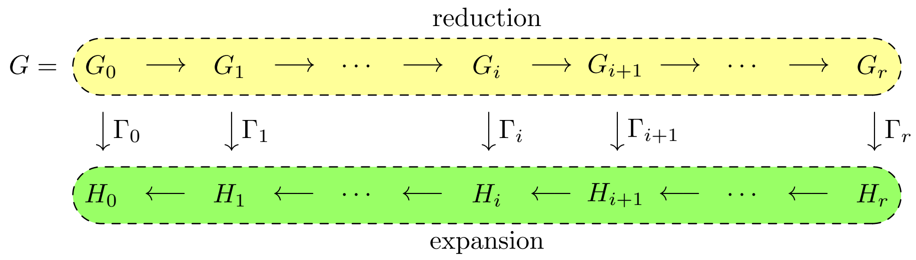
The constructed quotients contain colored edges, loops and half-edges corresponding to atoms. Each half-edge in H_{i+1} is created from a halvable edge if an automorphism of Γ_{i+1} fixes this halvable edge and exchanges its endpoints. Roughly speaking, the regular covering projection cuts the edge in half. The following theorem is our main result and it describes every possible expansion of H_{i+1} to H_i:

Theorem 6.6.1. *Let G_{i+1} be a reduction of G_i. Every quotient H_i of G_i can be constructed from some quotient H_{i+1} of G_{i+1} by replacing each edge, loop and half-edge of H_{i+1} by the subgraph corresponding to the edge-, the loop-, or a half-quotient of an atom of G_i, respectively.*

Suppose that some regular quotient of the primitive graph G_r is chosen, so $H_r = G_r/\Gamma_r$. The above theorem allows to describe all regular quotients H of G rising from H_r, as depicted in the diagram in Fig. 6.11.

Direct Proof of Negami Theorem. In 1988, Negami [288] proved that a connected graph H has a finite regular planar cover G if and only if H is projective planar. If the graph G is 3-connected, then $\mathrm{Aut}(G)$ is a spherical group (Sections 1.4.1 and 8.1). Therefore the conjecture can be easily proved using geometry, since a quotient of the sphere is either the disk, the sphere, or the projective plane. The hard part of the proof is to deal with graphs G containing 1-cuts and 2-cuts (Section 10.4).

Negami considered a minimal counterexample. In his proof, an essence of the crucial notion of an atom appears. A regular covering projection can behave on an


reduction

$$\begin{tikzcd}
G = G_0 \arrow[r] \arrow[d, "\Gamma_0"] & G_1 \arrow[r] \arrow[d, "\Gamma_1"] & \cdots \arrow[r] & G_i \arrow[r] \arrow[d, "\Gamma_i"] & G_{i+1} \arrow[r] \arrow[d, "\Gamma_{i+1}"] & \cdots \arrow[r] & G_r \arrow[d, "\Gamma_r"] \\
H_0 & H_1 \arrow[l] & \cdots \arrow[l] & H_i \arrow[l] & H_{i+1} \arrow[l] & \cdots \arrow[l] & H_r \arrow[l]
\end{tikzcd}$$

expansion

Figure 6.11: The reduction is on top, the expansion is on bottom. It holds that $H_i = G_i/\Gamma_i$ and Γ_i is a group extension of Γ_{i+1}.

atom in three different ways, and this understanding can be used to make the minimal counterexample smaller which forces a contradiction. In comparison, our work goes further and structurally describes all possible quotients H of a planar graph G. Quotients of 3-connected planar graphs can be described geometrically; see Section 10.4. Therefore, Theorem 6.6.1 describes all quotients of planar graphs.

6.6.3 Regular Covering Testing

Despite all described applications in Section 6.6.1, the computational complexity of regular covering was not yet studied. In this thesis, we initiate the study of the following computational problem.

Problem:	RegularCover
Input:	Connected graphs G and H.
Question:	Does G regularly cover H?

For a fixed graph H, the computational complexity of RegularCover was first asked as an open problem by Abello et al. [1]: “Are there graphs H for which the problem of determining if an input graph G is a regular cover of H is NP-hard?” Currently, no NP-hardness reduction is known for RegularCover, even when H is a part of the input.

The main algorithmic result shows that if G is planar, there exists no such graph H for which RegularCover is NP-complete. We use the complexity notation $f = \mathcal{O}^*(g)$ which omits polynomial factors. We establish the following FPT algorithm:

Theorem 6.6.2. *For planar graphs G, the* RegularCover *problem can be solved in time $\mathcal{O}^*(2^{\mathrm{e}(H)/2})$.*

6.6.4 Related Computational Problems

We discuss other computational problems related to RegularCover. The notion of regular covers builds a bridge between two seemingly different problems: Cayley graph recognition and the graph isomorphism problem.

Covering Testing. The complexity of general covering was widely studied before, pioneered by Bodlaender [36] in the context of networks of processors in parallel computing. Abello et al. [1] introduced the H-Cover problem which asks for an input graph G whether it covers a fixed graph H. Unless H is very simple, the problem turned out to be mostly NP-complete, the general complexity is still unresolved but the papers [240, 117] show that it is NP-complete for every r-regular graph H where $r \geq 3$. For a survey of the complexity results, see [121].

We try to understand how much the additional algebraic structure of regular covering changes the computational complexity. For planar inputs G, the change is significant: the problem H-Cover remains NP-complete for several small fixed graphs H (such as K_4, K_5) [29], while RegularCover can be solved in polynomial time for every fixed graph H by Theorem 6.6.2.

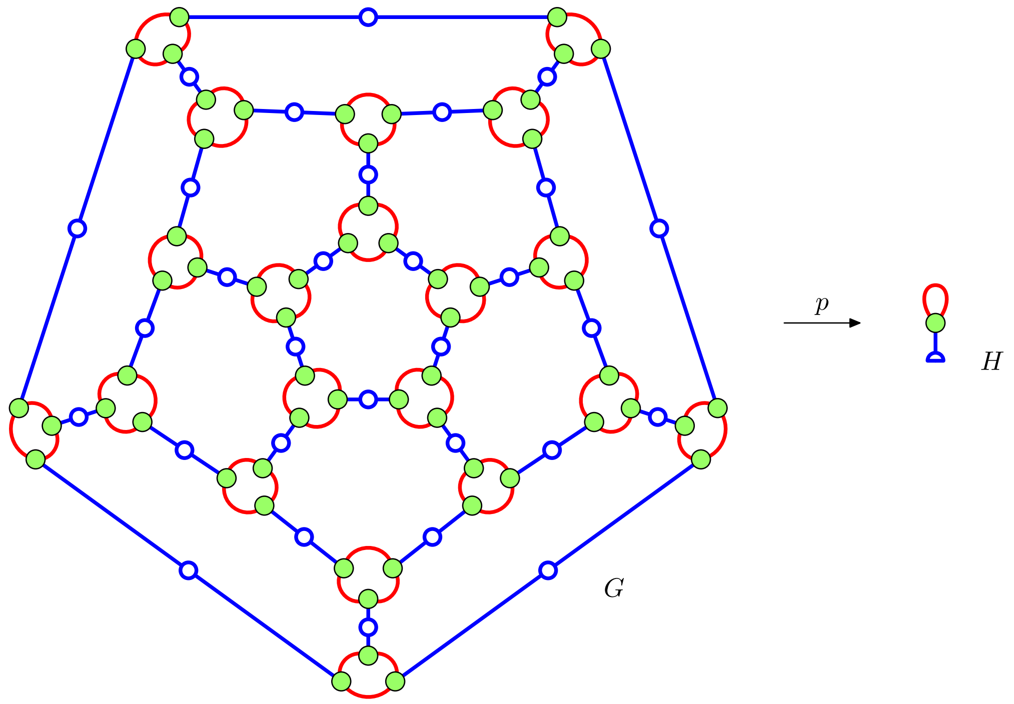


Figure 6.12: The truncated dodecahedron G with 60 vertices (depicted in green) is a planar Cayley graph of the group $\mathbb{A}_5$. It regularly covers the depicted one-vertex graph H. The blue edges (depicted with circles) forming a perfect matching are cut by the regular covering projection p in half, and correspond to a half-edge in H.

Cayley Graphs Testing. If the graph H consists of a single vertex with attached loops and half-edges, then RegularCover corresponds to Cayley graph recognition whose computational complexity is widely open. No hardness results are known and a polynomial-time algorithm is known only for recognition of circulant graphs [107]. In contrast, if H consists of a vertex with three half-edges attached, then G covers H if and only if G is a cubic 3-edge-colorable graph, so H-Cover is NP-complete [183].

The reader may notice that Theorem 6.6.2 gives a polynomial-time algorithm for recognize planar Cayley graphs. The input is a k-regular planar graph G, for $k \le 5$. We test RegularCover for all graphs H which a single vertex of degree k. The graph G is a Cayley graph if and only if it covers at least one of these graphs H. Unfortunately, finite planar Cayley graphs G are very limited: either G is a cycle, or G is 3-connected. Therefore, $\mathrm{Aut}(G)$ is a spherical group, and G is either finite (with $\boldsymbol{v}(G) \le 120$), representing one of the sporadic groups (for instance, Fig. 6.12), or very simple (a cycle, a prism, an antiprism, e.g.).

Graph Isomorphism Problem. The other extreme is when both graphs G and H have the same size, for which RegularCover is the famous graph isomorphism problem described in Section 6.4. Since RegularCover generalizes GraphIso, we cannot hope to solve it in polynomial time (unless solving GraphIso as well). It is natural to ask which results and techniques for GraphIso translate to RegularCover. Our results show that some technique for 3-connected decomposition translate, but the RegularCover problem is significantly more involved.

Theoretical motivation for studying the graph isomorphism problem is very similar to RegularCover. For practical instances, one can solve GraphIso very efficiently using various heuristics. But a polynomial-time algorithm working for all graphs is not known and it is very desirable to understand the complexity of GraphIso. It is known that testing graph isomorphism is equivalent to testing isomorphism of general mathematical structures [178]. The notion of isomorphism is widely used in mathematics when one wants to show that two seemingly different structures are the same. One proceeds by guessing a mapping and proving that this mapping is an isomorphism. The natural complexity question is whether there is a better algorithmic way to derive an isomorphism. Similarly, regular covering is a well-known mathematical notion which is algorithmically interesting and not understood.

Computing Automorphism Groups. A regular covering is described by a semiregular subgroup of the automorphism group $\mathrm{Aut}(G)$. Since a good understanding of $\mathrm{Aut}(G)$ is needed to solve RegularCover, it is closely related to AutGroup. See Sections 6.3 and 6.4.

Homomorphisms and CSP. Since regular covering is a locally bijective homomorphism, we give an overview of complexity results concerning homomorphisms. Hell and Nešetřil [180] studied the problem H-Hom which asks whether there exists a homomorphism between an input graph G and a fixed graph H. Their celebrated dichotomy result for simple graphs states that the problem H-Hom is polynomially solvable if H is bipartite, and it is NP-complete otherwise. Homomorphisms can be described in the language of constraint satisfaction (CSP), and the famous dichotomy conjecture [116] claims that every CSP is either polynomial-time solvable, or NP-complete.

6.6.5 Other Covering Problems

We introduce and discuss several other problems related to (regular) graph covering.

Lifting and Quotients. In the RegularCover problem, the input gives two graphs G and H. For the following problems, the input specifies only one graph and we ask for existence of the other graph:

Problem:	RegularLifting
Input:	A connected graph H and an integer k.
Question:	Does there exist a graph G regularly covering H such that $\lvert G\rvert = k\lvert H\rvert$?

Problem:	RegularQuotient
Input:	A connected graph G and an integer k.
Question:	Does there exist a graph H regularly covered by G such that $\lvert H\rvert = \frac{\lvert G\rvert}{k}$?

Concerning RegularLifting, the answer is always positive. The theory of covering describes a technique called voltage assignment which can be applied to generate all k-folds G. We do not deal with lifting in Chapters 10 and 11, but there are nevertheless many interesting computational questions with applications. For instance, is it possible to generate efficiently all (regular) lifts up to isomorphism? (This is non-trivial since different voltage assignments might lead to isomorphic graphs.) Or, does there exists a lift with some additional properties?

Concerning RegularQuotient, by Theorem 6.5.2, this problem is NP-complete even for the fixed $k = 2$. (We ask for existence of a half-quotient H of G which is equivalent to existence of a fixed-point free involution in $\mathrm{Aut}(G)$.) This hardness reduction can be easily generalized for every fixed even k, but the complexity remains open for odd values of k.

The reduction of Theorem 6.5.2 is from 3-satisfiability, each variable is represented by a variable gadget which is an even cycle attached to the rest of the graph. Each cycle has two possible regular quotients, either the cycle of half length (obtained by the 180° rotation), or the path of half length with attached half-edges (obtained by a reflection through opposite edges), corresponding to true and false values, respectively. These variable gadgets are attached to clause gadgets, and a quotient of a clause gadget can be constructed if and only if at least one literal of the clause is satisfied. This reduction does not imply NP-completeness for the RegularCover problem since the input also gives a graph H, so one can decode the assignment of the variables from it. See also Section 9.3.

k-Fold Covering. To simplify the RegularCover problem, instead of fixing H, we can fix the ratio $k = |G|/|H|$. (When G covers H, then k is an integer.) We get the following two problems for general and regular graph covers, respectively:

Problem:	k-Fold(Regular)Cover
Input:	Connected graphs G and H such that $|G| = k|H|$.
Question:	Does G (regularly) cover H?

For $k = 1$, both problems are equivalent to GraphIso. Bodlaender [36] proved that the k-FoldCover problem is GI-hard for every fixed k. The same reduction also works for k-FoldRegularCover, see Lemma 11.1.3. Chaplick et al. [57] proved NP-completeness of 3-FoldCover and their reduction can be easily modified for all $k > 3$.

The complexities of 2-FoldCover and k-FoldRegularCover for all $k \geq 2$ are open and very interesting. We note that for $k = 2$, every covering is a regular covering, so the problems 2-FoldRegularCover and 2-FoldCover are identical, and NP-hardness of 2-FoldCover would imply NP-hardness for RegularCover as well. On the other hand, if k-FoldRegularCover is not NP-complete for any value k, the k-FoldRegularCover problems would be natural generalizations of GraphIso.

6.6.6 Three Properties

Let $\mathcal{C}$ be a class of connected multigraphs. By $\mathcal{C}/\Gamma$ we denote the class of all regular quotients of graphs of $\mathcal{C}$ (note that $\mathcal{C} \subseteq \mathcal{C}/\Gamma$). For instance, when $\mathcal{C}$ is the class of planar graphs, then the class $\mathcal{C}/\Gamma$ is, by Negami Theorem [288], the class of projective planar graphs. We define the following three properties of $\mathcal{C}$:

(P1) The classes $\mathcal{C}$ and $\mathcal{C}/\Gamma$ are closed under taking subgraphs and under replacing connected components attached to 2-cuts by edges.
(P2) For a 3-connected graph $G \in \mathcal{C}$, all semiregular subgroups Γ of $\mathrm{Aut}(G)$ can be computed in polynomial time. Here by semiregularity, we mean that the action of Γ has no non-trivial stabilizers of the vertices. See Section 10.1.
(P3) Let G and H be 3-connected graphs of $\mathcal{C}/\Gamma$, possibly with colored and directed edges, and the vertices of G be equipped with lists. We can decide ListIso of G and H in polynomial time. (Where the list-compatible isomorphism respects orientations and colors of edges.)

As we prove in Lemma 11.5.1, these three properties are tailored for the class of planar graphs. (The proof of the property (P3) is non-trivial, following from Theorem 9.5.3.) The main reason to state (P1) to (P3) is to make explicitely clear which properties of planar graphs are necessary for our algorithm.

Since ListIso is NP-complete in general, we also use the restricted version with only ColoredGraphIso to highlight places where ListIso can be avoided:

(P3*) Let G and H be 3-connected graphs of $\mathcal{C}/\Gamma$, possibly with colored and directed edges, and the vertices of G and H are colored. We can decide Colored-GraphIso of G and H in polynomial time.

6.6.7 The Meta-algorithm

Chapter 11 studies the complexity of regular covering testing, based on our structural results described in Chapter 10. We establish the following algorithmic result:

Theorem 6.6.3. *Let $\mathcal{C}$ be a class of graphs satisfying (P1) to (P3). There exists an* FPT *algorithm for* RegularCover *for $\mathcal{C}$-inputs G in time $\mathcal{O}^*(2^{\mathrm{e}(H)/2})$.*

Since the assumptions (P1) to (P3) are satisfied for planar graphs (Lemma 11.5.1), we get Theorem 6.6.2. Notice that if the input graph G is 3-connected, using our assumptions the RegularCover problem can be trivially solved, by enumerating all its regular quotients and testing graph isomorphism with H. Babai [11] proved that to solve graph isomorphism, it is sufficient to solve graph isomorphism for 3-connected graphs. We wanted to generalize this result to regular covers, but handling 2-cuts is very complicated and we need the assumptions (P2) and (P3).

On the graph G, we apply the reduction series of Chapter 7. When the reductions reach a 3-connected graph, the natural next step is to compute all its quotients; there are polynomially many of them according to (P2). What remains is the most difficult part: to test for each quotient whether it corresponds to H after expansion. The structural results of Chapter 10 describe all possible expansions of these quotients. The issue is that there may be exponentially many different ways to expand the graph, all described in Theorem 6.6.1. Therefore, we have to test in a clever way whether it is possible to reach H. Our algorithm consists of several subroutines, most of which we can perform in polynomial time. Only one subroutine (finding a certain "generalized matching") we have not been able to solve in polynomial time; see Section 11.4.

This slow subroutine can be avoided in some cases:

Corollary 6.6.4. *If G is a 3-connected graph, if H is a 2-connected graph, or if $k = |G|/|H|$ is odd, then the meta-algorithm of Theorem 6.6.3 can be modified to run in polynomial time.*

Corollary 6.6.5. *Let $\mathcal{C}$ be a class of graphs satisfying (P1), (P2), and (P3*). There exists an algorithm listing for $\mathcal{C}$-inputs G all their regular quotients, with a polynomial-time delay.*

Theorem 6.5.2 implies that to solve the RegularCover problem in general, one has to work with both graphs G and H from the beginning. Our algorithm starts only with G and tries to match its quotients to H only in the end.

7 3-connected Reduction

This chapter contains:

- *7.1: Definition of Extended Graphs.* We define extended graphs with half-edges used in Part II.
- *7.2: Group Theory and Automorphism Groups of Graphs.* An overview of the main concepts from group theory.
- *7.3: Block Trees and Their Automorphisms.* Block trees decompose graphs into 2-connected subgraphs. They capture automorphism groups.
- *7.4: Structural Properties of Atoms.* We introduce atoms which are 3-connected components of the reduction.
- *7.5: Reduction Series and Reduction Trees.* We describe the reduction series which replaces atoms by colored edges. It is captured by the reduction tree.
- *7.6: Reduction Epimorphism.* We describe changes in automorphism groups by reductions, with a group epimorphism $\mathbf{\Phi}_i : \mathrm{Aut}(G_i) \to \mathrm{Aut}(G_{i+1})$.
- *7.7: Polynomial-time Algorithms.* We describe algorithms for computing the reduction series and the reduction tree.

`http://pavel.klavik.cz/orgpad/3conn_reduction.html`

7.1 Definition of Extended Graphs

In this section, we introduce extended graphs, used in Chapters 7, 8, 10, 11 and in Section 9.5.

The concept of regular graph covering, studied in Chapters 10 and 11, comes from topological graph theory where graphs are understood as 1-dimensional CW-complexes. This means that edges are represented by real open intervals, vertices are points, and the topological closure of an edge e is either a closed interval, or a simple cycle. In the first case, e joins two different vertices u and v incident to e. In the second case, e is incident just to one vertex v and e is a loop based at v. When one considers regular quotients of graphs, a third type of "edges" may appear [267]. For a non-trivial involution swapping the end-vertices of an edge e, the regular covering projection maps e to an "edge" whose one end is incident to a vertex while the other is free. Its topological closure is homeomorphic to a half-closed interval, and we call it a half-edge.

Definition of Extended Graphs. An *extended multigraph* G (or just a *graph* G) is a tuple $(\boldsymbol{H}, \boldsymbol{V}, \iota, \lambda)$, where

- $\boldsymbol{H}$ is a set of half-edges,
- $\boldsymbol{V}$ is a set of vertices,
- $\iota : \boldsymbol{H} \to \boldsymbol{V}$ is a partial function of incidence, and
- $\lambda : \boldsymbol{H} \to \boldsymbol{H}$ is an involution, pairing half-edges.

The set of edges $\boldsymbol{E}$ is formed by orbits of λ of size 2, while orbits of size 1 form standalone half-edges.

Each edge $\{h, \lambda h\}$ is one of the four kinds:

- a *standard edge* if $\iota(h) \neq \iota(\lambda h)$,
- a *loop* if $\iota(h) = \iota(\lambda h)$,
- a *pendant edge* where exactly one of $\iota(h)$ and $\iota(\lambda h)$ is not defined, and
- a *free edge* where both $\iota(h)$ and $\iota(\lambda h)$.

For (standalone) half-edges h, we have $h = \lambda h$ and it is called a *free half-edge* when $\iota(h) = \iota(\lambda h)$ is not defined. Standalone half-edges are mostly called half-edges and we do not distinguish between h and the orbit $\{h\}$ of size one.[1] See Fig. 7.1a.

Unless the graph is K_2, we remove all vertices of degree 1 while keeping both half-edges (one with ι not defined). Assuming that the original graph contains no pendant edges, this removal does not change the automorphism group. A pendant edge attached to v is called a *single pendant edge* if it is the only pendant edge attached to v. Most graphs considered in this and the following chapters are assumed to be connected, so they contain no free edges and half-edges. (Sometimes we consider subgraphs which may be disconnected and may contain them.)

[1]When the distinction is needed, some papers call the elements of $\boldsymbol{H}$ as darts or arcs while standalone half-edges are called half-edges or semiedges.

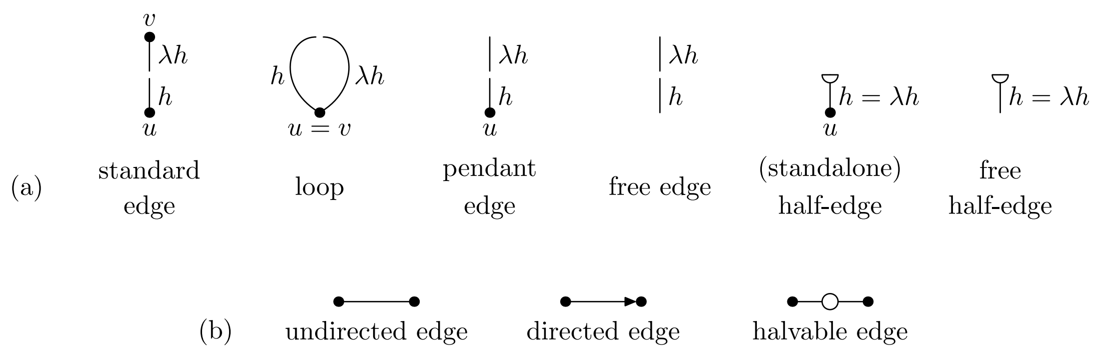


Figure 7.1: (a) Four kinds of edges and two kinds of half-edges are depicted. We highlight two half-edges composing each edge by a small gap, omitted in the remaining figures. To distinguish pendant edges from standalone half-edges, we end the latter by a half-circle.
(b) Three possible types for standard edges and loops (pendant edges are always undirected). We note that only halvable edges may be projected to standalone half-edges which corresponds to cutting the middle circle in half, explaining the symbol for half-edges.

When we work with several graphs, we use $\boldsymbol{H}(G)$, $\boldsymbol{V}(G)$, and $\boldsymbol{E}(G)$ to denote the sets of half-edges, vertices and edges of G, respectively. We denote $|\boldsymbol{H}(G)|$ by $\boldsymbol{h}(G)$, $|\boldsymbol{V}(G)|$ by $\boldsymbol{v}(G)$, $|\boldsymbol{E}(G)|$ by $\boldsymbol{e}(G)$. When G contains no standalone half-edges, clearly $\boldsymbol{h}(G) = 2\boldsymbol{e}(G)$.

Also, we consider graphs with colored edges and with three different edge types (directed edges, undirected edges and a special type called halvable edges used only in Chapters 10 and 11, see Fig. 7.1b). It might seem strange to consider such general objects. But when we apply reductions, we replace parts of the graph by edges and the colors encode isomorphism classes of replaced parts. Even if G is simple, the more general colored multigraphs are naturally constructed in the process of reductions.

7.2 Group Theory and Automorphism Groups of Graphs

In this section, we introduce key notions from group theory and properly define automorphism groups in the language of extended graphs.

7.2.1 Introduction to Group Theory

We quickly describe the main concepts from group theory and permutation group theory which are relevant for this thesis. For the undefined concepts and results, the reader is referred to [312]. For a visual introductory textbook, see [51].

We denote groups by Greek letters as for instance Ψ or Γ. Slightly abusing the notation, each group $\Psi = (\Psi, \cdot)$ consists of a set Ψ of elements and a binary operation $\cdot : \Psi \times \Psi \to \Psi$ which is associative and has a neutral element (denoted by 1 or id) and inverse elements. Group elements usually represent invertible transformations of some object (for instance, automorphisms of a graph) and the operation $\cdot$ is the composition. When written $\psi \cdot \sigma$, we first apply ψ and then σ. Figure 7.2 depicts Cayley graphs

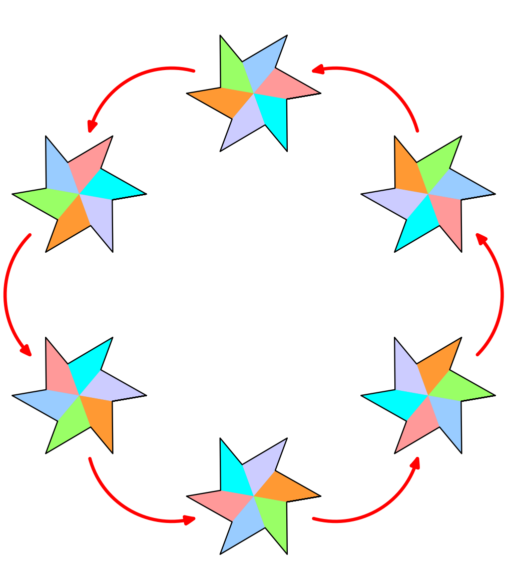
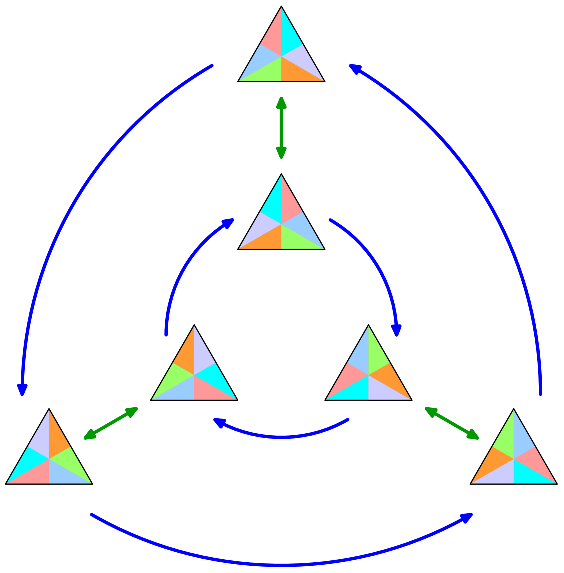



Figure 7.2: Cayley graphs of the groups of symmetric transformations of a fan (the group is isomorphic to $\mathbb{C}_6$) and of an equilateral triangle (the group is isomorphic to $\mathbb{D}_3$).

of symmetric transformations of two geometric objects. For a group Ψ, a *subgroup* Σ consists of a subset of elements which is closed under the group operation, and we denote this by $\Sigma \le \Psi$.

We use the following notation for some standard families of groups:

- $\mathbb{S}_n$ for the symmetric group of all n-element permutations,
- $\mathbb{C}_n$ for the cyclic group of integers modulo n,
- $\mathbb{D}_n$ for the dihedral group of the symmetries of a regular n-gon so $|\mathbb{D}_n| = 2n$, and
- $\mathbb{A}_n$ for the alternating group of all even n-element permutations.

We note that $\mathbb{D}_1 \cong \mathbb{C}_2$ and $\mathbb{D}_2 \cong \mathbb{C}_2^2$.

Group Actions. A group Ψ *acts* on a set S in the following way. Each element $g \in \Psi$ permutes the elements of S, and the *action* is described by a mapping $\cdot : \Psi \times S \to S$ where $1 \cdot x = x$ and $(gh) \cdot x = g \cdot (h \cdot x)$. Usually, actions satisfy further properties that arise naturally from the structure of S. In this thesis, by a group we usually mean a group of automorphisms of a graph acting on the set of half-edges.

For an action of Ψ on a set S, an orbit $[x]$ of $x \in S$ is $\{\psi(x) : \psi \in \Psi\} \subseteq S$. It follows that the orbits partition the set S. For $x \in S$, the stabilizer $\mathrm{Stab}(x)$ is the subgroup $\{\psi : \psi \in \Psi, \psi(x) = x\}$ of Ψ. For $A \subseteq S$, we distinguish the *point-wise stabilizer* $\{\psi : \psi \in \Psi, \psi(x) = x, \forall x \in A\}$ and the *set-wise stabilizer* $\{\psi : \psi \in \Psi, \psi(A) = A\}$. The action is called *semiregular* if it has no non-trivial stabilizers, i.e., if $\psi(x) = x$, then $\psi = \mathrm{id}$.

Suppose that a group Σ acts on two sets A and B. We say that the actions are *equivariant* if there exists an *equivariant map* $\varphi : A \to B$ which is a bijection and for every $\sigma \in \Sigma$, we have $\varphi(\sigma(x)) = \sigma(\varphi(x))$. Equivariance is an equivalence relation on orbits of the action of a group Σ, consisting of equivalent classes of equivariant orbits.

Group Quotients. If $\Sigma \le \Psi$, then Σ defines *left cosets* $\psi\Sigma = \{\psi \cdot \sigma : \sigma \in \Sigma\}$ and *right cosets* $\Sigma\psi = \{\sigma \cdot \psi : \sigma \in \Sigma\}$. One can imagine a coset as a shifted copy of the subgroup. All cosets are of equal size $|\Sigma|$ and the both types of cosets partition the elements of Ψ.

We would like to define the *quotient group* Ψ/Σ in which each element corresponds to one coset of Σ and the group operation is defined accordingly. This works only in the case when the left and the right cosets are equal, i.e., for each $\psi \in \Psi$, we have $\psi\Sigma = \Sigma\psi$. A subgroup satisfying this equality is called *normal*, denoted $\Sigma \trianglelefteq \Psi$. The operation which produces Ψ/Σ from Ψ is called *factorization* and Ψ is called an *group extension* of Σ by Ψ/Σ.

Simple Groups. The following idea was invented by Jordan. If $\Sigma \trianglelefteq \Psi$, then we can understand Ψ by studying two smaller groups: the subgroup Σ and the quotient group Ψ/Σ. We repeat the same idea on both of these groups, till they cannot be further simplified, and such groups are called *simple groups.* We obtain a composition series

$$\Psi = \Sigma_0 \triangleleft \Sigma_1 \triangleleft \cdots \triangleleft \Sigma_m = \{1\},$$

such that each quotient group Σ_{i-1}/Σ_i is simple. We can imagine these simple quotient groups as building blocks which construct Ψ; they play the role of prime numbers for groups.

The celebrated classification of finite simple groups describes all building blocks for finite groups. Therefore, it describes the structure of all finite groups. But this description gives just a part of information, since it is not clear how these building blocks are "put together" to form more complex groups. This is called the *group extension problem.* The problem to describe all extensions of a group Σ by Ψ/Σ is in general a hard problem. Below, we discuss when these extension can be described by semidirect products.

Group Homomorphism. For groups Ψ, Σ, a *group homomorphism* is a mapping of $\mathbf{\Phi} : \Psi \to \Sigma$ such that

$$\mathbf{\Phi}(\psi_1 \cdot \psi_2) = \mathbf{\Phi}(\psi_1) \cdot \mathbf{\Phi}(\psi_2), \qquad \psi_1, \psi_2 \in \Psi.$$

Each homomorphism $\mathbf{\Phi} : \Psi \to \Sigma$ defines two important subgroups: the *kernel* $\mathrm{Ker}(\mathbf{\Phi}) \trianglelefteq \Psi$ and the *image* $\mathrm{Im}(\mathbf{\Phi}) \le \Sigma$. A surjective homomorphism $\mathbf{\Phi}$ is called *epimorphism* and satisfies $\mathrm{Im}(\mathbf{\Phi}) = \Sigma$.

Homomorphism Theorem states that for every homomorphism $\mathbf{\Phi} : \Psi \to \Sigma$,

$$\mathrm{Im}(\mathbf{\Phi}) \cong \Psi/\mathrm{Ker}(\mathbf{\Phi});$$

see Fig. 7.3. It states that each homomorphism $\mathbf{\Phi}$ can be decomposed into two simpler homomorphisms: the *quotient homomorphism* $\mathbf{\Phi}_q : \Psi \to \Psi/\mathrm{Ker}(\mathbf{\Phi})$ (which is surjective) and the *embedding homomorphism* $\mathbf{\Phi}_e : \Psi/\mathrm{Ker}(\mathbf{\Phi}) \to \Sigma$ (which is injective). The quotient homomorphism $\mathbf{\Phi}_q$ compresses each coset of $\mathrm{Ker}(\mathbf{\Phi})$ into a single group element of $\Psi/\mathrm{Ker}(\mathbf{\Phi})$. The embedding homomorphism isomorphically maps $\Psi/\mathrm{Ker}(\mathbf{\Phi})$ into $\mathrm{Im}(\mathbf{\Phi})$.

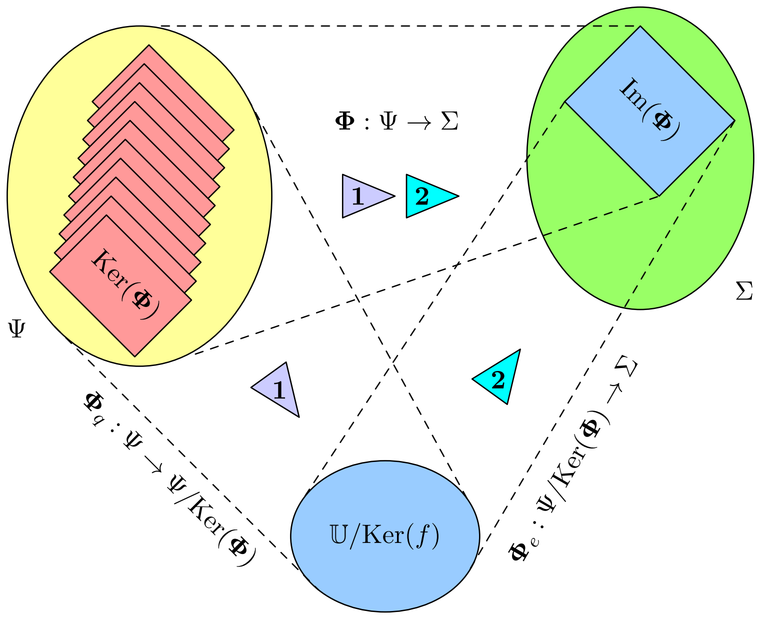

$\mathbf{\Phi} : \Psi \to \Sigma$
1
2
Ker($\mathbf{\Phi}$)
Im($\mathbf{\Phi}$)
Ψ
Σ
1
2
$\mathbf{\Phi}_q : \Psi \to \Psi/\mathrm{Ker}(\mathbf{\Phi})$
$\mathbb{U}/\mathrm{Ker}(f)$
$\mathbf{\Phi}_e : \Psi/\mathrm{Ker}(\mathbf{\Phi}) \to \Sigma$

Figure 7.3: Homomorphism $\mathbf{\Phi}$ can be decomposed in two parts: the quotient homomorphism $\mathbf{\Phi}_q$ and the embedding homomorphism $\mathbf{\Phi}_e$.

Overview of Group Products. Group products may be used to construct larger groups from smaller ones, which allows to describe certain group extensions. There are two basic approaches. *External group products* combine two groups Ψ and Σ into a new larger group whose elements are pairs (ψ, σ) for all $\psi \in \Psi$ and $\sigma \in \Sigma$. *Internal group products* describe an existing group Ψ by combining its subgroups.

Group products are often used to construct automorphism groups of graphs from automorphism groups of simpler graphs, e.g., the results described in Section 6.3.1. Also, in group theory, these products are often illustrated on graphs; see [312, 51]. For all of these products, if their groups are given in terms of permutation generators (and for external semidirect products, also generators of the image of φ), we can easily compute permutation generators of the product as well.

Direct Products. The *external direct product* $\Psi \times \Sigma$ has the Cartesian product $\Psi \times \Sigma$ equal the elements and the operation defined as

$$(\psi_1, \sigma_1) \cdot (\psi_2, \sigma_2) = (\psi_1 \cdot \psi_2, \sigma_1 \cdot \sigma_2).$$

In other words, $\Psi \times \Sigma$ consists of several identical copies of Ψ, each corresponding to one element of Σ, and vice versa. By Ψ^k, we denote the direct product of k groups Ψ. See Fig. 7.4 on the left.

Alternatively, a group Γ may be described by the *internal direct product* of normal subgroups $\Psi, \Sigma \trianglelefteq \Gamma$ which are *complementary*, meaning $\Psi \cap \Sigma = \{1\}$ and $\langle \Psi \cap \Sigma \rangle = \Gamma$. This condition implies that each $\gamma \in \Gamma$ is equal to $\psi \cdot \sigma$ for unique $\psi \in \Psi$ and $\sigma \in \Sigma$.

Semidirect Products. The direct product can be generalized by the *semidirect product*, and we first describe the *external semidirect product*. Given two groups Ψ and Σ, and a group homomorphism $\varphi \colon \Sigma \to \mathrm{Aut}(\Psi)$, we construct the semidirect

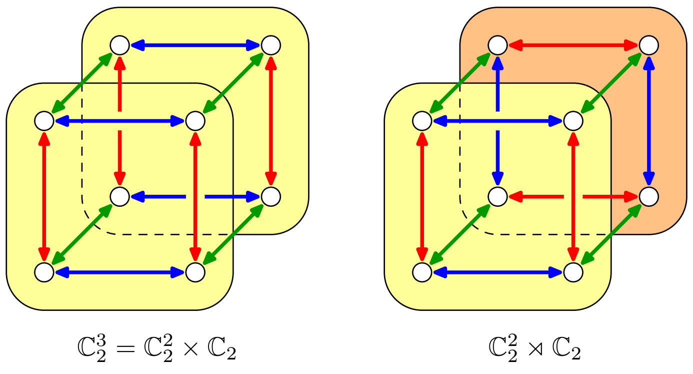

$\mathbb{C}_2^3 = \mathbb{C}_2^2 \times \mathbb{C}_2$
$\mathbb{C}_2^2 \rtimes \mathbb{C}_2$

Figure 7.4: On the left, the direct product $\mathbb{C}_2^2 \times \mathbb{C}_2$, consisting of two copies of $\mathbb{C}_2^2$, depicted in yellow. On the right, the semidirect product $\mathbb{C}_2^2 \rtimes_\varphi \mathbb{C}_2$, consisting of two different copies of $\mathbb{C}_2^2$ depicted in yellow ($\varphi(0) = \mathrm{id}$) and orange ($\varphi(1)$ exchanging the role of red and blue edges in $\mathbb{C}^2$).

product $\Psi \rtimes_\varphi \Sigma$ as follows. The elements are the Cartesian product $\Psi \times \Sigma$ and the operation is

$$(\psi_1, \sigma_1) \cdot (\psi_2, \sigma_2) = (\psi_1 \cdot \varphi(\sigma_1)(\psi_2), \sigma_1 \cdot \sigma_2).$$

In some statements, we omit φ, but it is always described in the proofs. The automorphism group $\mathrm{Aut}(\Psi)$ consists of all "isomorphic copies" of Ψ. Similarly as for the direct product, $\Psi \rtimes_\varphi \Sigma$ consists of several copies of Ψ, each corresponding to one element of Σ. But these copies are only isomorphic, not identical, and the mapping φ chooses one isomorphic copy of Ψ for each $\sigma \in \Sigma$. To define the group operation on $\Psi \rtimes_\varphi \Sigma$ correctly, φ has to be a group homomorphism. See Fig. 7.4 on the right.

For the *internal semidirect product*, we again have two complementary subgroups, but only one has to be normal. Let Γ be a group with complementary subgroups $\Psi \trianglelefteq \Gamma$ and $\Sigma \le \Gamma$. Then Γ can be constructed as $\Psi \rtimes \Sigma$. For the quotient Γ/Ψ, we have the normal subgroup Ψ. Therefore, the group extension Γ of Ψ by Γ/Ψ can be described by the semidirect product if and only if there exists a subgroup $\Sigma \le \Gamma$ which is complementary to Ψ. (We always have $\Sigma \cong \Gamma/\Psi$.)

This may be applied to a group homomorphism $\mathbf{\Phi} : \Gamma \to \Upsilon$ to construct Γ from $\mathrm{Ker}(\mathbf{\Phi})$ and from the complementary subgroup $\Sigma \le \Gamma$; see Fig. 7.5a. (It always follows that $\mathbf{\Phi}|_\Sigma : \Sigma \to \mathrm{Im}(\mathbf{\Phi})$ is an isomorphism.) We use this approach for the reduction epimorphism Φ_i in the proof of Proposition 7.6.4.

Wreath Products. In permutation group theory, a special type of semidirect products, called a *wreath product*, is often used. Suppose that Σ acts on $\{1, \dots, n\}$. The wreath product $\Psi \wr \Sigma$ is a shorthand for the semidirect product $\Psi^n \rtimes_\varphi \Sigma$ where φ is defined naturally by

$$\varphi(\sigma) = (\psi_1, \dots, \psi_n) \mapsto (\psi_{\sigma(1)}, \dots, \psi_{\sigma(n)}).$$

In this thesis, we always have $\Sigma = \mathbb{S}_n$ or $\Sigma = \mathbb{C}_n$ with the natural action $\{1, \dots, n\}$. In other words, we have n copies of Ψ and Σ permutes their indexes according to its action on $\{1, \dots, n\}$. Figure 7.5b shows an example. For more details, see [51, 312].

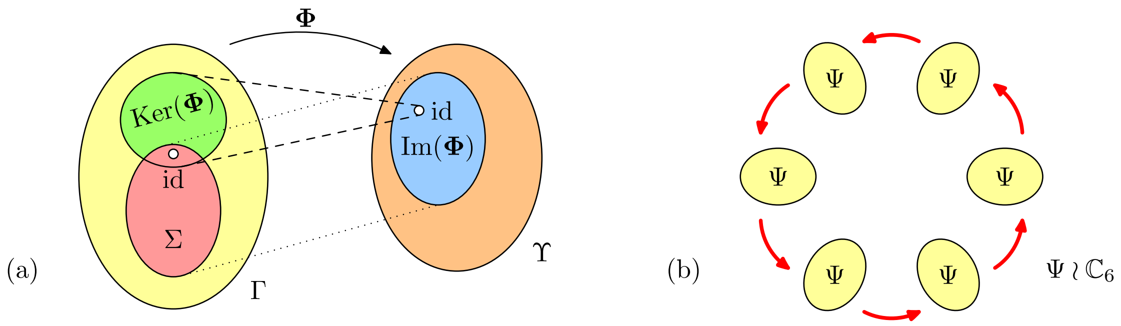


Figure 7.5: (a) The group Γ can be constructed from $\mathrm{Ker}(\boldsymbol{\Phi})$ and $\mathrm{Im}(\boldsymbol{\Phi})$ by a semidirect product if and only if there exists a subgroup $\Sigma \le \Gamma$ complementary to $\mathrm{Ker}(\boldsymbol{\Phi})$. (b) The wreath product $\Psi \wr \mathbb{C}_6$ is the semidirect product $\Psi^6 \rtimes \mathbb{C}_6$ such that the elements of $\mathbb{C}_6$ permute indexes in Ψ^6 as depicted.

Robinson [308] introduced *generalized wreath product* to describe automorphism groups of graphs by combining automorphism groups of their 2-connected blocks; see Section 7.3. The group Σ acts on $\{1,\dots,n\}$ and has k orbits $[x_1],\dots,[x_k]$ of size $\ell_1,\dots,\ell_k$. Let $\Psi_1,\dots,\Psi_k$ be arbitrary groups, one for each orbit. Then the generalized wreath product

$$(\Psi_1,\dots,\Psi_k) \wr \Sigma = (\Psi_1^{\ell_1} \times \cdots \times \Psi_k^{\ell_k}) \rtimes_\varphi \Sigma,$$

where $\varphi(\sigma)$ permute indexes in the same way as σ acts on $1,\dots,n$. When two orbits $[x_i]$ and $[x_j]$ are equivariant, then $\ell_i = \ell_j$ and the generalized wreath product can be simplified. We merge $[x_i]$ and $[x_j]$ in the action of Σ and use the group $\Psi_i \times \Psi_j$ for this merged orbit.

All semidirect products described in Section 6.3.1 are generalized wreath products. On the other hand, the semidirect products of the Jordan-like characterization of the automorphism groups of planar graphs, described in Chapter 8, are more involved and cannot be expressed using these generalized wreath products.

7.2.2 Automorphism Groups of Extended Graphs

An *automorphism* π is fully described by a permutation $\pi_h : \boldsymbol{H}(G) \to \boldsymbol{H}(G)$ preserving edges and incidences between half-edges and vertices, i.e., $\pi_h(\lambda h) = \lambda \pi_h(h)$ and $\pi_h(\iota(h)) = \iota(\pi_h(h))$ (where either both, or none are defined). Thus, π_h induces two permutations $\pi_v : \boldsymbol{V}(G) \to \boldsymbol{V}(G)$ and $\pi_e : \boldsymbol{E}(G) \to \boldsymbol{E}(G)$ connected together by the very natural property $\pi_e(uv) = \pi_v(u)\pi_v(v)$ for every $uv \in \boldsymbol{E}(G)$. Since we mostly consider connected graphs with at least one half-edge, the action of an automorphism π on the vertex set is induced by the action of π on half-edges. If G is a simple, then π is determined by the action on the vertices, as is expected. In most situations, we omit subscripts and simply use $\pi(u)$ or $\pi(uv)$.

We similarly define *isomorphisms* $\pi : G \to H$ between different graphs G and H, and we denote existence of an isomorphism by $G \cong H$.

In addition, for colored graphs with three edge types, we require that automorphisms and isomorphisms always preserves the colors, the edge types and the direction of oriented edges.

Automorphism Groups. All automorphisms of G form a group called the *automorphism group of* G, denoted by $\mathrm{Aut}(G)$. Each element $\pi \in \mathrm{Aut}(G)$ acts on G, permutes its vertices, edges and half-edges while it preserves edges and incidences between the half-edges and the vertices.

The *orbit* $[v]$ of a vertex $v \in \boldsymbol{V}(G)$ is the set of all vertices $\{\pi(v) \mid \pi \in \Psi\}$, and the orbits $[h]$ of a half-edge $h \in \boldsymbol{H}(G)$ and $[e]$ of an edge $e \in \boldsymbol{E}(G)$ are defined similarly as $\{\pi(h) \mid \pi \in \Psi\}$ and $\{\pi(e) \mid \pi \in \Psi\}$. Similarly, we consider stabilizers of vertices, edges, half-edges, and point-wise and set-wise stabilizers. We discuss semiregularity in the context of regular graph covers in Section 10.1.

Automorphism Groups of Disconnected Graphs. To illustrate the power of group products, we prove that the automorphism group of a graph can be constructed by direct and wreath products from automorphism groups of its connected components, due to Jordan [202]. Also, describing this simple proof in detail is helpful, because similar arguments are later used proofs of Proposition 7.6.4 and Lemma 10.3.4.

Theorem 7.2.1 (Jordan [202]). *If $G_1, \dots, G_n$ are pairwise non-isomorphic connected graphs and G is the disjoint union of k_i copies of each G_i, then*

$$\mathrm{Aut}(G) \cong \mathrm{Aut}(G_1) \wr \mathbb{S}_{k_1} \times \cdots \times \mathrm{Aut}(G_n) \wr \mathbb{S}_{k_n}.$$

Proof. Since the action of $\mathrm{Aut}(G)$ is independent on non-isomorphic components, it is clearly the direct product of factors, each corresponding to the automorphism group of one isomorphism class of components. It remains to show that if G consists of k isomorphic components $H_1, \dots, H_k$ of a connected graph H, then

$$\mathrm{Aut}(G) \cong \mathrm{Aut}(H) \wr \mathbb{S}_k.$$

An example is given in Fig. 7.6.

For $i > 1$, let $\sigma_{1,i}$ be an arbitrarily chosen isomorphism from H_1 to H_i, and we put $\sigma_{1,1} = \mathrm{id}$ and $\sigma_{i,j} = \sigma_{1,j}\sigma_{1,i}^{-1}$. Observe that each isomorphism from H_i to H_j can

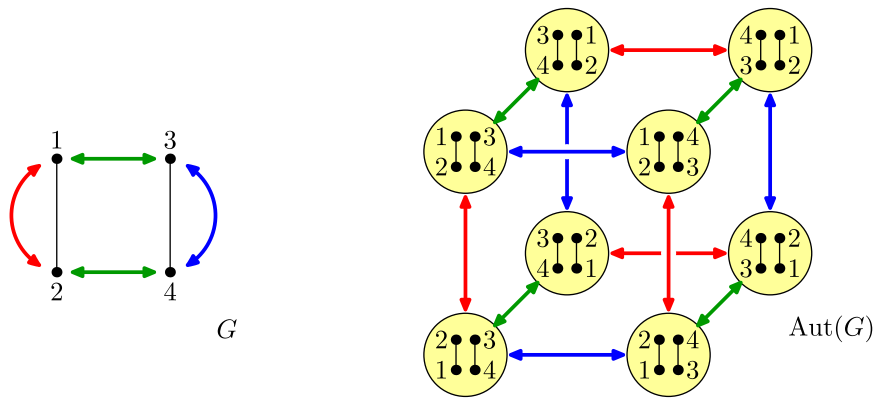


Figure 7.6: On the left, the graph G consisting of two copies of K_2, together with the action of three generators of $\mathrm{Aut}(G)$. On the right, the Cayley graph of $\mathrm{Aut}(G) \cong \mathbb{C}_2^2 \rtimes \mathbb{C}_2 \cong \mathbb{C}_2 \wr \mathbb{S}_2$. It is not $\mathbb{C}_2^3$ since applying the green generator exchanges the roles of the red and blue generators.

be decomposed into $\sigma_{i,j}$ and some automorphism of H_j. Let $\pi \in \mathrm{Aut}(G)$. It can be decomposed into a composition $\mu \cdot \sigma$ of two automorphisms. The automorphism σ permutes the components as in π, so when $\pi(H_i) = H_j$, then $\sigma|_{H_i} = \sigma_{i,j}$. The automorphism μ maps each component H_i to itself, so $\mu|_{H_i} = \pi|_{H_i} \cdot \sigma_{i,j}^{-1}$. We have $\pi = \mu \cdot \sigma$ since

$$\mu|_{H_i} \cdot \sigma|_{H_i} = \pi|_{H_i} \cdot \sigma_{i,j}^{-1}\sigma_{i,j} = \pi|_{H_i}.$$

The automorphisms μ can be bijectively identified with the elements of $\mathrm{Aut}(H)^k$ and the automorphisms σ with the elements of $\mathbb{S}_k$.

Let $\pi, \pi' \in \mathrm{Aut}(G)$. Consider the composition $\mu \cdot \sigma \cdot \mu' \cdot \sigma'$, we want to swap σ with μ' and rewrite this as a composition $\mu \cdot \hat{\mu} \cdot \hat{\sigma} \cdot \sigma$. Clearly the components are permuted in $\pi \cdot \pi'$ exactly as in $\sigma \cdot \sigma'$, so $\hat{\sigma} = \sigma$. On the other hand, $\hat{\mu}$ is not necessarily equal μ'. Let μ' be identified with the vector $(\mu'_1, \dots, \mu'_k) \in \mathrm{Aut}(H)^k$. Since μ' is applied after σ, it acts on the components permuted according to σ. Therefore $\hat{\mu}$ is constructed from μ' by permuting the coordinates of its vector by σ:

$$\hat{\mu} = (\mu'_{\sigma(1)}, \dots, \mu'_{\sigma(k)}).$$

This is precisely the definition of the wreath product, so $\mathrm{Aut}(G) \cong \mathrm{Aut}(H) \wr \mathbb{S}_k$. □

7.3 Block Trees and Their Automorphisms

The *block tree* T of G is a tree defined as follows. Consider all *articulations* in G and all maximal 2-connected subgraphs which we call *blocks* (with bridge-edges and pendant edges also counted as blocks). The block tree T is the incidence graph between articulations and blocks. For an example, see Fig. 7.7.

7.3.1 Properties of Automorphisms

Let $\pi \in \mathrm{Aut}(G)$. Every block B is mapped to a block $\pi(B)$ and every articulation u is mapped to an articulation $\pi(u)$ while the incidencies between blocks and articulations are preserved by π. Therefore, we get the following well-known fact:

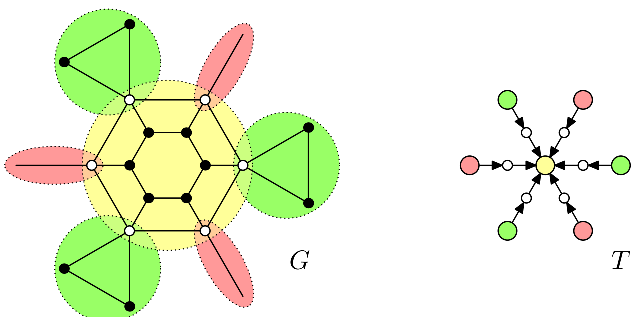


Figure 7.7: On the left, an example graph G with depicted blocks. On the right, the corresponding block tree T is depicted, rooted at the central block. The white vertices correspond to the articulations and the colored vertices to the blocks.

Lemma 7.3.1. *Every automorphism $\pi \in \mathrm{Aut}(G)$ induces a unique automorphism $\pi' \in \mathrm{Aut}(T)$. More precisely, this correspondence $\pi \mapsto \pi'$ defines a group homomorphism $\mathbf{\Phi} : \mathrm{Aut}(G) \to \mathrm{Aut}(T)$.* □

Central Articulations or Blocks. For a tree, its *center* is either the central vertex or the central pair of vertices of a longest path, depending on the parity of its length. For the block tree T, all leaves are blocks and each longest path is of an even length. Therefore, T has a central vertex which is either a *central articulation*, or a *central block* of G. Since every automorphism of a tree preserves its center, we get the following property:

Lemma 7.3.2. *Every automorphism $\pi \in \mathrm{Aut}(G)$ preserves the central articulation or the central block of G.* □

We orient the edges of the block tree T towards the central vertex, so the block tree becomes rooted. A *subtree* of the block tree is defined by any vertex different from the central vertex acting as the *root* and by all its predecessors. Every block B which is not the root is directed in T towards one articulation which we call the *outer articulation*. The other articulations in B are called the *inner articulations*. If T has a central block B, then B only contains inner articulations.

Let $T_1, \dots, T_k$ be the subtrees attached to the central vertex, corresponding to subgraphs $G_1, \dots, G_k$. We can relate $\mathrm{Aut}(G)$ and $\mathrm{Aut}(T)$ in the following way:

Lemma 7.3.3. *Let $\pi' \in \mathrm{Aut}(T)$ be the automorphism induced by $\pi \in \mathrm{Aut}(G)$, i.e, $\pi' = \mathbf{\Phi}(\pi)$. If $\pi'(T_i) = T_j$, then $G_i \cong G_j$.* □

7.3.2 Characterization of Automorphism Groups

The generalized wreath products were used by Robinson [308] to construct automorphism groups of connected graphs from automorphism groups of their blocks, generalizing Jordan's Theorem 7.2.1. Below, we sketch the main ideas since we generalize them in the rest of this chapter.

Let G be a connected graph. We want to determine $\mathrm{Aut}(G)$ from the block tree T and from the automorphism groups $\mathrm{Aut}(B)$ of its blocks B. The issue is that for a block B, not every automorphism has to be extendable to an automorphism of $\mathrm{Aut}(G)$. For instance, the central block B in Fig. 7.7 has the 60° rotation as an automorphism which cannot be extended to an automorphism of G since attached subtrees are non-isomorphic.

This issue can be resolved by a suitable coloring of articulations of B. We color two inner articulations the same if and only if their subtrees are isomorphic; see Fig. 7.8a. Further, if B is not a central block, we color the outer articulation by a special color. We consider only the color-preserving automorphism group $\mathrm{Aut}(B)$.

Lemma 7.3.4. *Every color-preserving automorphism of B can be extended to an automorphism of G.*

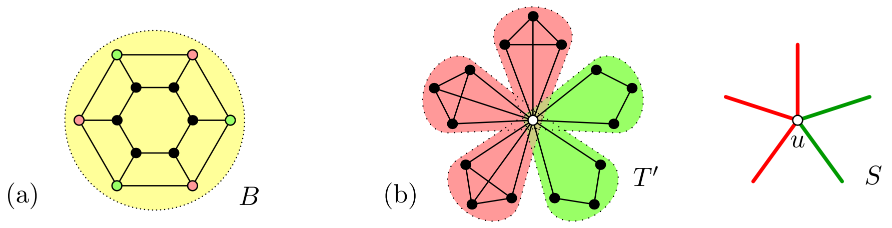


Figure 7.8: (a) The central block B from Fig. 7.7 with colored vertices according to the isomorphism classes of attached subtrees. We have $\mathrm{Aut}(B) \cong \mathbb{D}_3$.
(b) We represent an articulation u with attached subtrees by a star S with colored pendant edges. We have $\mathrm{Aut}(S) \cong \mathbb{S}_2 \times \mathbb{S}_3$.

Proof. Let $\pi \in \mathrm{Aut}(B)$. We construct $\pi' \in \mathrm{Aut}(G)$ extending π as follows. If B is not the central block, then outside the subtree of B, we put $\pi' = \mathrm{id}$. This choice is correct since $\mathrm{Aut}(B)$ stabilizes the outer articulation. It remains to define π' on the subtrees attached to B.

Let $u_1, \dots, u_k$ be one vertex-orbit of $\mathrm{Aut}(B)$ consisting of articulations having subtrees $T_1, \dots, T_k$ attached. Since these vertices have the same color, we have $T_i \cong T_j$. Similarly as in the proof of Theorem 7.2.1, we choose arbitrary isomorphisms $\sigma_{1,i}$ from T_i to T_j, and put $\sigma_{1,1} = \mathrm{id}$ and $\sigma_{i,j} = \sigma_{1,j}\sigma_{1,i}^{-1}$. If $\pi(u_i) = u_j$, we define $\pi'|_{T_i} = \sigma_{i,j}$. It is easy to prove that $\pi' \in \mathrm{Aut}(G)$. $\square$

Inductive Description. Similarly to Jordan's characterization of automorphism groups of trees (Theorem 6.3.1), we construct the automorphism group of G from the leaves to the root of T. We need to describe how to construct the automorphism group of a subtree assuming that we know automorphism groups of children subtrees. We need to deal with three cases: a subtree T' of a block, a subtree T' of an articulation, and the root vertex. In the first two cases, we only consider the stabilizer of the outer articulations denoted $\mathrm{Fix}(T')$. In the third case, the central block B (if it exists) has no outer articulation, so each automorphism of B may be extendable to an automorphism of G. The induction starts with the stabilizers of outer articulations for all leaf blocks of T.

Case 1: A subtree T' of a block B. We know stabilizers of attached subtrees of all inner articulations. We color the articulations of B as described above and let $\mathrm{Aut}(B)$ be the color-preserving automorphism group. To highlight that $\mathrm{Aut}(B)$ stabilizes the outer articulation, we denote it by $\mathrm{Fix}(B)$. Let $[u_1], \dots, [u_k]$ be the orbits of inner articulations of $\mathrm{Aut}(B)$, with attached subtrees $T'_1, \dots, T'_k$. Following a similar argument as in the proofs of Lemma 7.3.4 and Theorem 7.2.1, we get that $\mathrm{Fix}(T')$ is the following generalized wreath product:

$$\mathrm{Fix}(T') \cong \Big(\mathrm{Fix}(T'_1), \dots, \mathrm{Fix}(T'_k)\Big) \wr \mathrm{Fix}(B).$$

Case 2: A subtree T' of an articulation u. This case is even simpler. We have several subtrees attached to u for which we know stabilizers. The action of $\mathrm{Fix}(T')$ is independent on each isomorphism class of attached subtrees, so we get several factors combined by the direct product. Each factor corresponds to one isomorphism class which can be arbitrarily permuted. Suppose that we have k isomorphism classes, each consisting of ℓ_i subtrees T'_i. We get the same result as in Jordan's Theorem 7.2.1:

$$\mathrm{Fix}(T') \cong \mathrm{Fix}(T'_1) \wr \mathbb{S}_{\ell_1} \times \mathrm{Fix}(T'_2) \wr \mathbb{S}_{\ell_2} \times \cdots \times \mathrm{Fix}(T'_k) \wr \mathbb{S}_{\ell_k}.$$

But this case is also conceptually interesting for the rest of this chapter. Suppose that we would like to color the vertex u to encode isomorphism classes of the attached subtrees, to get a similar result as in Lemma 7.3.4 for blocks. Since multiple subtrees may be attached, we would have to use multiple colors with multiplicities on u. Instead, we choose to represent u with attached subtrees by a star S having u in the center and a pendant edge for each attached subtree. Then we color these pendant edges and obtain the color-preserving automorphism group $\mathrm{Aut}(S)$, for which a similar result as Lemma 7.3.4 holds; see Fig. 7.8b. To unify our approach, we represent outer articulations of a block B by attached colored single pendant edges. (Another reason becomes more apparent in Chapter 10.)

Case 3A: The central articulation. Same as Case 2.

Case 3B: The central block B. Similar as Case 1, but no outer articulation has to be stabilized. Therefore, we use $\mathrm{Aut}(B)$ instead of $\mathrm{Fix}(B)$ and we get:

$$\mathrm{Aut}(G) \cong \big(\mathrm{Fix}(T'_1), \dots, \mathrm{Fix}(T'_k)\big) \wr \mathrm{Aut}(B).$$

7.3.3 Why Not Just 2-connected Graphs?

In many situations, only connected graphs may be considered since the problems for disconnected graphs can be solved independently on each connected component. Similarly, block trees are used to reduce problems for connected graphs to problems for 2-connected graphs. For instance, Whitney [359] characterized planar graphs exactly as graphs whose blocks are planar. Similarly, graph isomorphism and computation of automorphism groups of graphs can be solved by graph isomorphism and computation of automorphism groups of their blocks.

In the rest of this chapter, we describe the tree decomposition of a connected graph in terms of 3-connected components, generalizing block trees. This decomposition captures automorphism groups and it is a non-trivial modification of previous results described in Section 6.2.3, e.g., [265, 340, 191, 95]. One of the key differences is that we assume connected graphs, but all previous results start with 2-connected graphs. We prefer starting with only connected graphs for the following reasons.

1. We can capture the full decomposition with only one tree, instead of having a tree for each block.
2. As illustrated, the automorphism group $\mathrm{Aut}(G)$ acts differently on the central articulation or block than on the attached subtrees to it. Therefore, we would need to work differently with the tree corresponding to the central block (if it exists) than with the trees corresponding to the remaining blocks. This unification makes Chapter 8 more understandable.
3. The original motivation for building our decomposition was the study of regular graphs covers. The issue is that regular quotients of 2-connected graphs are connected graphs but might not be 2-connected. Therefore, we cannot divide both

graphs G and H into blocks and work separately with them. This is described in more details in Chapters 10 and 11.

7.4 Structural Properties of Atoms

In this section, we introduce special inclusion-minimal subgraphs of G called atoms, also called 3-connected components. We investigate their structural properties with respect to the automorphism groups.

7.4.1 Definition and Basic Properties of Atoms

Let B be one block of G, so B is a 2-connected graph. Two vertices u and v form a *2-cut* $U = \{u, v\}$ if $B \setminus U$ is disconnected. We say that a 2-cut U is *non-trivial* if $\deg_B(u) \geq 3$ and $\deg_B(v) \geq 3$.

Lemma 7.4.1. *Let U be a 2-cut and let C be a component of $B \setminus U$. Then U is uniquely determined by C.*

Proof. If C is a component of $B \setminus U$, then U has to be the set of all neighbors of C in B. Otherwise B would not be 2-connected, or C would not be a component of $B \setminus U$. □

The Definition. We first define a set $\mathcal{P}$ of subgraphs of G called *parts* which are candidates for atoms:

- A *block part* is a subgraph of G induced by the vertices of the blocks of a subtree of the block-tree, non-isomorphic to a pendant edge.
- A *proper part* is a subgraph S of G defined by a non-trivial 2-cut U of a block B. The subgraph S consists of a connected component C of $G \setminus U$ (not $B \setminus U$) together with u and v and all edges between $\{u, v\}$ and C. In addition, we require that S does not contain the central block/articulation. Therefore, S consists a subgraph of B together with the vertices of the blocks of subtrees of all block-trees attached to C.
- A *dipole part* is any dipole defined as follows. Let u and v be two distinct vertices of degree at least three joined by at least two parallel edges. Then the subgraph induced by u and v is called a *dipole*.

The inclusion-minimal elements of $\mathcal{P}$ are called *atoms*. We distinguish *block atoms*, *proper atoms* and *dipoles* according to the type of the defining part. Block atoms are either stars of pendant edges called *star block atoms*, or pendant blocks possibly with single pendant edges attached to them called *non-star block atoms*. Also each proper atom is a subgraph of a block, together with some single pendant edges attached to it. Notice that a dipole part is by definition always inclusion-minimal, and therefore it is an atom. For an example, see Fig. 7.9. The above concepts of a proper atom and dipoles have their counter-parts in the literature, they are called pseudo-bricks and

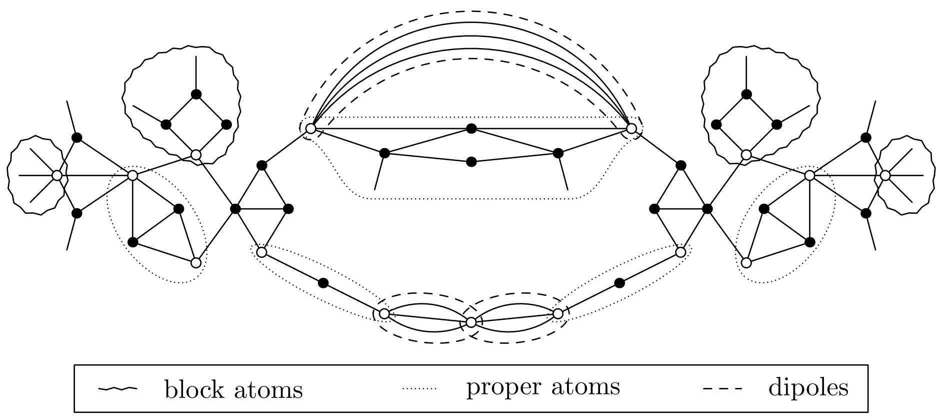


Figure 7.9: An example of a graph with denoted atoms. The white vertices belong to the boundary of some atom, possibly several of them.

bonds, respectively [352]. Some of the following properties and results can be found in literature, see [340, 344, 191, 78, 352] for instance.

We use the topological notation to denote the *boundary* ∂A and the *interior* $\mathring{A}$ of an atom A. If A is a dipole, we set $\partial A = \boldsymbol{V}(A)$. If A is a proper or block atom, we put ∂A equal to the set of vertices of A which are incident with an edge not contained in A. For the interior, we use the standard topological definition $\mathring{A} = A \setminus \partial A$ where we only remove the vertices ∂A, the edges adjacent to ∂A are kept in $\mathring{A}$ as pendant edges.

Note that $|\partial A| = 1$ for a block atom A, and $|\partial A| = 2$ for a proper atom or dipole A. The interior of a star block atom or a dipole is a set of free edges. Observe for a proper atom A that the vertices of ∂A are exactly the vertices $\{u, v\}$ of the non-trivial 2-cut used in the definition of proper parts. Also the vertices of ∂A of a proper atom are never adjacent in A. Further, no block or proper atom contains parallel edges; otherwise a dipole would be its subgraph, so it would not be inclusion minimal.

Remark. We define atoms with respect to the central block/articulation. It could be defined with respect to an arbitrary block/articulation of the block tree, and this is used later in this thesis.

7.4.2 Structure of Primitive Graphs and Atoms

We characterize the possible structure of atoms and primitive graphs, which explains why they are also called 3-connected components. For star block atoms and dipoles, their structure is clear by the definition. So it remains to describe the structure of non-star block atoms and proper atoms.

Primitive Graphs. A graph is called *primitive* if it contains no atoms. The following lemma characterizing primitive graphs can be alternatively obtained from the well-known theorem by Trakhtenbrot [340].[2]

[2]We consider K_1 with an attached single pendant edge as a graph with a central articulation.

Lemma 7.4.2. *Let G be a primitive graph. If G has a central block, then it is a 3-connected graph, a cycle C_n for $n \geq 2$, or K_2, or can be obtained from the aforementioned graphs by attaching single pendant edges to at least two vertices. If G has a central articulation, then it is K_1, possible with a single pendant edge attached.*

Proof. The graph G has a central block/articulation. All blocks attached to it have to be single pendant edges, otherwise G would contain a block atom. If G has a central articulation u, after removing all pendant edges, we get a single vertex u, so G is K_1, possibly with a single pendant edge with free ends attached. If G has a central block, after removing all pendant edges, we get the 2-connected graph B consisting of only the central block. We argue that B is one of the stated graphs.

Now, let u be a vertex of the minimum degree in B. If $\deg(u) = 1$, the graph B has to be K_2, otherwise it would not be 2-connected. If $\deg(u) = 2$, then either the graph B is a cycle C_n, or u is an inner vertex of a path connecting two vertices x and y of degree at least three such that all inner vertices are of degree two. But then this path is an atom, a contradiction. Finally, if $\deg(u) \geq 3$, then every 2-cut is non-trivial, and since B contains no atoms, it has to be 3-connected. $\square$

Clearly, the graphs mentioned in the statement are primitive; see Fig. 7.10.

Structure of Non-star Block Atoms. We call a graph *essentially 3-connected* if it is a 3-connected graph possibly with some single pendant edges attached to it. Similarly, a graph is called *essentially a cycle* if it is a cycle possibly with some single pendant edges attached to it. Similarly to the characterization of primitive graphs in Lemma 7.4.2, non-star block and proper atoms are either very simple, or almost 3-connected:

Lemma 7.4.3. *Every non-star block atom A is either K_2 with an attached single pendant edge, essentially a cycle, or essentially 3-connected.*

Proof. Clearly, the described graphs are possible non-star block atoms. Since A does not contain any smaller block atom, then A is 2-connected graph, possibly with some single pendant edges attached. By removing all single pendant edges, we get a 2-connected graph B, otherwise A contains a smaller block part, which is a smaller block part in G as well.

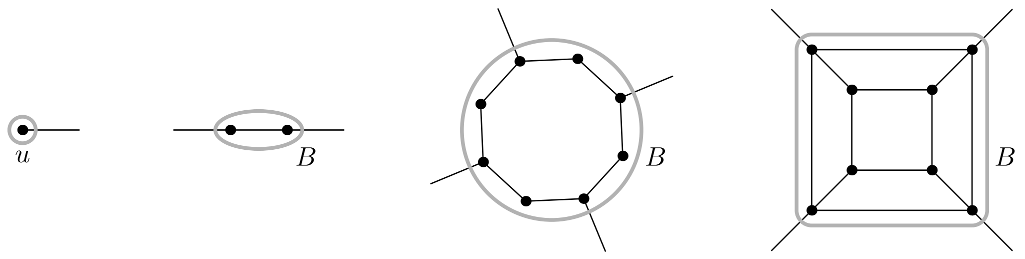


Figure 7.10: A primitive graph with a central articulation is K_1, and with a central block is either K_2, C_n, or a 3-connected graph, in all these cases with possible single pendant edges attached to it.

Let u be a vertex of the minimum degree in B. We have $\deg(u) > 0$, otherwise $B = K_1$ and $A = K_2$. If $\deg(u) = 1$, the graph B has to be K_2, otherwise it would not be 2-connected. If $\deg(u) = 2$, then either the graph B is a cycle C_n, or u is an inner vertex of a path connecting two vertices x and y of degree at least three such that all inner vertices are of degree two. But then this path determines a proper atom in B which is also a proper atom in G, a contradiction. Finally, if $\deg(u) \geq 3$, then every 2-cut is non-trivial, and since B contains no proper atoms, it has to be 3-connected. □

Structure of Proper Atoms. Let A be a proper atom with $\partial A = \{u, v\}$. We define the *extended proper atom* A^+ as A with the additional edge uv.

Lemma 7.4.4. *For every proper atoms A, the extended proper atom A^+ is either essentially a cycle, or essentially 3-connected.*

Proof. Clearly, the described graphs are possible extended proper atoms A^+. Notice that A^+ consists a 2-connected graph, possibly with single pendant edges attached, otherwise A contains a smaller block part. By removing all single pendant edges, we get a 2-connected graph B^+, otherwise A^+ contains a smaller block part. Let $\partial A = \{u, v\}$, we have $\deg(u) \geq 2$ and $\deg(v) \geq 2$ in A^+ (and their degrees are preserved in B^+).

Let w be a vertex of the minimum degree in B^+. We have $\deg(w) > 1$, otherwise A again contains a smaller block part. If $\deg(w) = 2$, then either the graph B^+ is a cycle C_n, or w is an inner vertex of a path connecting two vertices x and y of degree at least three such that all inner vertices are of degree two. But then this path is a proper atom in A^+. It corresponds to a proper atom in the original graph since the edge uv in A^+ corresponds to some path in G, so we get a contradiction with the minimality of A. Finally, if $\deg(w) \geq 3$, then every 2-cut is non-trivial, and since B^+ contains no atoms, it has to be 3-connected. □

For all atoms A, single pendant edges are always attached to $\mathring{A}$.

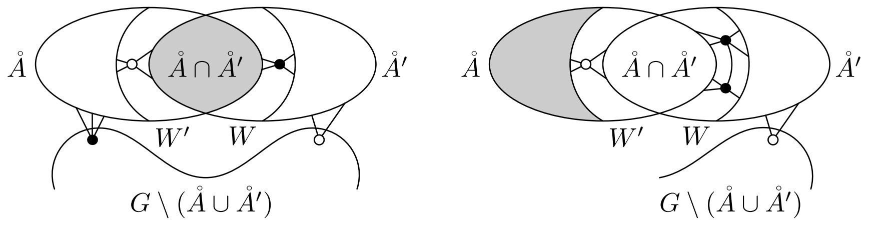
Lemma 7.4.5. *Let A be an essentially 3-connected graph, and we construct B from A by removing the single pendant edges of A. Then* $\mathrm{Aut}_{\partial A}(A)$ *is a subgroup of* $\mathrm{Aut}_{\partial B}(B)$.

Proof. These single pendant edges behave like markers, giving a 2-partition of $\boldsymbol{V}(G)$ which $\mathrm{Aut}_{\partial A}(A)$ has to preserve. □

7.4.3 Non-overlapping Atoms

Our goal is to replace atoms by edges, and so it is important to know that the atoms cannot overlap too much. The reader can see in Fig. 7.9 that the atoms only share their boundaries. This is true in general, and we are going to prove it in two steps.

Lemma 7.4.6. *The interiors of distinct atoms are disjoint.*



$\mathring{A}$
$\mathring{A} \cap \mathring{A}'$
$\mathring{A}'$
W'
W
$G \setminus (\mathring{A} \cup \mathring{A}')$
$\mathring{A}$
$\mathring{A} \cap \mathring{A}'$
$\mathring{A}'$
W'
W
$G \setminus (\mathring{A} \cup \mathring{A}')$

Figure 7.11: We depict the vertices of ∂A in black and the vertices of $\partial A'$ in white. In both cases, we find a subset of A belonging to $\mathcal{P}$ (highlighted in gray).

Proof. For contradiction, let A and A' be two distinct atoms with non-empty intersections of $\mathring{A}$ and $\mathring{A}'$. First suppose that one of them, say A, is a block atom. Then A corresponds to a subtree of the block-tree which is attached by an articulation u to the rest of the graph. If A' is a block atom then it corresponds to some subtree, and we can derive that $A \subseteq A'$ or $A' \subseteq A$. If A' is a dipole, then it is a subgraph of a block, and thus subgraph of A. If A' is a proper atom, it is defined with respect to some block B. If B belongs to the subtree corresponding to A, then $A' \subsetneq A$. Otherwise, a subtree of blocks containing A is attached to A', so $A \subsetneq A'$. In both cases, we get contradiction with the minimality. Similarly, if one of the atoms is a dipole, we can easily argue contradiction with the minimality.

The last case is that both A and A' are proper atoms. Since the interiors are connected and the boundaries are defined as neighbors of the interiors, it follows that both $W' = A \cap \partial A'$ and $W = A' \cap \partial A$ are nonempty. We have two cases according to the sizes of these intersections depicted in Fig. 7.11.

If $|W| = |W'| = 1$, then $W \cup W'$ is a 2-cut separating $\mathring{A} \cap \mathring{A}'$ which contradicts the minimality of A and A'. If, without loss of generality, $|W| = 2$, then there is no edge between $\mathring{A} \setminus (\mathring{A}' \cup W')$ and the remainder of the graph $G \setminus (\mathring{A} \cup \mathring{A}')$. Therefore, $\mathring{A} \setminus (\mathring{A}' \cup W')$ is separated by a 2-cut W' which again contradicts the minimality of A. We note that in both cases the constructed 2-cut is non-trivial since it is formed by vertices of non-trivial cuts ∂A and $\partial A'$. □

Next we show a stronger version of the previous lemma which states that two atoms can intersect only in their boundaries.

Lemma 7.4.7. *Let A and A' be two atoms. Then $A \cap A' = \partial A \cap \partial A'$.*

Proof. We already know from Lemma 7.4.6 that $\mathring{A} \cap \mathring{A}' = \emptyset$. It remains to argue that, say, $\mathring{A} \cap \partial A' = \emptyset$. If A' is a block atom, then $\partial A'$ is the articulation separating A. If A contains this articulation as its interior, it also contains A' as its interior, contradicting $\mathring{A} \cap \mathring{A}' = \emptyset$. Similarly, if A is a block atom, then A' has to be contained in $\mathring{A}$ or vice versa which again contradicts $\mathring{A} \cap \mathring{A}' = \emptyset$.

It remains to argue the case when both A and A' are proper atoms or dipoles. Let $\partial A = \{u, v\}$ and $\partial A' = \{u', v'\}$. First we deal with dipoles. When A is a dipole, it holds since $\mathring{A}$ contains no vertices. If A' is a dipole and A is a proper atom with $u' \in \mathring{A}$, then also the edges of A' belong to A and $A' \subsetneq A$, contradicting the minimality.

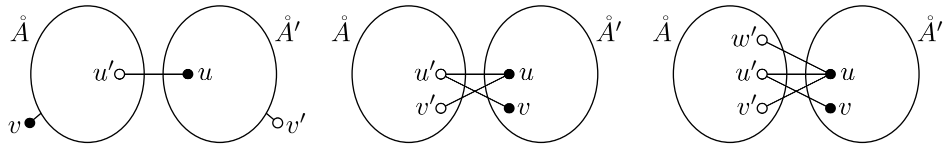


Figure 7.12: An illustration of the main steps of the proof of Lemma 7.4.7.

We conclude with the remaining case that both A and A' are proper atoms. Recall that ∂A is defined as neighbors of $\mathring{A}$ in G, and that $\partial A'$ are neighbors of $\mathring{A}'$ in G. The proof is illustrated in Fig. 7.12.

Suppose for contradiction that $\mathring{A} \cap \partial A' \neq \emptyset$ and let $u' \in \mathring{A}$. By definition, u' has at least one neighbor in $\mathring{A}'$, and since $\mathring{A} \cap \mathring{A}' = \emptyset$, this neighbor does not belong to $\mathring{A}$. Therefore, without loss of generality, we have $u \in \mathring{A}'$ and $uu' \in \boldsymbol{E}(G)$. Since A is a proper atom, the set $\{u', v\}$ is not a 2-cut, so there is another neighbor of u in $\mathring{A}$, which has to be equal v'. Symmetrically, u' has another neighbor in $\mathring{A}'$ which is v. So $\partial A \subseteq \mathring{A}'$ and $\partial A' \subseteq \mathring{A}$. If $\partial A = \mathring{A}'$ and $\partial A' = \mathring{A}$, the graph is K_4 (since the minimal degree of cut-vertices is three) which contradicts existence of 2-cuts and atoms. If for example $\mathring{A} \neq \partial A'$, then $\partial A'$ does not cut a subset of $\mathring{A}$, so there exists $w' \in \mathring{A}$ which is a neighbor of $\mathring{A}'$, which contradicts that $\partial A'$ cuts $\mathring{A}'$ from the rest of the graph. □

7.4.4 Symmetry Types of Atoms

We distinguish two symmetry types of atoms which describe how symmetric each atom is. For an atom A, we denote by $\mathrm{Aut}_{\partial A}(A)$ the setwise stabilizer of ∂A. If A is a block atom, then it is by definition *symmetric*. Let A be a proper atom or dipole with $\partial A = \{u, v\}$. Then we distinguish the following two symmetry types, see Fig. 7.13:

- *A symmetric atom A.* There exists an automorphism in $\mathrm{Aut}_{\partial A}(A)$ which exchanges u and v.
- *An asymmetric atom A.* The atom A which is not symmetric.

When an atom is reduced, we replace it by an edge carrying the type. Therefore we work with multigraphs with two edge types: *undirected edges* and *directed edges.* For these multigraphs, we naturally consider only the automorphisms which preserve these edge types and of course the orientation of directed edges, and we use this generalized definition to define symmetry types of their atoms.

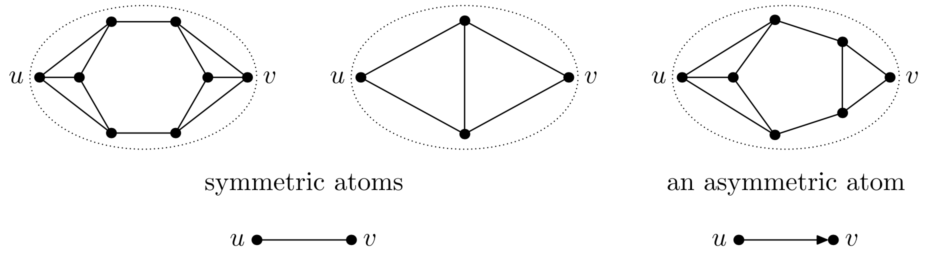


Figure 7.13: The two symmetry types of atoms and the corresponding edge types which we use in the reduction.

Action of Automorphisms on Atoms. We show a simple lemma which states how automorphisms behave with respect to atoms.

Lemma 7.4.8. *Let A be an atom and let $\pi \in \mathrm{Aut}(G)$. Then the following holds:*

(a) The image $\pi(A)$ is an atom isomorphic to A. Further $\pi(\partial A) = \partial\pi(A)$ and $\pi(\mathring{A}) = \mathring{\pi}(A)$.
(b) If $\pi(A) \neq A$, then $\pi(\mathring{A}) \cap \mathring{A} = \emptyset$.
(c) If $\pi(A) \neq A$, then $\pi(A) \cap A = \partial A \cap \partial\pi(A)$.

Proof. (a) Every automorphism permutes the set of articulations and non-trivial 2-cuts. (Recall the definition from the first paragraph of Section 7.4.1.) So $\pi(\partial A)$ separates $\pi(\mathring{A})$ from the rest of the graph. It follows that $\pi(A)$ is an atom, since otherwise A would not be an atom. And π clearly preserves the boundaries and the interiors.

For the rest, (b) follows from Lemma 7.4.6 and (c) follows from Lemma 7.4.7. □

It follows that every automorphism $\pi \in \mathrm{Aut}(G)$ gives a permutation of atoms and $\mathrm{Aut}(G)$ induces an action on the set of all atoms.

7.5 Reduction Series and Reduction Trees

In this section, we describe structural properties of reduction series. The reduction initiates with a graph G and produces a sequence of graphs $G = G_0, G_1, \dots, G_r$. To produce G_{i+1} from G_i, we find the collection of all atoms $\mathcal{A}$ in G_i and replace each of them by an edge of the corresponding type. We stop after r steps when a primitive graph G_r containing no further atoms is reached. We call this sequence of graphs starting with G and ending with a primitive graph G_r as the *reduction series* of G.

Definition of Reduction. The reduction produces a series of graphs $G = G_0, \dots, G_r$, by replacing atoms with colored edges encoding isomorphism classes and by edge types encoding symmetry types of atoms.

Remark: In what follows, we work with multigraphs with colored directed and undirected edges. For every automorphism/isomorphism, we require that it preserves colors, edge types and direction of oriented edges.

We note that the results established in Section 7.4 transfer to colored graphs and colored atoms without any problems.

For a graph G_i, we find the collection of all atoms $\mathcal{A}$. Two atoms A and A' are isomorphic if there exists an isomorphism which maps ∂A to $\partial A'$. We obtain isomorphism classes for the set of all atoms $\mathcal{A}$ of G_i such that A and A' belong to the same class if and only if $A \cong A'$. To each isomorphism class, we assign one new color not yet used in the graph. The graph G_{i+1} is constructed from G_i by replacing each atom in $\mathcal{A}$ by an edge as follows:

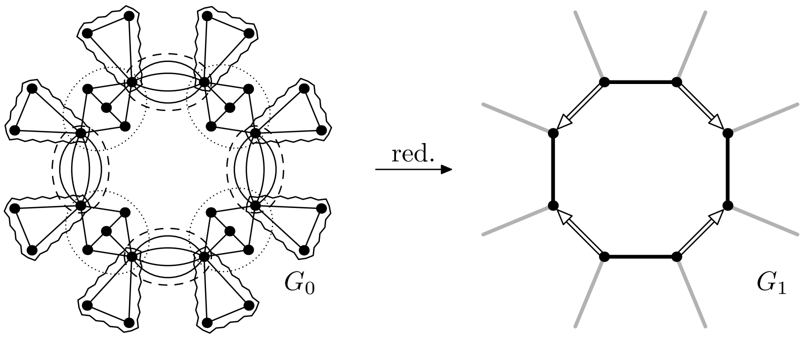


Figure 7.14: On the left, we have a graph G_0 with three isomorphism classes of atoms, each having four atoms. The dipoles and the block atoms are symmetric and the proper atoms are asymmetric. We reduce G_0 to G_1 which is an eight cycle with single pendant edges, with four black undirected edges replacing the dipoles, four gray undirected edges replacing the block atoms, and four white directed edges replacing the proper atoms. The reduction series ends with G_1 since it is primitive. Notice the consistent orientation of the directed edges.

- *A block atom* A with $\partial A = \{u\}$ is replaced by a pendant edge attached to u of the color assigned to the isomorphism class containing A.
- *A proper atom or a dipole* A with $\partial A = \{u, v\}$, which is symmetric or assymmetric, is replaced by a new undirected or directed edge uv, respectively, of the color assigned to the isomorphism class containing A. It remains to say that for each isomorphism class of asymmetric atom, we consistently choose an arbitrary orientation of the directed edges replacing these atoms.

For an example of the reduction, see Fig. 7.14.

According to Lemma 7.4.7, the replaced interiors of the atoms of $\mathcal{A}$ are pairwise

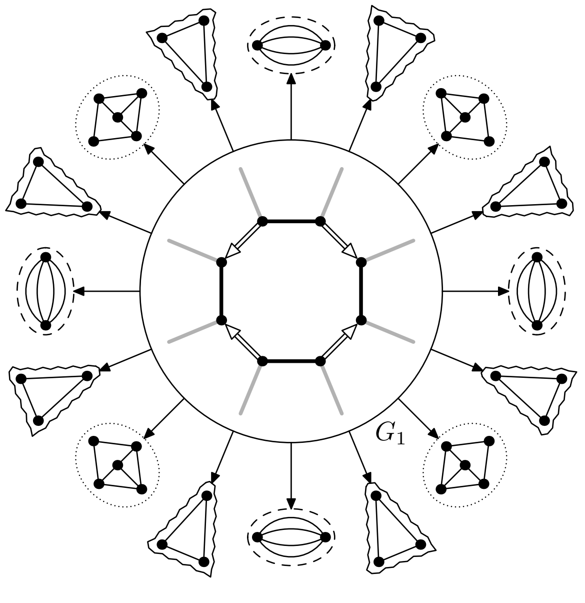


Figure 7.15: The reduction tree for the reduction series in Fig. 7.14. The root is the primitive graph G_1 and each leaf corresponds to one atom of G_0.

disjoint, so the reduction is well defined. We stop in the step r when G_r is a primitive graph containing no atoms. (Recall Lemma 7.4.2 characterizing all primitive graphs.)

Also, in general, the central block/articulation does not have to be preserved and one has to define atoms in all steps of the reduction with respect to the same block/articulation.

Reduction Tree. For every graph G, the reduction series corresponds to the *reduction tree* which is a rooted tree defined as follows. The root is the primitive graph G_r, and the other nodes are the atoms obtained during the reductions. If a node contains a colored edge, it has the corresponding atom as a child. Therefore, the leaves are the atoms of G_0, after removing them, the new leaves are the atoms of G_1, and so on. For an example, see Fig. 7.15.

7.6 Reduction Epimorphism

In this section, we study changes in automorphism groups done by reductions. When G_i is reduced to G_{i+1}, $\mathrm{Aut}(G_{i+1})$ certainly may differ from $\mathrm{Aut}(G_i)$. But the reduction is done right and important information of $\mathrm{Aut}(G_i)$ is preserved in $\mathrm{Aut}(G_{i+1})$. This is formally described by a reduction epimorphism $\mathbf{\Phi}_i : \mathrm{Aut}(G_i) \to \mathrm{Aut}(G_{i+1})$.

Definition of Reduction Epimorphism. We describe algebraic properties of the reductions, in particular how the groups $\mathrm{Aut}(G_i)$ and $\mathrm{Aut}(G_{i+1})$ are related. There exists a natural mapping $\mathbf{\Phi}_i : \mathrm{Aut}(G_i) \to \mathrm{Aut}(G_{i+1})$ called *reduction epimorphism* which we define as follows. Let $\pi \in \mathrm{Aut}(G_i)$. For the common vertices and edges of G_i and G_{i+1}, we define $\mathbf{\Phi}_i(\pi)$ exactly as in π. If A is an atom of G_i, then according to Lemma 7.4.8a, $\pi(A)$ is an atom isomorphic to A. In G_{i+1}, we replace the interiors of both A and $\pi(A)$ by the edges e_A and $e_{\pi(A)}$ of the same type and color. We define $\mathbf{\Phi}_i(\pi)(e_A) = e_{\pi(A)}$. It is easy to see that each $\mathbf{\Phi}_i(\pi) \in \mathrm{Aut}(G_{i+1})$.

For purpose of Section 10.3.1, we also define $\mathbf{\Phi}_i$ on the half edges. Let $e_A = uv$ and let h_u and h_v be the half-edges composing e_A, and similarly let $h_{\pi(u)}$ and $h_{\pi(v)}$ be the half-edges composing $e_{\pi(A)}$. Then we define $\mathbf{\Phi}_i(\pi)(h_u) = h_{\pi(u)}$ and $\mathbf{\Phi}_i(\pi)(h_v) = h_{\pi(v)}$.

7.6.1 Properties of Reduction Epimorphism

First, we prove that the mapping $\mathbf{\Phi}_i$ is indeed an epimorphism:

Proposition 7.6.1. *The mapping $\mathbf{\Phi}_i : \mathrm{Aut}(G_i) \to \mathrm{Aut}(G_{i+1})$ is a group epimorphism, i.e., $\mathbf{\Phi}_i$ is a surjective group homomorphism.*

Proof. First, we argue that $\mathbf{\Phi}_i$ is a group homomorphism. Clearly, $\mathbf{\Phi}_i(\mathrm{id}) = \mathrm{id}$. Let $\pi, \sigma \in \mathrm{Aut}(G_i)$. We need to show that $\mathbf{\Phi}_i(\sigma\pi) = \mathbf{\Phi}_i(\sigma)\mathbf{\Phi}_i(\pi)$. This is clearly true outside the interiors of the atoms. Let A be an atom. By the definition, $\mathbf{\Phi}_i(\sigma\pi)$ maps e_A to $e_{\sigma(\pi(A))}$ while $\mathbf{\Phi}_i(\pi)$ maps e_A to $e_{\pi(A)}$ and $\mathbf{\Phi}(\sigma)$ maps $e_{\pi(A)}$ to $e_{\sigma(\pi(A))}$. So the equality holds everywhere and $\mathbf{\Phi}_i$ is a group homomorphism.

It remains to show that $\mathbf{\Phi}_i$ is surjective. Let $\pi' \in \mathrm{Aut}(G_{i+1})$, we want to extend π' to $\pi \in \mathrm{Aut}(G_i)$ such that $\mathbf{\Phi}_i(\pi) = \pi'$. We just describe this extension on a single edge $e = uv$. If e is an original edge of G, there is nothing to extend. Suppose that e was created in G_{i+1} from an atom A in G_i. Then $\hat{e} = \pi'(e)$ is an edge of the same color and the same type as e, and therefore $\hat{e}$ is constructed from an isomorphic atom $\hat{A}$ of the same symmetry type. The automorphism π' prescribes the action on the boundary ∂A. We need to show that it is possible to define an action on $\mathring{A}$ consistently:

- *A is a block atom:* The edges e and $\hat{e}$ are pendant, attached by articulations u and u'. We define π on $\mathring{A}$ by an isomorphism σ from A to $\hat{A}$ which takes ∂A to $\partial\hat{A}$.
- *A is an asymmetric proper atom/dipole:* By the definition, the orientation of e and $\hat{e}$ is consistent with respect to π'. Since $\mathring{A}$ is isomorphic to the interior of $\hat{A}$, we define π on $\mathring{A}$ according to one such isomorphism σ.
- *A is a symmetric proper atom/dipole:* Let σ be an isomorphism of A and $\hat{A}$. Either σ maps ∂A exactly as π', and then we can use σ for defining π. Or we compose σ with an automorphism of A exchanging the two vertices of ∂A. (We know that such an automorphism exists since A is symmetric.)

So $\mathbf{\Phi}_i$ is a surjective mapping. □

The above statement is an example of a phenomenon known in permutation group theory. Interiors of atoms behave as *blocks of imprimitivity* in the action of $\mathrm{Aut}(G_i)$. It is well-known that the kernel of the action on the imprimitivity blocks is a normal subgroup of $\mathrm{Aut}(G_i)$.

Using Proposition 7.6.1, we can describe change in the automorphism group done by the 3-connected reduction. We are ready to prove Proposition 6.2.1 which states that $\mathrm{Aut}(G_{i+1}) \cong \mathrm{Aut}(G_i)/\mathrm{Ker}(\mathbf{\Phi}_i)$:

Proposition 6.2.1. By Proposition 7.6.1b, $\mathbf{\Phi}_i$ is surjective. By the Homomorphism Theorem, $\mathrm{Aut}(G_{i+1}) \cong \mathrm{Aut}(G_i)/\mathrm{Ker}(\mathbf{\Phi}_i)$. □

Corollary 7.6.2. *We have* $\mathrm{Aut}(G_r) = \mathrm{Aut}(G_0)/\mathrm{Ker}(\mathbf{\Phi}_{r-1} \circ \mathbf{\Phi}_{r-2} \circ \cdots \circ \mathbf{\Phi}_0)$.

Proof. We have already proved that $\mathrm{Aut}(G_{i+1}) = \mathrm{Aut}(G_i)/\mathrm{Ker}(\mathbf{\Phi}_i)$. This equality easily follows from group theory. □

We can also describe the structure of $\mathrm{Ker}(\mathbf{\Phi}_i)$:

Lemma 7.6.3. *The group* $\mathrm{Ker}(\mathbf{\Phi}_i)$ *is the direct product* $\prod_{A \in \mathcal{A}} \mathrm{Fix}(\partial A)$ *where* $\mathrm{Fix}(\partial A)$ *is the point-wise stabilizer of* $G_i \setminus \mathring{A}$ *in* $\mathrm{Aut}(G_i)$.

Proof. According to Lemma 7.4.6, the interiors of the atoms are pairwise disjoint, so $\mathrm{Ker}(\mathbf{\Phi}_i)$ acts independently on each interior. Thus we get $\mathrm{Ker}(\mathbf{\Phi}_i)$ as the direct product of actions on each interior $\mathring{A}$ which is precisely $\mathrm{Fix}(\partial A)$. □

Alternatively, $\mathrm{Fix}(\partial A)$ can be defined as the point-wise stabilizer of ∂A in $\mathrm{Aut}_{\partial A}(A)$. (So $\mathrm{Fix}(\partial A) \le \mathrm{Aut}_{\partial A}(A)$ and it is a proper subgroup if and only if A is symmetric.) Let $A_1, \dots, A_s$ be pairwise non-isomorphic atoms in G_i, appearing with multiplicities $m_1, \dots, m_s$. According to Lemma 7.6.3, we get

$$\mathrm{Ker}(\mathbf{\Phi}_i) \cong \mathrm{Fix}(\partial A_1)^{m_1} \times \cdots \times \mathrm{Fix}(\partial A_s)^{m_s}.$$

For the example of Fig. 7.14, we have $\mathrm{Ker}(\mathbf{\Phi}_0) \cong \mathbb{C}_2^8 \times \mathbb{C}_2^4 \times \mathbb{S}_4^4$.

7.6.2 Semidirect Product

Our aim is to investigate when

$$\mathrm{Aut}(G_i) \cong \mathrm{Ker}(\mathbf{\Phi}_i) \rtimes \mathrm{Aut}(G_{i+1}). \tag{7.1}$$

Let A be an atom with $\partial A = \{u, v\}$. If A is symmetric, there exists some automorphism of A exchanging u and v. If A is a symmetric dipole, one can always find an involution exchanging u and v. This is not true when A is a symmetric proper atom. Figure 7.16a gives an example of a symmetric proper atom with no involution exchanging the two vertices of the boundary. When all symmetric proper atoms have such involutions, we derive (7.1). Figure 7.16b explains that this assumption is necessary.

Proposition 7.6.4. *Suppose that every symmetric proper atom A of G_i with $\partial A = \{u, v\}$ has an involutory automorphism τ exchanging u and v. Then the following holds:*

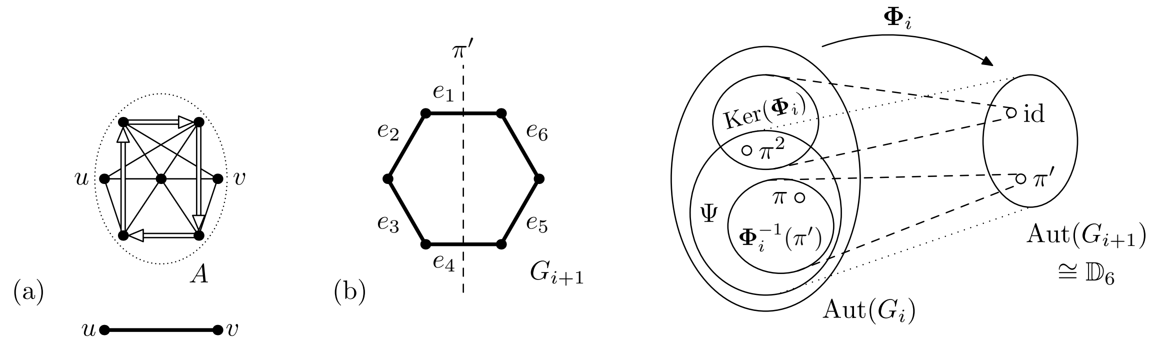


Figure 7.16: (a) An example of a symmetric proper atom A with no involution exchanging u and v. There are two automorphisms which exchange u and v, one rotates the four-cycle formed by white directed edges by one clockwise, the other one counterclockwise. The set-wise stabilizer of $\{u, v\}$ is $\mathbb{C}_4$.
(b) On the left, the graph G_{i+1} having colored edges $e_1, \dots, e_6$ corresponding to copies $A_1, \dots, A_6$ of A. On the right, the groups $\mathrm{Aut}(G_i)$ and $\mathrm{Aut}(G_{i+1}) \cong \mathbb{D}_6$ with the homomorphism $\mathbf{\Phi}_i$. While the rotations in $\mathrm{Aut}(G_{i+1})$ can be easily extended, consider the depicted reflection π'. Let $\pi \in \mathrm{Aut}(G_i)$ such that $\mathbf{\Phi}_i(\pi) = \pi'$. The automorphism $\pi|_{A_1}$ is one of the two automorphisms of A exchanging u and v described in (a), and similarly $\pi|_{A_4}$. Therefore, $\pi^2 \ne \mathrm{id}$ (since $\pi^2|_{A_1} \ne \mathrm{id}$ and $\pi^2|_{A_4} \ne \mathrm{id}$) while $(\pi')^2 = \mathrm{id}$, and only $\pi^4 = \mathrm{id}$. Therefore, no complementary subgroup $\Psi \le \mathrm{Aut}(G_i)$ exists and $\mathrm{Aut}(G_i)$ cannot be constructed using the semidirect product (7.1).

(a) There exists $\Psi \le \mathrm{Aut}(G_i)$ *such that* $\mathbf{\Phi}_i(\Psi) = \mathrm{Aut}(G_{i+1})$ *and* $\mathbf{\Phi}_i|_\Psi$ *is an isomorphism.*

(b) $\mathrm{Aut}(G_i) \cong \mathrm{Ker}(\mathbf{\Phi}_i) \rtimes \mathrm{Aut}(G_{i+1})$.

Proof. (a) In the proof of Proposition 7.6.1, we have described how to extend $\pi' \in \mathrm{Aut}(G_{i+1})$ to $\pi \in \mathrm{Aut}(G_i)$ such that $\mathbf{\Phi}_i(\pi) = \pi'$. To establish (a), we need to do this consistently for entire $\mathrm{Aut}(G_{i+1})$, in such a way that these extensions form a subgroup Ψ which is isomorphic to $\mathrm{Aut}(G_{i+1})$.

Let $e_1, \dots, e_\ell$ be colored edges of one orbit of the action of $\mathrm{Aut}(G_{i+1})$ such that these edges replace isomorphic atoms $A_1, \dots, A_\ell$ in G_i; see Fig. 7.17 for an overview. We divide the argument into three cases:

Case 1: The atom A_1 is a block atom: Let $u_1, \dots, u_\ell$ be the articulations such that $\partial A_i = \{u_i\}$. Choose arbitrarily isomorphisms $\sigma_{1,i}$ from A_1 to A_i such that $\sigma_{1,i}(u_1) = u_i$, and put $\sigma_{1,1} = \mathrm{id}$ and $\sigma_{i,j} = \sigma_{1,j}\sigma_{1,i}^{-1}$. If $\pi'(e_i) = e_j$, we set $\pi|_{\mathring{A}_i} = \sigma_{i,j}|_{\mathring{A}_i}$. Since

$$\sigma_{i,k} = \sigma_{j,k}\sigma_{i,j}, \qquad \forall i,j,k, \tag{7.2}$$

the composition of the extensions π_1 and π_2 of π'_1 and π'_2 is defined on the interiors of $A_1, \dots, A_\ell$ exactly as the extension of $\pi'_2\pi'_1$. Also, by (7.2), an identity $\pi'_k\pi'_{k-1}\cdots\pi'_1 = \mathrm{id}$ is extended to an identity.

Case 2: The atom A_1 is an asymmetric proper atom or dipole: Let $e_i = u_iv_i$. We approach it exactly as in Case 1, just we require that $\sigma_{1,i}(u_1) = u_i$ and $\sigma_{1,i}(v_1) = v_i$.

Case 3: The atom A_1 is a symmetric proper atom or a dipole: For each e_i, we arbitrarily choose one endpoint as u_i and one as v_i. Again, we arbitrarily choose isomorphisms $\sigma_{1,i}$ from A_1 to A_i such that $\sigma_{1,i}(u_1) = u_i$ and $\sigma_{1,i}(v_1) = v_i$, and define $\sigma_{i,j} = \sigma_{1,j}\sigma_{1,i}^{-1}$.

We further consider an involution τ_1 of A_1 which exchanges u_1 and v_1. (Such an involution exists for symmetric proper atoms by the assumptions, and for symmetric dipoles by the definition.) Then τ_1 defines an involution of A_i by conjugation as $\tau_i = \sigma_{1,i}\tau_1\sigma_{1,i}^{-1}$. It follows that

$$\tau_j = \sigma_{i,j}\tau_i\sigma_{i,j}^{-1}, \qquad \text{and consequently} \qquad \sigma_{i,j}\tau_i = \tau_j\sigma_{i,j}, \qquad \forall i,j.$$

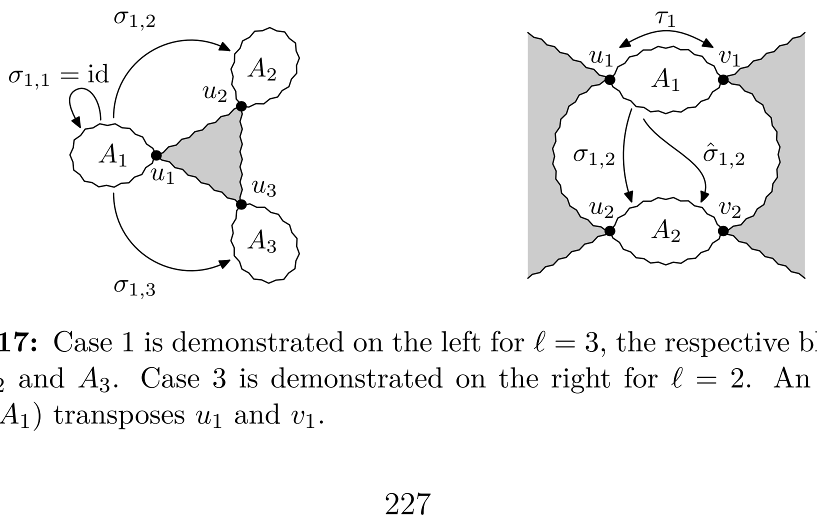


Figure 7.17: Case 1 is demonstrated on the left for $\ell = 3$, the respective block atoms are A_1, A_2 and A_3. Case 3 is demonstrated on the right for $\ell = 2$. An involution $\tau_1 \in \mathrm{Fix}(\partial A_1)$ transposes u_1 and v_1.

We put $\hat{\sigma}_{i,j} = \sigma_{i,j}\tau_i = \tau_j\sigma_{i,j}$ which is an isomorphism mapping A_i to A_j such that $\hat{\sigma}_{i,j}(u_i) = v_j$ and $\hat{\sigma}_{i,j}(v_i) = u_j$. In the extension, we put $\pi|_{\mathring{A}_i} = \sigma_{i,j}|_{\mathring{A}_i}$ if $\pi'(u_i) = u_j$, and $\pi|_{\mathring{A}_i} = \sigma'_{i,j}|_{\mathring{A}_i}$ if $\pi'(u_i) = v_j$.

Aside (7.2), we get the following additional identities:

$$\hat{\sigma}_{i,k} = \sigma_{j,k}\hat{\sigma}_{i,j}, \qquad \hat{\sigma}_{i,k} = \hat{\sigma}_{j,k}\sigma_{i,j}, \quad \text{and} \quad \sigma_{i,k} = \hat{\sigma}_{j,k}\hat{\sigma}_{i,j}, \qquad \forall i,j,k. \tag{7.3}$$

We just argue the last identity:

$$\hat{\sigma}_{j,k}\hat{\sigma}_{i,j} = \tau_k(\sigma_{j,k}\sigma_{i,j})\tau_i = \tau_k\sigma_{i,k}\tau_i = \tau_k\tau_k\sigma_{i,k} = \sigma_{i,k},$$

where the last equality holds since τ_k is an involution. It follows that the composition $\pi_2\pi_1$ is correctly defined as above, and it maps identities to identities.

We have described how to extend the elements of $\mathrm{Aut}(G_{i+1})$ on one edge-orbit, and we apply this process repeatedly to all edge-orbits. The set $\Psi \le \mathrm{Aut}(G_i)$ consists of all these extensions π from every $\pi' \in \mathrm{Aut}(G_{i+1})$. It is a subgroup by (7.2) and (7.3), and since the extension $\pi' \mapsto \pi$ is injective, $\Psi \cong \mathrm{Aut}(G_{i+1})$.

(b) By (a), we know that $\mathrm{Ker}(\mathbf{\Phi}_i) \trianglelefteq \mathrm{Aut}(G_i)$ has a complement subgroup Ψ isomorphic to $\mathrm{Aut}(G_{i+1})$. This already proves that $\mathrm{Aut}(G_i)$ has the structure of the internal semidirect product.

We give more insight into its structure by describing it as an external semidirect product, similarly as in the proof of Theorem 7.2.1. Each element of $\mathrm{Aut}(G_i)$ can be written as a pair (π', σ) where $\pi' \in \mathrm{Aut}(G_{i+1})$ and $\sigma \in \mathrm{Ker}(\mathbf{\Phi}_i)$. We first apply the extension $\pi \in \Psi$ of π' and permute G_i, mapping interiors of the atoms as blocks. Then σ permutes the interiors of the atoms, preserving the remainder of G_i.

It remains to understand how composition of two automorphisms (π', σ) and $(\hat{\pi}', \hat{\sigma})$ works. We get this as a composition of four automorphisms $\hat{\sigma} \circ \hat{\pi} \circ \sigma \circ \pi$, which we want to write as a pair (τ, ρ). Therefore, we need to swap $\hat{\pi}$ with σ. This clearly preserves $\hat{\pi}$, since the action $\hat{\sigma}$ on the interiors does not influence it; so we get $\tau = \hat{\pi} \circ \pi$.

But σ is changed by this swapping. According to Lemma 7.6.3, we get $\sigma = (\sigma_1, \dots, \sigma_s)$ where each $\sigma_i \in \mathrm{Fix}(\partial A_i)^{m_i}$. Since π preserves the isomorphism classes of atoms, it acts on each σ_i independently and permutes the isomorphic copies of A_i. Suppose that A and A' are two isomorphic copies of A_i and $\pi(A) = A'$. Then the action of σ_i on the interior of A corresponds after the swapping to the same action on the interior of $A' = \pi(A)$. This can be described using the semidirect product, since each π defines an automorphism of $\mathrm{Ker}(\mathbf{\Phi}_i)$ which permutes the coordinates of each $\mathrm{Fix}(\partial A_i)^{m_i}$, following the action of π' on the colored edges of G_{i+1}. □

For the example in Fig. 7.14, $\mathrm{Ker}(\mathbf{\Phi}_0) \cong \mathbb{C}_2^4 \times \mathbb{C}_2^4 \times \mathbb{S}_4^4$, so

$$\mathrm{Aut}(G_1) \cong \mathbb{C}_2^2 \qquad \text{and} \qquad \mathrm{Aut}(G_0) \cong (\mathbb{C}_2^4 \times \mathbb{C}_2^4 \times \mathbb{S}_4^4) \rtimes \mathbb{C}_2^2.$$

We note that the semidirect product in

$$\mathrm{Aut}(G_i) \cong \mathrm{Ker}(\mathbf{\Phi}_i) \rtimes \mathrm{Aut}(G_{i+1})$$

cannot be rewritten as the generalized wreath product of Robinson [308] (see Section 7.2). The reason is that we apply an involution τ on a symmetric atoms in G_i when the corresponding edge in G_{i+1} is reflected (i.e., its half-edges are exchanged). So the action of $\mathrm{Aut}(G_i)$ on atoms of G_i is not a simple permutation. We note that Babai [11] uses a different generalization of the wreath product but his characterization is not inductive; see Sections 6.3.2 and 8.4.

7.6.3 Inductive Characterization

Similarly as in the case of connected components (Jordan's Theorem 7.2.1) and block trees (Section 7.3.2), we show that the reduction tree can be used to inductively describe automorphism groups of graphs from the automorphism groups of their 3-connected components. We need to deal with more cases and the products, if they even exist, get more involved. In particular, for Proposition 7.6.4, we need the existence of involutory automorphism exchanging boundaries of symmetric proper atoms, otherwise $\mathrm{Aut}(G)$ may not be expressible by semidirect products. This inductive characterization is applied in Chapter 8 to describe the Jordan-like characterization of the automorphism groups of planar graphs.

Expanded Atoms. Consider the reduction tree of a connected graph G. For an atom A in G_i, let A^* denote the subgraph of G corresponding to the node A and all its descendants in the reduction tree. In other words, A^* is the fully expanded atom A. Let $\mathrm{Fix}(\partial A^*)$ be the point-wise stabilizer of $\partial A^* = \partial A$ in $\mathrm{Aut}_{\partial A^*}(A^*)$.

Fixers of Expanded Atoms. We relate $\mathrm{Fix}(\partial A)$ and $\mathrm{Fix}(\partial A^*)$ similarly as in Proposition 7.6.4. Let $\mathbf{\Phi} : \mathrm{Fix}(\partial A^*) \to \mathrm{Fix}(\partial A)$ be the reduction epimorphism $\mathbf{\Phi} : \pi^* \mapsto \pi$ defined as follows. For $\pi^* \in \mathrm{Fix}(\partial A^*)$, the automorphism $\pi = \mathbf{\Phi}(\pi^*)$ maps the common parts of A and A^* the same while π maps the colored edges in A as π^* maps the expanded atoms in A^*. Note that

$$\mathbf{\Phi} = (\mathbf{\Phi}_0 \circ \mathbf{\Phi}_1 \circ \cdots \circ \mathbf{\Phi}_i)|_{\mathrm{Fix}(\partial A^*)}.$$

Similarly as in the proof of Lemma 7.6.3, we have $\mathrm{Ker}(\mathbf{\Phi}) \cong \prod \mathrm{Fix}(\partial \hat{A}^*)$. Assuming that all symmetric (expanded) atoms have involution exchanging boundaries, exactly as in Proposition 7.6.4b, we can prove

$$\mathrm{Fix}(\partial A^*) \cong \mathrm{Ker}(\mathbf{\Phi}) \rtimes \mathrm{Fix}(\partial A). \tag{7.4}$$

Similarly as in the proof of Jordan's Theorem 7.2.1, we describe the semidirect product in (7.4) in more detail. The group $\mathrm{Ker}(\mathbf{\Phi})$ consists of all automorphisms which fix A and only act non-trivially on interiors of all expanded atoms $\hat{A}^*$. Each automorphism $\pi^* \in \mathrm{Fix}(A^*)$ can be written as a composition $\sigma \cdot \pi'$ of two automorphisms. First, the automorphism $\sigma \in \mathrm{Ker}(\mathbf{\Phi})$ acts on interiors of all $\hat{A}^*$. Then, the automorphism π' acts on A^* as $\pi \in \mathrm{Fix}(A)$ acts on A, and π' maps interiors of $\hat{A}^*$ exactly as π maps the corresponding colored edges. When we compose $\sigma_1 \cdot \pi'_1 \cdot \sigma_2 \cdot \pi'_2$, we want to swap π'_1 with σ_2 to write the resulting automorphism in the form $\sigma \cdot \pi'$, which is done by the semidirect product. In the proof of Proposition 7.6.4a, we explain how to define the correspondence $\pi \mapsto \pi'$ consistently.

Inductive Characterization. Let G be a connected graph. We want to describe $\mathrm{Aut}(G)$ from the reduction tree T and the automorphism groups of the atoms and of the primitive graph is the 3-connected reduction of G. As in Section 7.3.2, we construct the groups from the bottom to the root. The equation (7.4) describes how to construct $\mathrm{Fix}(\partial A^*)$ when we know $\mathrm{Fix}(\partial \hat{A}^*)$ for all atoms $\hat{A}$ represented by colored edges in A. To describe this product in more detail, we may distinguish different types of atoms.

For dipoles and star block atoms, $\mathrm{Fix}(\partial A^*)$ may be constructed by direct products and wreath products with symmetric groups from $\mathrm{Fix}(\partial \hat{A}^*)$ of all atoms $\hat{A}$; see Lemma 8.2.3 for details. For non-star block atoms and proper atoms, the situation is more complex. If nothing in known about G, the semidirect product is determined by the edge-orbits in A. (Which are fixed, which are reflected; see 8.2.)

The last step is with in the root, with the primitive graph G_r. Again, we have the reduction epimorphism $\mathbf{\Phi} : \mathrm{Aut}(G) \to \mathrm{Aut}(G_r)$ where $\mathbf{\Phi} = \mathbf{\Phi}_{r-1} \circ \mathbf{\Phi}_{r-2} \circ \cdots \circ \mathbf{\Phi}_0$. Since $\mathrm{Aut}(G)$ does not have to stabilize anything in G_r, we have the semidirect product with $\mathrm{Aut}(G_r)$ instead.

$$\mathrm{Aut}(G) \cong \mathrm{Ker}(\mathbf{\Phi}) \rtimes \mathrm{Aut}(G_r). \tag{7.5}$$

Even in the case of planar graphs, it is involved to describe all possible combinations of edge-orbits in $\mathrm{Aut}(G_r)$ (see Tables 8.1, 8.2, and 8.3 of [218]).

7.7 Polynomial-time Algorithms

Last, we prove that the reduction series and the corresponding reduction tree can be computed in polynomial time for a graph G belonging to a graph class $\mathcal{C}$ satisfying (P1) and (P3*).

Simplified Reduction and Reduction Tree. The reduction series is called *simplified* if it is done without colored and oriented edges. Similarly, the *simplified reduction tree* contains no colored and oriented edges. The results from Section 7.6 no longer hold, but the simplified reduction may be applied for other reasons. For instance, it capture all geometric representations of planar graphs, so it can be applied for graph drawing; it corresponds to the SPQR trees [95, 96, 97, 167].

Lemma 7.7.1. *The simplified reduction series and the simplified reduction tree can be computed in time $\mathcal{O}(n+m)$.*

Proof. We just describe how to compute the simplified reduction tree. First, we divide the graph G into blocks and articulations in time $\mathcal{O}(n+m)$ using DFS [190]. Then, we run the linear-time algorithm of [191, 167] to compute 3-connected components of each block. The reduction tree can be easily constructed. □

In the rest of the section, we deal with the complexity of computing the non-simplified reduction series and reduction tree.

Testing Graph Isomorphism for Primitive Graphs and Atoms.

Lemma 7.7.2. *If primitive graphs G and G' belong to $\mathcal{C}$ satisfying (P3*), then we can test $G \cong G'$ in polynomial time.*

Proof. In both graphs, we replace single pendant edges with colored vertices. If G and G' are K_1, or K_2, the problem is trivial. If they are cycles, we use the standard cycle isomorphism algorithms. If they are 3-connected, we test $G \cong G'$ using (P3*). $\square$

Lemma 7.7.3. *For every atoms A and A' of a graph G belonging to $\mathcal{C}$ satisfying (P1) and (P3*), we can test $A \cong A'$ in polynomial time.*

Proof. If both A and A' are dipoles and star block atoms, we can test $A \cong A'$ trivially in polynomial time. If they are non-star block atoms, by Lemma 7.4.3 they are either K_2 with attached single pendant edge, or essentially a cycle, or essentially 3-connected. The first two possibilities can be solved trivially, so we assume that A and A' are essentially 3-connected. Let B and B' be the 3-connected graph created from A and A' by removing pendant edges, where existence of pendant edges is coded by colors of $\boldsymbol{V}(B)$ and $\boldsymbol{V}(B')$, and we further color ∂B and $\partial B'$ by a special color. We have $A \cong A'$ if and only if there exists a color-preserving isomorphism between B and B' which can be tested using (P3*). When A and A' are proper atoms, we proceed similarly on extended proper atoms, using Lemma 7.4.4. $\square$

Computing Symmetry Types of Atoms.

Lemma 7.7.4. *For a dipole A, we can determine its symmetry type in linear time.*

Proof. The symmetry type depends only on the quantity of distinguished types of the parallel edges. We have directed edges from u to v, directed edges from v to u, undirected edges and halvable edges. We call a dipole *balanced* if the number of directed edges in the both directions is the same. The dipole is symmetric if and only if it is balanced. This clearly can be tested in linear time. $\square$

Lemma 7.7.5. *For a proper atom A of $\mathcal{C}$ satisfying (P1), (P2), and (P3*), we can determine its symmetry type in polynomial time.*

Proof. Let $\partial A = \{u, v\}$. By Lemma 7.4.4, A^+ is either essentially a cycle (which is easy to deal with), or an essentially 3-connected graph. Let B be the 3-connected graph created from A^+ by removing pendant edges, where existence of pendant edges is coded by colors of $\boldsymbol{V}(B)$. By (P1), both A^+ and B belong to $\mathcal{C}$. We apply (P3*) on

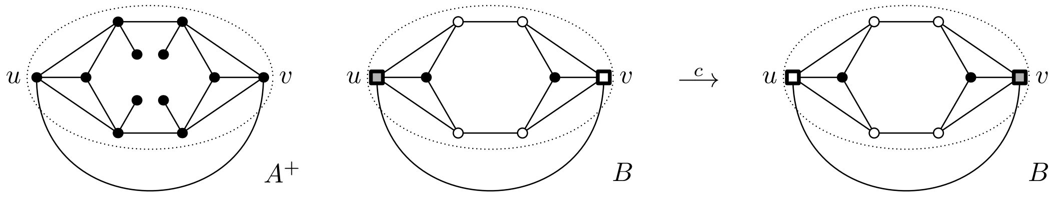


Figure 7.18: For the depicted atom A, we test using (P3*) whether $B \xrightarrow{c} B$. In this case yes, so A is either symmetric, or halvable.

two copies of B. In one copy, we color u by a special color, and v by another special color. In the other copy, we swap the colors of u and v. Using (P3*), we check whether there exists a color-preserving automorphism which exchanges u and v; see Fig. 7.18. The atom A is symmetric if and only if such an automorphism exists. $\square$

Computing Reduction Series and Reduction Tree.

Lemma 7.7.6. *If a graph G belongs to $\mathcal{C}$ satisfying (P1) and (P3*), then the reductions series $G = G_0, \dots, G_r$ and the reduction tree can be computed in polynomial time.*

Proof. To compute G_{i+1} from G_i, we find all atoms $\mathcal{A}$ in G_i and $\mathcal{A}'$ in G'_i, compute their isomorphism classes by Lemma 7.7.3 and assign new colors to them. By Lemmas 7.7.4 and 7.7.5, we compute symmetry types of these atoms. We end up with a primitive graph G_r containing the atoms. The reduction tree can be easily constructed and the algorithm runs in polynomial time.

Alternatively, we apply Lemma 7.7.1 and just compute the missing colors and symmetry types of atoms from the leaves to the root of the simplified reduction tree, using Lemmas 7.7.3, 7.7.4, and 7.7.5. $\square$

Deciding Graph Isomorphism. The following result is usually credited to Babai [11]:

Lemma 7.7.7 ([11])**.** *If graphs G and G' belong to $\mathcal{C}$ satisfying (P1) and (P3*), we can test $G \cong G'$ in polynomial time.*

Proof. Using Lemma 7.7.6, we simultaneously apply reduction series on both G and G' in polynomial time, using identical colors for isomorphic atoms in G_i and G'_i. We end up with two primitive graphs G_r and G'_r and we test their isomorphism using Lemma 7.7.2. $\square$

7.8 Comparison with Previous Results

We conclude this chapter by comparison of the described 3-connected reduction and reduction trees with previously known results. We have postponed this discussion till the end of this chapter, so we have the necessary definitions.

Rooted versus Unrooted Decompositions. Many decompositions in graph theory and computer science are unrooted. At first, choosing a root seems arbitrary. For instance, the standard definition of block trees is unrooted. Similarly, SPQR trees are usually unrooted (for instance, see their description in Wikipedia). In this chapter, we have explained that having a root is necessary to work with automorphism groups and related algebraic properties.

Further, this is not imposing a root into an unrooted structure. Algebra reveals that these decompositions should be rooted. Decompositions which repeatedly divide graph into two pieces (such as MacLane's [265] description of 3-connected reduction or the split decomposition described in Section 2.6) have these pieces on equal setting.

The advantage of our definition of the reduction series that naturally distinguishes them. Atoms are clearly completely different objects than the reduced graph G_{i+1} created from G_i by replacing atoms with edges. The difference is that the structure of the primitive graph, being the root node of the reduction tree, is preserved in G_{i+1} but it is not contained in replaced atoms. This distinction is clear when working with automorphism groups. (A similar phenomenon occurs for the rooted split tree created by the minimal split decomposition of [79] described in Section 2.6, and we believe it should be further investigated.)

SPQR Trees. The 3-connected decomposition was rediscovered in the graph drawing community under the name SPQR trees [95, 96, 97, 167] while ignoring many of previous results. The papers mostly cite the algorithm of Hopcroft and Tarjan [191] and the paper of MacLane [265]. We believe that other papers, e.g. [340, 344, 78, 352], deserve to be more widely known.

SPQR trees are mostly applied to planar graphs but they also work for general graphs. We quickly sketch the main ideas of [95]. Ignoring Q, they are reduction trees with three types of nodes:

- *Series nodes S* correspond to paths.
- *Parallel nodes P* correspond to dipoles.
- *Rigid nodes R* correspond to 3-connected graphs.

In [95], SPQR trees are defined in the context of s, t-graphs in which the graph G remains planar when the edge st is added into the drawing. Such graphs correspond to proper or dipole parts in our definitions, and the added edge creates A^+. (Lemma 7.4.4 states that proper atoms are S and R.) The decomposition is done in [95] in the opposite order than our decomposition, by finding maximal disjoint pieces and decomposing each recursively.

The problem is that different papers often define SPQR trees differently. For instance, it is unclear what are precisely the nodes. Some papers allow arbitrary cycles, other reduce it non-canonically to several nodes isomorphic to K_3. In [167], some nodes are collapsed back together, in relation to the results of MacLane [265]. SPQR trees are often unrooted, while other papers add an arbitary root in the beginning (to start with an s, t graph). These descriptions are not precise enough to capture all fine details of automorphism groups and regular graph covers.[3]

On the other hand, SPQR trees certainly deserve a credit. The graph drawing community greatly popularized this idea. It is used in many papers studying restricted planar embeddings and other problems. Also, an error in the original linear-time algorithm of Hopcroft and Tarjan [191] was discovered and fixed in [167]. An efficient tested implementation of SPQR trees is programmed in [167] and it is available in the

[3]While working on [118], we had to rework the definition of atoms and of the reduction for several times just because some details were later broken. For instance, originally, we wanted to restrict ourselves to 2-connected graphs, as well, but it did not capture regular graph covers. Unfortuntaly, this process of creating a mathematical definition cannot be described in this thesis (or most mathematical papers).

graph drawing library AGD. As usual, many great ideas in mathematics and computer science are rediscovered.

Connected versus 2-connected Graphs. To the best of our knowledge, our reduction tree is the only tree for 3-connected reduction which applies on connected graphs instead of 2-connected graphs. In Sections 6.2.3, 7.3.3, and 10.1, we comment advantages of this unified approach. What are the disadvantages? The definition of atoms in Section 7.4 is involved. But after working out the basic results in Sections 7.4.2 and 7.4.3, atoms become quite easy to use. Our decomposition has five different objects: proper atoms, dipoles, star and non-star block atoms, and primitive graphs. But they serve a different purpose, e.g., behave differently with respect to automorphism groups; see Chapter 8. So it is natural to analyze these five cases separately. In comparison, there is not much difference between S and R nodes of the SPQR trees.

4-connected Reduction. We conclude with an open problem. We have described the way how to reduce a graph to a 3-connected one while preserving its essential structural information. This approach is highly efficient for planar graphs since many problems are much simpler for 3-connected planar graphs; for instance planar embeddings, automorphism groups or regular graph covers. Suppose that we would like to push our results further, say to toroidal or projective planar graphs. The issue is that 3-connectivity does not restrict them much. Is it possible to apply some “4-connected reduction”, to reduce the input graphs even further? Suppose that one would generalize proper atoms to be inclusion minimal parts of the graph separated by a 3-cut. Would it be possible to replace them by triangles?

8 Jordan-like Characterization of Automorphism Groups of Planar Graphs

This chapter contains:

- *8.1: Automorphism Groups of 3-connected Planar Graphs.* We describe spherical groups and the geometry of automorphism groups of planar graphs. We also characterize automorphism groups of planar atoms and primitive graphs.
- *8.2: The Jordan-like Characterization.* We give the first inductive characterization of the automorphism groups of planar graphs. First, we characterize the stabilizers as the class of groups closed under 5 group products. Then, we combine the stabilizers with spherical groups.
- *8.3: Applications of Jordan-like Characterization.* We describe automorphism groups of 2-connected planar graphs, outerplanar graphs, and series-parallel graphs.
- *8.4: Comparison with Babai's Characterization.* We compare the Jordan-like inductive characterization with Babai's characterization [11] from 1975.
- *8.5: Quadratic-time Algorithm.* Our characterization implies a quadratic-time algorithm for computing automorphism groups of planar graphs.
- *8.6: Conclusions.* We describe open problems and a spacial visualization of the automorphism groups of planar graphs.

`http://pavel.klavik.cz/orgpad/geom_aut_groups.html`

8.1 Automorphism Groups of 3-connected Planar Graphs

In this section, we review geometric properties of automorphism groups of 3-connected planar graphs. They are based on Whitney's Theorem [359] stating that 3-connected planar graphs have unique embeddings onto the sphere. Using these properties, we describe possible automorphism groups of planar atoms and primitive graphs.

Spherical Groups. A group is *spherical* if it is the group of the isometries of a tiling of the sphere. The first class of spherical groups are the subgroups of the automorphism groups of the platonic solids. Their automorphism groups are $\mathbb{S}_4$ for the tetrahedron, $\mathbb{S}_4 \times \mathbb{C}_2$ for the cube and the octahedron, and $\mathbb{A}_5 \times \mathbb{C}_2$ for the dodecahedron and the icosahedron; see Fig. 1.27. The second class of spherical groups is formed by four infinite families, namely $\mathbb{C}_n$, $\mathbb{D}_n$, $\mathbb{C}_n \times \mathbb{C}_2$, and $\mathbb{D}_n \times \mathbb{C}_2$, $n \geq 2$. They act as groups of automorphism of n-sided prisms.

Maps. A (spherical) map $\mathcal{M}$ is a 2-cell decomposition of the sphere S. A map is usually defined by a 2-cell embedding of a connected graph $i\colon G \hookrightarrow S$. The connected components of $S \setminus i(G)$ are called faces of $\mathcal{M}$. So it corresponds to spherical embeddings, defined in Section 1.4. An automorphism of a map is an automorphism of the graph preserving the incidences between vertices, edges, and faces. Clearly, $\mathrm{Aut}(\mathcal{M})$ is one of the spherical groups and with the exception of paths and cycles, it is a subgroup of $\mathrm{Aut}(G)$. As a consequence of Whitney's theorem [359] we have the following.

Theorem 8.1.1. *Let $\mathcal{M}$ be the map given by the unique 2-cell embedding of a 3-connected graph into the sphere. Then* $\mathrm{Aut}(G) \cong \mathrm{Aut}(M)$.

Geometry of Automorphisms. It is not possible to capture $\mathrm{Aut}(G)$ of a planar graph G as isometries of a planar embedding, even when G is 3-connected, since isometries of a planar embedding form a subgroup of a dihedral group. But recall from Section 1.4.1 that Mani [268] proved that for every 3-connected planar graph G, there exists a polyhedron P such that $\mathrm{Aut}(G)$ coincides with the group of isometries of P. Since P can be drawn symmetrically onto the sphere, each isometry of P correspond to some isometry of the sphere, so automorphisms in $\mathrm{Aut}(G)$ can be geometrically viewed as isometries of the sphere with G drawn onto it, and this is essential for the Jordan-like characterization in for Section 8.2. In particular, we have:

Theorem 8.1.2. *Let G be a 3-connected graph. Then* $\mathrm{Aut}(G)$ *is isomorphic to one of the spherical groups.*

We recall some basic definitions from geometry [331, 287]. An automorphism of a 3-connected planar graph G is called *orientation preserving*, if the respective isometry preserves the global orientation of the sphere. It is called *orientation reversing* if it changes the global orientation of the sphere. A subgroup of $\mathrm{Aut}(G)$ is called *orientation preserving* if all its automorphisms are orientation preserving, and *orientation reversing* otherwise. We note that every orientation reversing subgroup contains an orientation preserving subgroup of index two. (The reason is that the composition of two orientation reversing automorphisms is an orientation preserving automorphism.)

Stabilizers. Let $u \in \boldsymbol{V}(G)$. The stabilizer of u in $\mathrm{Aut}(G)$ is a subgroup of a dihedral group and it has the following description in the language of isometries. If $\mathrm{Stab}(u) \cong \mathbb{C}_n$, for $n \geq 3$, it is generated by a *rotation* of order n that fixes u and the opposite point of the sphere, and fixing no other point of the sphere. The opposite point of the sphere may be another vertex or a center of a face. If $\mathrm{Stab}(u) \cong \mathbb{D}_n$, it consists of rotations and *reflections* fixing a circle passing through u and the opposite point of the sphere. Each reflection always fixes either a center of some edge, or another vertex. When $\mathrm{Stab}(u) \cong \mathbb{D}_1 \cong \mathbb{C}_2$, it is generated either by a 180° rotation or by a reflection.

Let $e \in \boldsymbol{E}(G)$. The stabilizer of e in $\mathrm{Aut}(G)$ is a subgroup of $\mathbb{C}_2^2$. When $\mathrm{Stab}(e) \cong \mathbb{C}_2^2$, it contains the following three non-trivial isometries. First, the 180° rotation around the center of e and the opposite point of the sphere that is a vertex, center of an edge, or center of an even face. Next, two reflections orthogonal to each other which fix circles through u and the opposite point of the sphere. When $\mathrm{Aut}(G) \cong \mathbb{C}_2$, it is generated by only one of these three isometries.

8.1.1 Automorphism Groups of Planar Primitive Graphs and Atoms

Theorem 8.1.2 allows us to describe possible automorphism groups of planar atoms and primitive graphs which appear in the reduction tree for a planar graph G. First, we describe the automorphism groups of planar primitive graphs.

Lemma 8.1.3. *The automorphism group* $\mathrm{Aut}(G)$ *of a planar primitive graph G is a spherical group.*

Proof. Recall that a graph is essentially 3-connected if it is a 3-connected graph with attached single pendant edges to some of its vertices. If G is essentially 3-connected, then $\mathrm{Aut}(G)$ is a spherical group from Theorem 8.1.2. Since the family of spherical groups is closed under taking subgroups, the subgroup of color- and orientation-preserving automorphism is spherical as well. If G is K_1, K_2 or C_n with attached single pendant edges, then $\mathrm{Aut}(G)$ is a subgroup of $\mathbb{C}_2$ or $\mathbb{D}_n$. □

Next, we deal with the automorphism groups of planar atoms. Let A be a planar atom. Recall that $\mathrm{Aut}_{\partial A}(A)$ is the set-wise stabilizer of ∂A, and $\mathrm{Fix}(\partial A)$ is the point-wise stabilizer of ∂A. The following lemma determines $\mathrm{Aut}_{\partial A}(A)$; see Fig. 8.1 for examples.

Lemma 8.1.4 ([119], Lemma 5.3)**.** *Let A be a planar atom.*

- *(a) If A is a star block atom, then* $\mathrm{Aut}_{\partial A}(A) = \mathrm{Fix}(\partial A)$ *which is a direct product of symmetric groups.*
- *(b) If A is a non-star block atom, then* $\mathrm{Aut}_{\partial A}(A) = \mathrm{Fix}(\partial A)$ *and it is a subgroup of a dihedral group.*
- *(c) If A is a proper atom, then* $\mathrm{Aut}_{\partial A}(A)$ *is a subgroup of* $\mathbb{C}_2^2$ *and* $\mathrm{Fix}(\partial A)$ *is a subgroup of* $\mathbb{C}_2$*.*
- *(d) If A is a dipole, then* $\mathrm{Fix}(\partial A)$ *is a direct product of symmetric groups. If A is symmetric, then* $\mathrm{Aut}_{\partial A}(A) = \mathrm{Fix}(\partial A) \rtimes \mathbb{C}_2$*. If A is asymmetric, then* $\mathrm{Aut}_{\partial A}(A) = \mathrm{Fix}(\partial A)$*.*

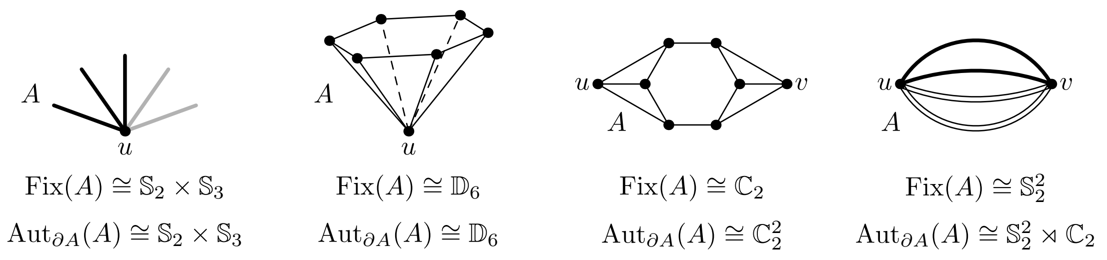

A
u
$\mathrm{Fix}(A) \cong \mathbb{S}_2 \times \mathbb{S}_3$
$\mathrm{Aut}_{\partial A}(A) \cong \mathbb{S}_2 \times \mathbb{S}_3$
A
u
$\mathrm{Fix}(A) \cong \mathbb{D}_6$
$\mathrm{Aut}_{\partial A}(A) \cong \mathbb{D}_6$
u
v
A
$\mathrm{Fix}(A) \cong \mathbb{C}_2$
$\mathrm{Aut}_{\partial A}(A) \cong \mathbb{C}_2^2$
u
v
A
$\mathrm{Fix}(A) \cong \mathbb{S}_2^2$
$\mathrm{Aut}_{\partial A}(A) \cong \mathbb{S}_2^2 \rtimes \mathbb{C}_2$

Figure 8.1: An atom A together with its groups $\mathrm{Fix}(\partial A)$ and $\mathrm{Aut}_{\partial A}(A)$. From left to right, a star block atom, a non-star block atom, a proper atom, and a dipole.

Proof. (a) The edges of each color class of the star block atom A can be arbitrarily permuted, so $\mathrm{Aut}_{\partial A}(A) = \mathrm{Fix}(\partial A)$ which is a direct product of symmetric groups.

(b) For the non-star block atom A, the boundary $\partial A = \{u\}$ is stabilized. We have one vertex in both $\mathrm{Aut}_{\partial A}(A)$ and $\mathrm{Fix}(\partial A)$ fixed, thus the groups are the same. By Lemma 7.4.3, we have that A is either a essentially a cycle, K_2 with attached single pendant edges, or essentially 3-connected, so $\mathrm{Aut}_{\partial A}(A)$ is a subgroup of $\mathbb{D}_n$ where n is the degree of u.

(c) Let A be a proper atom with $\partial A = \{u, v\}$. By Lemma 7.4.4, A^+ is either essentially a cycle, or essentially 3-connected. The first case is trivial, so we deal with the latter case. Since $\mathrm{Aut}_{\partial A}(A)$ preserves ∂A, we have $\mathrm{Aut}_{\partial A}(A) = \mathrm{Aut}_{\partial A^+}(A^+)$, and $\mathrm{Aut}_{\partial A^+}(A^+)$ fixes in addition the edge uv. Because A^+ is essentially 3-connected, $\mathrm{Aut}_{\partial A^+}(A^+)$ corresponds to the stabilizer of uv in $\mathrm{Aut}(\mathcal{M})$ for a map $\mathcal{M}$ of A^+. But such a stabilizer is a subgroup of $\mathbb{C}_2^2$. Since $\mathrm{Fix}(\partial A)$ stabilizes the vertices of ∂A, it is a subgroup of $\mathbb{C}_2$.

(d) For an asymmetric dipole, we have $\mathrm{Aut}_{\partial A}(A) = \mathrm{Fix}(\partial A)$ which is a direct product of symmetric groups. For a symmetric dipole, we can permute the vertices in ∂A, so we get the semidirect product with $\mathbb{C}_2$. □

Last, we argue that every planar symmetric atom A has an involutory automorphism exchanging ∂A, so the assumptions of Proposition 7.6.4 are always satisfied for planar graphs.

Lemma 8.1.5. *For every planar symmetric proper atom A with $\partial A = \{u, v\}$, there exists an involutory automorphism exchanging u and v.*

Proof. Since $\mathrm{Aut}_{\partial A}(A)$ is a subgroup of $\mathbb{C}_2^2$, all elements are involutions. □

8.2 The Jordan-like Characterization

The automorphism groups of planar graphs are constructed using Theorem 7.2.1 from the automorphism groups of its connected components. It remains to deal with the automorphism groups of connected planar graphs. We describe them in this section, using the results of Chapter 7 and Section 8.1, thus proving the main result of this chapter. We show that Aut(connected PLANAR) can be described by a semidirect product series composed from few basic groups. In Subsection 8.2.1, we prove Theorems 6.3.7

and 6.3.8 as an easy consequence of our previous results. In Subsection 8.2.2, we give a Jordan-like characterization of all point-wise stabilizers of a vertex, or of a pair of vertices, in the automorphism groups of connected planar graphs. In Subsection 8.2.3, we describe all possible compositions of actions of spherical groups with the stabilizers described in Subsection 8.2.2.

8.2.1 Characterization by Semidirect Product Series

First, we prove Theorems 6.3.7 and 6.3.8 which can be viewed as a rough approximation of the main result.

Proof of Theorem 6.3.7. We define an epimorphism $\Theta_i\colon \mathrm{Aut}(G) \to \mathrm{Aut}(G_i)$ by $\Theta_i = \mathbf{\Phi}_0 \circ \cdots \circ \mathbf{\Phi}_{i-1}$, for $i = 1, \dots, r-1$. We have $\mathrm{Ker}(\Theta_{r-i}) > \mathrm{Ker}(\Theta_{r-i-1})$ and since all the $\mathrm{Ker}(\Theta_{r-i})$ are normal in $\mathrm{Ker}(\Theta_r)$, we can write $\mathrm{Ker}(\Theta_{r-i}) \rhd \mathrm{Ker}(\Theta_{r-i-1})$. By definition, $\Theta_{r-i} = \Theta_{r-i-1} \circ \mathbf{\Phi}_{r-i}$. Therefore, $\mathrm{Ker}(\Theta_{r-i})/\mathrm{Ker}(\Theta_{r-i-1}) \cong \mathrm{Ker}(\mathbf{\Phi}_{r-i})$, for $i = 1, \dots, r-1$. By Lemmas 7.4.2 and 8.1.4, $\mathrm{Ker}(\mathbf{\Phi}_{r-i})$ is isomorphic to a direct product of symmetric, cyclic, and dihedral groups. Moreover, $\mathrm{Aut}(G)/\mathrm{Ker}(\Theta_{r-1}) \cong \mathrm{Aut}(G_r)$. By Lemma 8.1.3 $\mathrm{Aut}(G_r)$ is isomorphic to a spherical group. We have a subnormal chain $\mathrm{Aut}(G) = \Psi_0 \rhd \Psi_1 \rhd \cdots \rhd \Psi_{r-1} = 1$, where $\Psi_i = \mathrm{Ker}(\Theta_{r-i})$ such that Ψ_i/Ψ_{i+1} is a product of the required groups. By Jordan-Hölder theorem there exists a refinement satisfying the statement of the above subnormal chain. □

Proof of Theorem 6.3.8. The primitive graph G_r has $\mathrm{Aut}(G_r)$ isomorphic to a spherical group by Lemma 8.1.3. By Lemma 8.1.5, we can apply Proposition 7.6.4 and $\mathrm{Aut}(G_i) \cong \mathrm{Ker}(\mathbf{\Phi}_i) \rtimes \mathrm{Aut}(G_{i+1})$. By Lemma 7.6.3, the kernel $\mathrm{Ker}(\mathbf{\Phi}_i)$ is the direct product of the groups $\mathrm{Fix}(\partial A)$ for all atoms A in G_i. Each of these groups is isomorphic to either to a cyclic, or to a dihedral group, or to a direct product of symmetric groups. □

Theorems 6.3.7 and 6.3.8 impose some necessary conditions fulfilled by the automorphism groups of a planar graph. On the other hand, not every abstract group satisfying these conditions is isomorphic to the automorphism group of some planar graph. First, $\mathrm{Aut}(G_{i+1})$ admits an induced action on the groups $\mathrm{Fix}(\partial A)$, where A ranges through all atoms of G_i. In particular, the sizes of orbits of $\mathrm{Aut}(G_{i+1})$ are reflected in $\mathrm{Aut}(G_i)$, since an orbit of length m gives rise to m copies of $\mathrm{Fix}(\partial A)$, for some atom A. For instance, if $\mathrm{Aut}(G_r) \cong \mathbb{C}_n$, then every orbit is of size 1, or n. Therefore, the possible powers of $\mathrm{Fix}(\partial A)$ in $\mathrm{Ker}(\mathbf{\Phi}_{r-1})$ are restricted. Applying Theorem 6.3.8 repeatedly, we can construct $\mathrm{Aut}(G)$ recursively, starting in the root of the reduction tree and terminating its leaves.

8.2.2 Fixer of the Boundary of an Expanded Atom

In the remainder of this section, we use the approach of Section 7.6.3.

$$\mathrm{Fix}(\textsf{connected PLANAR}) = \Big\{\mathrm{Fix}(\partial A^*) : A \text{ is an atom} \\ \text{of the reduction tree of a planar graph}\Big\}.$$

Equivalently, Fix(connected PLANAR) consists of all point-wise stabilizers of a vertex, or of a pair of vertices, in the automorphism groups of connected planar graphs.

Theorem 8.2.1. *The class* Fix(connected PLANAR) *is defined inductively as follows:*

(a) $\{1\} \in$ Fix(connected PLANAR).
(b) *If* $\Psi_1, \Psi_2 \in$ Fix(connected PLANAR), *then* $\Psi_1 \times \Psi_2 \in$ Fix(connected PLANAR).
(c) *If* $\Psi \in$ Fix(connected PLANAR), *then* $\Psi \wr \mathbb{S}_n, \Psi \wr \mathbb{C}_n \in$ Fix(connected PLANAR).
(d) *If* $\Psi_1, \Psi_2, \Psi_3 \in$ Fix(connected PLANAR), *then*

$$(\Psi_1^{2n} \times \Psi_2^n \times \Psi_3^n) \rtimes \mathbb{D}_n \in \text{Fix}(\textsf{connected PLANAR}), \quad \forall n \textit{ odd.}$$

(e) *If* $\Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5 \in$ Fix(connected PLANAR), *then*

$$(\Psi_1^{2n} \times \Psi_2^n \times \Psi_3^n \times \Psi_4^n \times \Psi_5^n) \rtimes \mathbb{D}_n \in \text{Fix}(\textsf{connected PLANAR}), \quad \forall n \geq 4, \textit{ even.}$$

(f) *If* $\Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5, \Psi_6 \in$ Fix(connected PLANAR), *then*

$$(\Psi_1^4 \times \Psi_2^2 \times \Psi_3^2 \times \Psi_4^2 \times \Psi_5^2 \times \Psi_6) \rtimes \mathbb{C}_2^2 \in \text{Fix}(\textsf{connected PLANAR}).$$

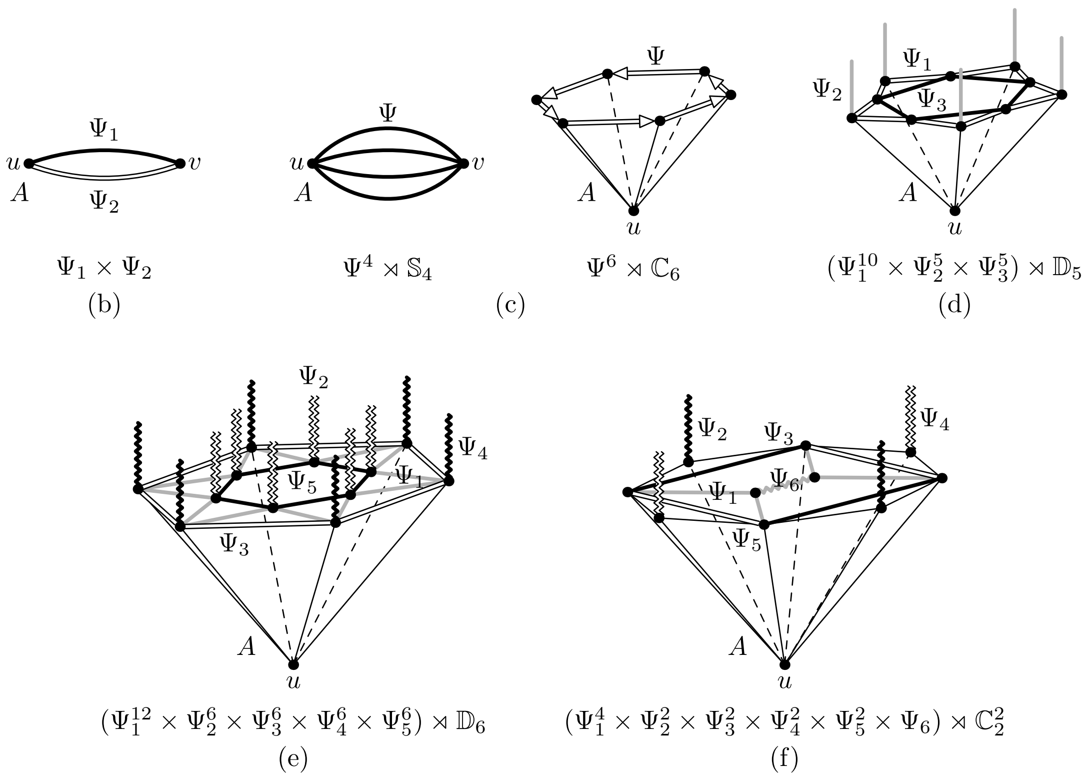
We prove this theorem in a series of lemmas below. We note that the homomorphisms defining the semidirect products in the operations (d), (e), and (f) are specified in the proofs of Lemmas 8.2.4, 8.2.7, 8.2.8, and 8.2.9.

Lemma 8.2.2. *The class* Fix(connected PLANAR) *is closed under operations (a)–(f). Further, every such group can be realized by a block atom, by a proper atom, or by a dipole, in arbitrarily many non-isomorphic ways.*

Proof. It is clear for (a) and Fig. 8.2 shows the constructions for the operations (b)–(f). Concerning the second part, for every group Ψ, arbitrarily many non-isomorphic atoms A such that $\Psi \cong \text{Fix}(\partial A)$ can be constructed. For instance, we can do it by replacing edges of a realization of Ψ by suitable rigid planar graphs (having no non-trivial automorphisms) consistently with the action of Ψ. Similarly, if some group can be realized by, say block atom, we can attach the corresponding pendant edge to some, say, rigid proper atom which preserves the group. □

In the rest of this section, we prove that each group in Fix(connected PLANAR) arises by using operations (b)–(f) repeatedly. Similarly as in Section 7.6.3, we prove this by induction according to the depth of the reduction tree. Let A be an atom in G_{i+1}, with each colored edge corresponding to some atom $\hat{A}$ in G_i which is expanded to $\hat{A}^*$. The expanded atom A^* is constructed from A by replacing all colored edges with expanded atoms $\hat{A}^*$. By induction hypothesis, we assume that the groups $\text{Fix}(\partial \hat{A}^*)$ can be constructed using (a)–(f).

Lemma 8.1.4 describes the group $\text{Fix}(\partial A)$, depending on the type of the atom A. Also, Lemma 8.1.5 generalizes to planar symmetric expanded atoms $\hat{A}^*$. Therefore, every planar symmetric atom $\hat{A}$ has an involution $\hat{\tau}^* \in \text{Aut}_{\partial \hat{A}^*}(\hat{A}^*)$ swapping the boundary ∂A^*. Therefore, we get (7.4).



Ψ_1
u
v
A
Ψ_2
$\Psi_1 \times \Psi_2$
(b)
Ψ
$\Psi^4 \rtimes \mathbb{S}_4$
$\Psi^6 \rtimes \mathbb{C}_6$
(c)
Ψ_1
Ψ_2
Ψ_3
$(\Psi_1^{10} \times \Psi_2^5 \times \Psi_3^5) \rtimes \mathbb{D}_5$
(d)
Ψ_4
Ψ_5
Ψ_6
$(\Psi_1^{12} \times \Psi_2^6 \times \Psi_3^6 \times \Psi_4^6 \times \Psi_5^6) \rtimes \mathbb{D}_6$
(e)
$(\Psi_1^4 \times \Psi_2^2 \times \Psi_3^2 \times \Psi_4^2 \times \Psi_5^2 \times \Psi_6) \rtimes \mathbb{C}_2^2$
(f)

Figure 8.2: Constructions for the operations (b)–(f), every colored edge corresponds to an atom $\hat{A}$ with $\mathrm{Fix}(\partial\hat{A})$ isomorphic to the denoted group. In (d), we have three equivariant classes of edge-orbits in the action of $\mathbb{D}_n$ when n is odd. In (e), we have two further equivariant classes of edge-orbits in the action of $\mathbb{D}_n$ when $n \geq 4$ is even. In (f), there is one extra equivariant class consisting of one edge-orbit in the action of $\mathbb{D}_2 = \mathbb{C}_2^2$, generated by two reflections.

Below, we divide the proof into several lemmas, according to the type of A, and further simplify (7.4) to get the operations (b) to (f). Suppose that two edge-orbits of A, corresponding to expanded atoms $\hat{A}_1^*$ and $\hat{A}_2^*$, respectively, are equivariant in $\mathrm{Fix}(\partial A)$. Then using (b), we can construct $\mathrm{Fix}(\partial\hat{A}_1^*) \times \mathrm{Fix}(\partial\hat{A}_2^*)$ and work with it, using distributivity, as with one group in (7.4). Therefore, we need to identity all equivariance classes of edge-orbits in $\mathrm{Fix}(\partial A)$.

The following two types of edge-orbits are considered. An edge-orbit of size k is called *fixed*, denoted $\boldsymbol{k}$, if the corresponding half-edges form two orbits of size k. An edge-orbit of size k is called *reflected*, denoted $\boldsymbol{k}_{\leftrightarrow}$, if the corresponding half-edges form one orbit of size $2k$. We distinguish different geometric actions on the set of half-edges $\boldsymbol{H}(A)$, so a fixed edge-orbit of size k is non-equivariant with a reflected orbit of size k.

Dipoles and Star Block Atoms.

Lemma 8.2.3. *Let A be a star block atom or a dipole. Then* $\mathrm{Fix}(\partial A^*)$ *can be constructed from the groups* $\mathrm{Fix}(\partial\hat{A}^*)$*, using operations (b) and operations (c) for symmetric groups, where $\hat{A}$ ranges through all atoms corresponding to colored edges in A.*

Proof. The edges of the same type (for dipoles, we have undirected, directed in one way, directed in the other way) and color can be arbitrarily permuted. By Lemma 8.1.4, the action of $\mathrm{Fix}(\partial A)$ has ℓ orbits, each consisting of all colored edges of one color and of the same type and orientation. These orbits have sizes $m_1, \dots, m_\ell$, so $\mathrm{Fix}(\partial A) \cong \mathbb{S}_{m_1} \times \cdots \times \mathbb{S}_{m_\ell}$. Colored edges in these orbits correspond to atoms $\hat{A}_1, \dots, \hat{A}_\ell$.

Since $\mathrm{Fix}(\partial A^*)$ acts independently on the atoms corresponding to each orbit of colored edges in $\mathrm{Fix}(\partial A)$, each orbit contributes by one factor and $\mathrm{Fix}(\partial A^*)$ is the direct product of these factors. The atoms corresponding to each orbit can be arbitrarily permuted, thus each factor is isomorphic to $\mathrm{Fix}(\partial A_i^*) \wr \mathbb{S}_{m_i}$. □

Proper Atoms.

Lemma 8.2.4. *Let A be a proper atom. Then $\mathrm{Fix}(\partial A^*)$ can be constructed from the groups $\mathrm{Fix}(\partial \hat{A}^*)$, using operations (b) and operations (d) for $\mathbb{D}_1 \cong \mathbb{C}_2$, where $\hat{A}$ ranges through all atoms corresponding to colored edges in A.*

Proof. By Lemma 8.1.4, we know that $\mathrm{Fix}(\partial A)$ is a subgroup of $\mathbb{C}_2$. If $\mathrm{Fix}(\partial A) \cong \mathbb{C}_1$, then $\mathrm{Fix}(\partial A^*)$ can be easily constructed only using (b). Otherwise, $\mathrm{Fix}(\partial A) \cong \mathbb{C}_2$. Then the non-trivial automorphism $\pi \in \mathrm{Fix}(\partial A)$ corresponds to a reflection through ∂A. Therefore, $\mathrm{Fix}(\partial A)$ has some edge-orbits of colored edges of size two, and at most two types of edge-orbits of colored edges of size one, as depicted in Fig. 8.3:

- *Edge-orbits of type* **2**. We have ℓ_1 equivariant edge-orbits of size two, whose edges are reflected to each other by π. The colored edges in these orbits correspond to atoms $A_1, \dots, A_{\ell_1}$.
- *Edge-orbits of type* **1**. We have ℓ_2 equivariant edge-orbits of size one, in which both half-edges forming each edge are fixed by π, together with the incident vertices. The colored edges in these orbits correspond to atoms $B_1, \dots, B_{\ell_2}$.
- *Edge-orbits of type* $\mathbf{1}_{\leftrightarrow}$. We have ℓ_3 equivariant edge-orbits of size one, in which the half-edges forming each edge are exchanged by π. Therefore, these half-edges belong to one orbit, the incident vertices also belong to one orbit and

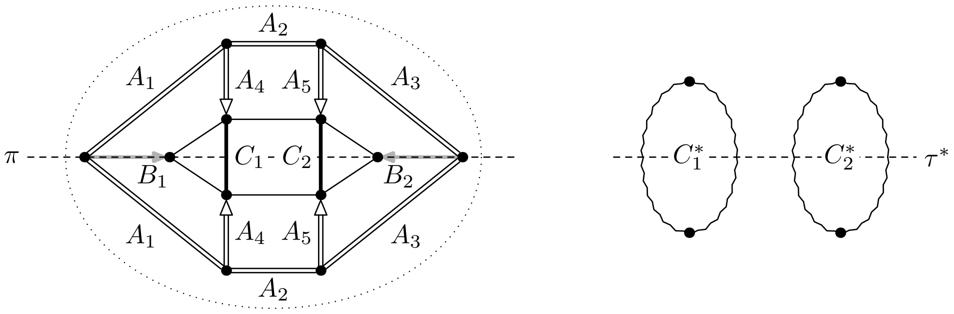


Figure 8.3: On the left, the action of $\mathrm{Fix}(\partial A)$ is generated by the reflection π. Observe that π acts differently on the edges corresponding to B_1 and B_2 (π fixes them) than on the edges corresponding to C_1 and C_2 (π reflects them). Therefore, in $\mathrm{Fix}(\partial A^*)$, we compose π with an involution τ^* reflecting C_1^* and C_2^*, depicted on the right.

the corresponding edges are reflected by π. The colored edges in these orbits correspond to (necessarily) symmetric atoms $C_1, \dots, C_{\ell_3}$. Let $\tau \in \mathrm{Aut}_{\partial C_1}(C_1) \times \cdots \times \mathrm{Aut}_{\partial C_{\ell_3}}(C_{\ell_3})$ be an involution which exchanges the boundaries of each of these atoms (ensured by Lemma 8.1.5), and τ^* be a corresponding involution in $\mathrm{Aut}_{\partial C_1^*}(C_1^*) \times \cdots \times \mathrm{Aut}_{\partial C_{\ell_3}^*}(C_{\ell_3}^*)$.

We need to distinguish two equivariant classes of edge-orbits of size one since the reflection π behaves differently with respect to them. For size one, fixed, two half-edges also form orbits of size one. On the other hand, for size one, reflected, both half-edges belong to the same orbit of size one. In π^*, the boundaries of B_i^* are fixed, but the boundaries of C_i^* are swapped, by applying τ^* on C_i^*. To be able to distinguish these two cases in A, it is important to consider automorphisms on half-edges instead of edges.

To construct $\mathrm{Fix}(\partial A^*)$, we put

$$\Psi_1 = \prod_{i=1}^{\ell_1} \mathrm{Fix}(\partial A_i^*), \qquad \Psi_2 = \prod_{i=1}^{\ell_2} \mathrm{Fix}(\partial B_i^*), \qquad \Psi_3 = \prod_{i=1}^{\ell_3} \mathrm{Fix}(\partial C_i^*)$$

using (b). Then it easily follows that

$$\mathrm{Fix}(\partial A^*) \cong (\Psi_1^2 \times \Psi_2 \times \Psi_3) \rtimes_\varphi \mathbb{C}_2, \tag{8.1}$$

where φ is the homomorphism defined as

$$\varphi(0) = \mathrm{id}, \qquad \varphi(1) = (\alpha_1, \alpha_1', \alpha_2, \alpha_3) \mapsto (\alpha_1', \alpha_1, \alpha_2, \tau^* \cdot \alpha_3),$$

$\alpha_1 \in \Psi_1$, $\alpha_1' \in \Psi_1' \cong \Psi_1$, $\alpha_2 \in \Psi_2$, and $\alpha_3 \in \Psi_3$. So $\mathrm{Fix}(\partial A^*)$ can be constructed using (b) and (d) (since $\mathbb{D}_1 \cong \mathbb{C}_2$). □

We note that semidirect product in (8.1) can be further simplified into $(\Psi_1^2 \times \Psi_3) \rtimes_\varphi \mathbb{C}_2 \times \Psi_2$ since φ acts trivially on the coordinate corresponding to Ψ_2. So the operation (d) for $\mathbb{D}_1 \cong \mathbb{C}_2$ could be simplified. We use this simplification in Section 8.3.

Non-star Block Atoms. Now, we deal with non-star block atoms A which are the most involved. By Lemma 8.1.4, we know that $\mathrm{Fix}(\partial A)$ is a subgroup of a dihedral group, so it is isomorphic either to $\mathbb{C}_n$, or to $\mathbb{D}_n$. By Lemma 7.4.3, A is either K_2 with an attached pendant edge, essentially a cycle, or essentially 3-connected. In the first two cases, $\mathrm{Fix}(\partial A)$ is a subgroup of $\mathbb{C}_2$. If $\mathrm{Fix}(\partial A)$ is not isomorphic to a subgroup of $\mathbb{C}_2$, then A is necessarily an essentially 3-connected graph.

Lemma 8.2.5. *Let A be a non-star block atom with $\mathrm{Fix}(\partial A) \cong \mathbb{C}_n$. Then $\mathrm{Fix}(\partial A^*)$ can be constructed from the groups $\mathrm{Fix}(\partial \hat{A}^*)$, using operations (b), operations (c) for $\mathbb{C}_n$, and operations (d) for $\mathbb{D}_1 \cong \mathbb{C}_2$, where $\hat{A}$ ranges through all atoms corresponding to colored edges in A.*

Proof. When $\mathrm{Fix}(\partial A) \cong \mathbb{C}_1$, it has no non-trivial automorphism and $\mathrm{Fix}(\partial A^*)$ can be constructed using only (b). When $\mathrm{Fix}(\partial A) \cong \mathbb{C}_2$, it has a single non-trivial automorphism π which is either a reflection or a 180° rotation around ∂A. In the case of

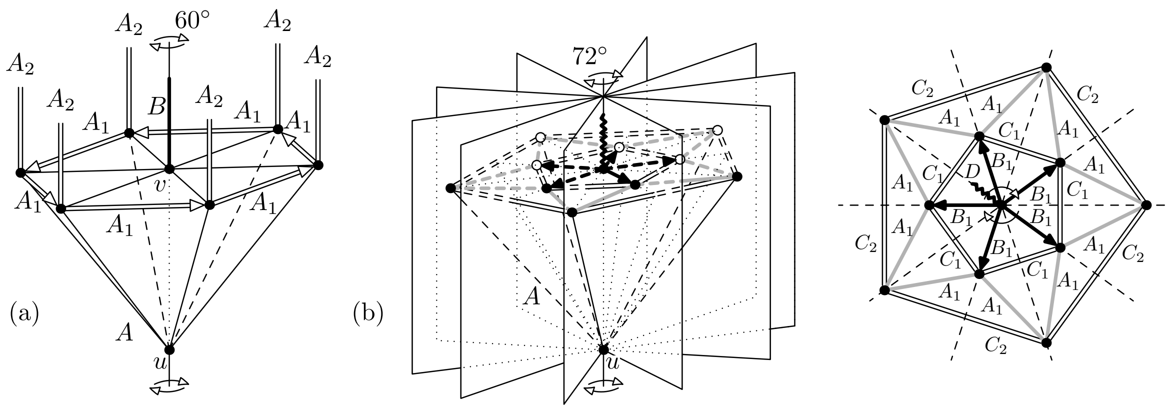


Figure 8.4: (a) An example of a non-star block atom A with $\mathrm{Fix}(\partial A) \cong \mathbb{C}_6$ generated by the 60° rotation through u and v. We have two orbits of colored edges of size 6, whose edges correspond to atoms A_1 and A_2. Further, the vertex v has attached a single pendant edge, corresponding to a block atom B.
(b) On the left, an example of a non-star block atom A with $\mathrm{Fix}(\partial A) \cong \mathbb{D}_5$ consisting of five rotations by multiples of 72° and five depicted reflections. On the right, the view from above, with colored edges labeled by the corresponding atoms. Different types of edge-orbits are depicted with different types of edges. We have $\ell_1 = \ell_2 = 1$ and $\ell_3 = 2$.

the 180° rotation, there is at most one edge-orbit of size 1 which is either fixed (for a pendant edge), or reflected (for a normal edge). Further, we proceed similarly as in the case of a proper atom in Lemma 8.2.4 to prove that $\mathrm{Fix}(\partial A^*)$ can be constructed using (b) and (d).

Suppose that $\mathrm{Fix}(\partial A) \cong \mathbb{C}_n$ for some $n \geq 3$. Recall that A is essentially 3-connected. The situation is depicted in Fig. 8.4a. The group $\mathrm{Fix}(\partial A)$ is the stabilizer of the unique vertex u in ∂A. Recall from Section 8.1 that in the language of isometries its action is generated by a rotation around u and the opposite point of the sphere. Therefore, every edge-orbit of $\mathrm{Fix}(\partial A)$ is of size one or n. All the edge-orbits of size n are equivariant. Suppose that the action of $\mathrm{Fix}(\partial A)$ consists of ℓ equivariant edge-orbits of colored edges of size n. The colored edges in these edge-orbits correspond to atoms $A_1, \dots, A_\ell$.

The opposite point of the sphere is a vertex v or the center of a face. (Since $n \geq 3$, it cannot contain the center of an edge.) In the former case, there might be an edge-orbit of size one consisting of a single pendant edge attached to v, and suppose that this pendant edge corresponds to a block atom B. In the later case, there is no edge-orbit of size one.

Let $\Psi = \mathrm{Fix}(\partial A_1^*) \times \cdots \times \mathrm{Fix}(\partial A_\ell^*)$, we can construct it using (b). Then we get

$$\mathrm{Fix}(\partial A^*) \cong \Psi \wr \mathbb{C}_n \times \mathrm{Fix}(\partial B^*),$$

where $\mathrm{Fix}(\partial B^*) = \{1\}$ if no edge-orbit of size one exists. So we construct $\mathrm{Fix}(\partial A^*)$ using (b) and (c). □

It remains to deal with dihedral groups. First, we determine the possible counts of equivariant classes of edge-orbits.

Lemma 8.2.6. *Let A be a non-star block atom with* $\mathrm{Fix}(\partial A) \cong \mathbb{D}_n$.

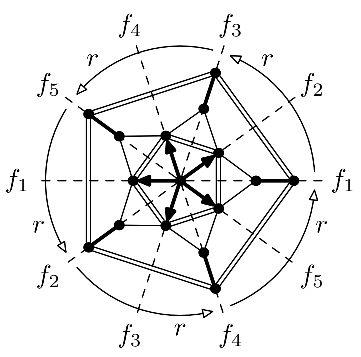
- *If n is odd, then all edge-orbits of type $\boldsymbol{n}$ are equivariant and all edge-orbits of type $\boldsymbol{n}_{\leftrightarrow}$ are equivariant.*
- *If n is even, then there are at most two equivariant classes of edge-orbits of types $\boldsymbol{n}$ and $\boldsymbol{n}_{\leftrightarrow}$.*

Proof. The statement clearly holds for $n = 1$, so in what follows, we assume $n \geq 2$. Recall from Section 8.1 that in the language of isometries the action of $\mathrm{Fix}(\partial A)$ consists of n rotations and n reflections. Each rotation fixes only ∂A and the opposite point of the sphere, and each reflection fixes a circle containing ∂A and the opposite point. Let r be the rotation by $360°/n$, then all rotations are $\mathrm{id}, r, r^2, \dots, r^{n-1}$. Let $f_1, f_2, \dots, f_n$ be the reflections as their planes are ordered cyclically, perpendicular to the axis of the rotations; see Fig. 8.5. These reflections are cyclically linked by the conjugation $f_{i+2} = r^{-1} f_i r$. The key distinction is that for n odd, all reflections are conjugate of each other, but for n even, we get two conjugacy classes $f_1, f_3, \dots, f_{n-1}$ and $f_2, f_4, \dots, f_n$.

Let e be an edge belonging to a fixed/reflected edge-orbit $[e]$ of size n. The rotation r does not stabilize any edge in $[e]$, so each edge is stabilized by some reflection. From the geometry, $[e] = \{e, r \cdot e, r^2 \cdot e, \dots, r^{n-1} e\}$. Suppose that e is stabilized by f_i. Then $r \cdot e$ is stabilized by f_{i+2}, $r^2 \cdot e$ by f_{i+4}, and so on.

When n is odd, each reflection stabilizes exactly one edge in $[e]$; see Fig.8.5 on the left. Therefore, two edge-orbits $[e]$ and $[e']$ of size n, both fixed or reflected, are equivariant: the edges in $[e]$ and $[e']$ having the same stabilizer can be matched.

When n is even, only one conjugacy class of reflections stabilizes edges in $[e]$, each stabilizing $r^i \cdot e$ and $r^{i+n/2} \cdot e$. When two edge-orbits $[e]$ and $[e']$ of size n, both fixed or reflected, are stabilized by the same conjugacy class of reflections, they are equivariant. Therefore, we get at most two equivariant classes of both fixed and reflected edge-orbits of size n. ◻



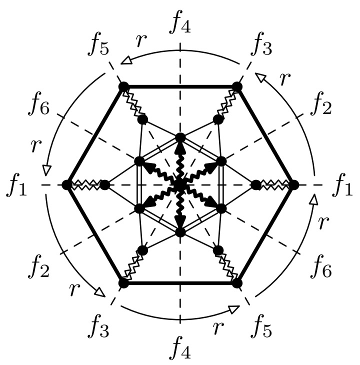


Figure 8.5: Two block atoms A with $\mathrm{Fix}(\partial A) \cong \mathbb{D}_n$ in the view from above, with $n = 5$ on the left and $n = 6$ on the right. The rotation r and the reflections $f_1, \dots, f_n$ are denoted. On the left, each edge of an edge-orbit of size n is stabilized by exactly one reflection. On the right, each pair of opposite edges of an edge-orbit of size n is stabilized by exactly one reflection from one of the conjugacy classes $f_1, f_3, \dots, f_{n-1}$ and $f_2, f_4, \dots, f_n$.

To specify the semidirect products in (d) and (e), we describe the action of $\mathbb{D}_n$ on an edge-orbit $[e]$ of size n. The rotation r maps $r^k \cdot e$ to $r^{k+1} \cdot e$. When the reflection f_i stabilizes $e' \in [e]$, then it swaps $r \cdot e'$ with $r^{-1} \cdot e'$, $r^2 \cdot e'$ with $r^{-2} \cdot e'$, and so on; it fixes e' and for n even also $r^{n/2} \cdot e'$. The reflection f_{i+1} swaps $r \cdot e'$ with e', $r^2 \cdot e'$ with $r^{-1} \cdot e'$, $r^3 \cdot e'$ with $r^{-2} \cdot e'$, and so on. As stated, the reflection f_{i+2} stabilizes $r^2 \cdot e'$, and so on.

Let h and h' be the half-edges corresponding to e. Consider $2n$ half-edges corresponding to edges in $[e]$. In the action of $\langle r \rangle$, they form two orbits $\{h, r \cdot h, r^2 \cdot h, \dots, r^{n-1} \cdot h\}$ and $\{h', r \cdot h', r^2 \cdot h', \dots, r^{n-1} \cdot h'\}$ of size n. When $[e]$ is fixed, these two orbits are preserved in the action of $\mathbb{D}_n$. When $[e]$ is reflected, each reflection f_k swaping $r^i \cdot e$ with $r^j \cdot e$ swaps $r^i \cdot h$ with $r^j \cdot h'$ and $r^i \cdot h'$ with $r^j \cdot h$, so we get one orbit of half-edges of size $2n$ in the action of $\mathbb{D}_n$. We note that the action of $\mathbb{D}_n$ on an edge-orbit of size $2n$ is the same as the action on the half-edges corresponding to $\boldsymbol{n}_{\leftrightarrow}$.

When $\text{Fix}(\partial A) \cong \mathbb{D}_1 \cong \mathbb{C}_2$, we use Lemma 8.2.5. We start with an easier case of $\text{Fix}(\partial A) \cong \mathbb{D}_n$, for $n \geq 3$, odd.

Lemma 8.2.7. *Let A be a non-star block atom with $\text{Fix}(\partial A) \cong \mathbb{D}_n$ for $n \geq 3$ and odd. Then $\text{Fix}(\partial A^*)$ can be constructed from the groups $\text{Fix}(\partial \hat{A}^*)$, using operations (b), and operations (d), where $\hat{A}$ ranges through all atoms corresponding to colored edges in A.*

Proof. As it is described in the proof of Lemma 8.2.6, the group $\text{Fix}(\partial A) \cong \mathbb{D}_n$ consists of n rotations and n reflections; see Fig. 8.4b. It acts semiregularly on the angles of the map and all edge-orbits are of size one, n, or $2n$. By Lemma 8.2.6, all fixed/reflected edge-orbits of size n are equivariant and $\text{Fix}(\partial A)$ acts on them as described below Lemma 8.2.6.

- *Edge-orbits of type* **1**. The opposite point of the sphere either contains a vertex v, or the center of a face. In the former case, there might be at most one edge-orbit of size one, consisting of a single pendant edge attached to v corresponding to a block atom D. In the latter case, no edge-orbit of size one exists.
- *Edge-orbits of type* $\boldsymbol{n}$. We have ℓ_2 equivariant fixed edge-orbits of colored edges of size n, corresponding to atoms $B_1, \dots, B_{\ell_2}$.
- *Edge-orbits of type* $\boldsymbol{n}_{\leftrightarrow}$. We have ℓ_3 equivariant reflected edge-orbits of colored edges of size n, corresponding to necessarily symmetric atoms $C_1, \dots, C_{\ell_3}$. Let $\tau_i^* \in \text{Aut}_{\partial C_i^*}(C_i^*)$ be an involution exchaning ∂C_i^*, ensured by Lemma 8.1.5.
- *Edge-orbits of type* $\boldsymbol{2n}$. We have ℓ_1 equivariant edge-orbits of colored edges of size $2n$, corresponding to atoms $A_1, \dots, A_{\ell_1}$.

We put

$$\Psi_1 = \prod_{i=1}^{\ell_1} \text{Fix}(\partial A_i^*), \qquad \Psi_2 = \prod_{i=1}^{\ell_2} \text{Fix}(\partial B_i^*), \qquad \Psi_3 = \prod_{i=1}^{\ell_3} \text{Fix}(\partial C_i^*).$$

It follows that

$$\text{Fix}(\partial A^*) \cong (\Psi_1^{2n} \times \Psi_2^n \times \Psi_3^n) \rtimes_\varphi \mathbb{D}_{2n} \times \text{Fix}(\partial D^*),$$

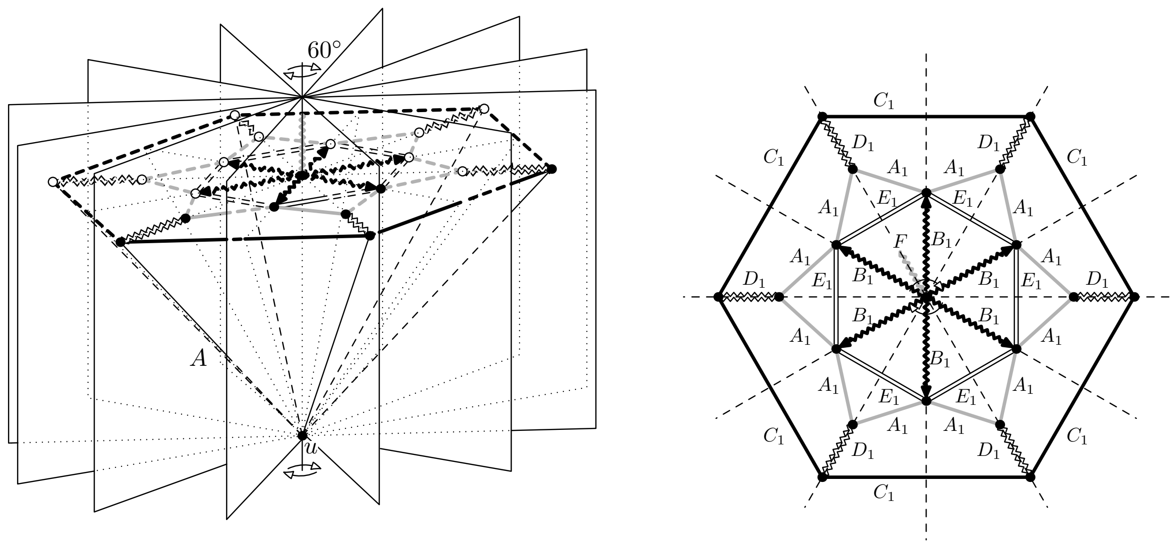


Figure 8.6: On the left, an example of a non-star block atom A with $\mathrm{Fix}(\partial A) \cong \mathbb{D}_6$ consisting of 6 rotations by multiples of 60° and six depicted reflections. On the right, the view from above, with colored edges labeled by the corresponding atoms. Different types of edge-orbits are depicted with different types of edges. We have $\ell_1 = \ell_2 = \ell_3 = \ell_4 = \ell_5 = 1$.

where $\mathrm{Fix}(\partial D^*) = \{1\}$ if there is no edge-orbit of size one. The homomorphism φ is defined based on the description of the action of $\mathbb{D}_{2n}$ below Lemma 8.2.6. It permutes the coordinates of Ψ_1^{2n} regularly as $\mathbb{D}_{2n}$ acts on half-edges of a reflected edge-orbit of size n. It permutes the coordinates in Ψ_2^n and Ψ_3^n following the action on the edges of fixed and reflected edge-orbits of size n, respectively. For the edges of reflected edge-orbits corresponding to C_i^*, when half-edges are swapped by an element $\pi \in \mathrm{Fix}(\partial A)$, the involution τ_i^* is used in the action of $\pi^* \in \mathrm{Fix}(\partial A^*)$ on the corresponding atoms C_i^*. Therefore, $\mathrm{Fix}(\partial A^*)$ can be constructed using (b) and (d). □

Next, we deal with the case $\mathrm{Fix}(\partial A) \cong \mathbb{D}_n$, for $n \geq 4$ and even.

Lemma 8.2.8. *Let A be a non-star block atom with $\mathrm{Fix}(\partial A) \cong \mathbb{D}_n$ for $n \geq 4$, even. Then $\mathrm{Fix}(\partial A^*)$ can be constructed from the groups $\mathrm{Fix}(\partial \hat{A}^*)$, using operations (b), and operations (e), where $\hat{A}$ ranges through all atoms corresponding to colored edges in A.*

Proof. As it is described in the proof of Lemma 8.2.6, the group $\mathrm{Fix}(\partial A) \cong \mathbb{D}_n$ consists of n rotations and n reflections; see Fig. 8.6. It acts semiregularly on the angles of the map and all edge-orbits are of size one, n, or $2n$. By Lemma 8.2.6, there are at most two equivariant classes of fixed/reflected edge-orbits of size n and $\mathrm{Fix}(\partial A)$ acts on them as described below Lemma 8.2.6.

- *Edge-orbits of type* **1**. Exactly as in the proof of Lemma 8.2.7, there is at most one edge-orbit of size one, consisting of a single pendant edge corresponding to a block atom F.
- *Edge-orbits of type* ***n***. We have two equivariant classes of ℓ_2 and ℓ_4 fixed edge-orbits of colored edges of size n, corresponding to atoms $B_1, \dots, B_{\ell_2}$ and $D_1, \dots, D_{\ell_4}$, respectively.

- *Edge-orbits of type* $\boldsymbol{n}_{\leftrightarrow}$. We have two equivariance classes of ℓ_3 and ℓ_5 reflected edge-orbits of colored edges of size n, corresponding to necessarily symmetric atoms $C_1, \dots, C_{\ell_3}$ and $E_1, \dots, E_{\ell_5}$. Let $\tau_i^* \in \mathrm{Aut}_{\partial C_i^*}(C_i^*)$ be an involution exchanging ∂C_i^* and let $\hat{\tau}_i^* \in \mathrm{Aut}_{\partial E_i^*}(E_i^*)$ be an involution exchanging ∂E_i^*, ensured by Lemma 8.1.5.
- *Edge-orbits of type* $\boldsymbol{2n}$. We have ℓ_1 equivariant edge-orbits of colored edges of size $2n$, corresponding to atoms $A_1, \dots, A_{\ell_1}$.

We put

$$\Psi_1 = \prod_{i=1}^{\ell_1} \mathrm{Fix}(\partial A_i^*), \qquad \Psi_2 = \prod_{i=1}^{\ell_2} \mathrm{Fix}(\partial B_i^*), \qquad \Psi_3 = \prod_{i=1}^{\ell_3} \mathrm{Fix}(\partial C_i^*),$$

$$\Psi_4 = \prod_{i=1}^{\ell_4} \mathrm{Fix}(\partial D_i^*), \qquad \Psi_5 = \prod_{i=1}^{\ell_5} \mathrm{Fix}(\partial E_i^*).$$

It follows that

$$\mathrm{Fix}(\partial A^*) \cong (\Psi_1^{2n} \times \Psi_2^n \times \Psi_3^n \times \Psi_4^n \times \Psi_5^n) \rtimes_\varphi \mathbb{D}_{2n} \times \mathrm{Fix}(\partial F^*),$$

where $\mathrm{Fix}(\partial F^*) = \{1\}$ if there is no edge-orbit of size one. The homomorphism φ is defined based on the description of the action of $\mathbb{D}_{2n}$ below Lemma 8.2.6. It permutes the coordinates of Ψ_1^{2n} regularly following the action of $\mathbb{D}_{2n}$ on half-edges of a reflected edge-orbit of size n. It permutes the coordinates in Ψ_2^n, Ψ_3^n, Ψ_4^n, and Ψ_5^n in the same way as the edges of two equivariance classes of fixed and reflected edge-orbits of size n, respectively. For the edges of reflected edge-orbits corresponding to C_i^* and E_i^*, when half-edges are swapped by an element $\pi \in \mathrm{Fix}(\partial A)$, the involutions τ_i^* and $\hat{\tau}_i^*$ are used in the action of $\pi^* \in \mathrm{Fix}(\partial A^*)$ on the corresponding atoms C_i^* and E_i^*, respectively. Therefore, $\mathrm{Fix}(\partial A^*)$ can be constructed using (b) and (e). □

It remains to deal with the last case of $\mathrm{Fix}(\partial A) \cong \mathbb{D}_2 \cong \mathbb{C}_2^2$ which may have the most involved structure of edge-orbits.

Lemma 8.2.9. *Let A be a non-star block atom with* $\mathrm{Fix}(\partial A) \cong \mathbb{D}_2 \cong \mathbb{C}_2^2$. *Then* $\mathrm{Fix}(\partial A^*)$ *can be constructed from the groups* $\mathrm{Fix}(\partial \hat{A}^*)$, *using operations (b) and (f), where $\hat{A}$ ranges through all atoms corresponding to colored edges in A.*

Proof. As it is described in the proof of Lemma 8.2.6, the group $\mathrm{Fix}(\partial A) \cong \mathbb{D}_2 \cong \mathbb{C}_2^2$ is generated by two reflections $(1,0)$ and $(0,1)$ through ∂A, orthogonal to each other. Composition of these two reflections forms the $180°$ rotation $(1,1)$ through ∂A and the opposite point of the sphere. (We identified the geometric transformations with the elements of the elementary abelian group of order 4.) Therefore, every edge-orbit of $\mathrm{Fix}(\partial A)$ is of size one, two, or four, and we describe them below; see Fig. 8.7 for an example.

- *Edge-orbits of type* $\mathbf{1}$ *and of type* $\mathbf{1}_{\leftrightarrow}$. The rotation $(1,1)$ and the reflections $(1,0)$ and $(0,1)$ stabilize, aside ∂A, the opposite point of the sphere which contains either a vertex, or the center of an edge, or the center of a face.

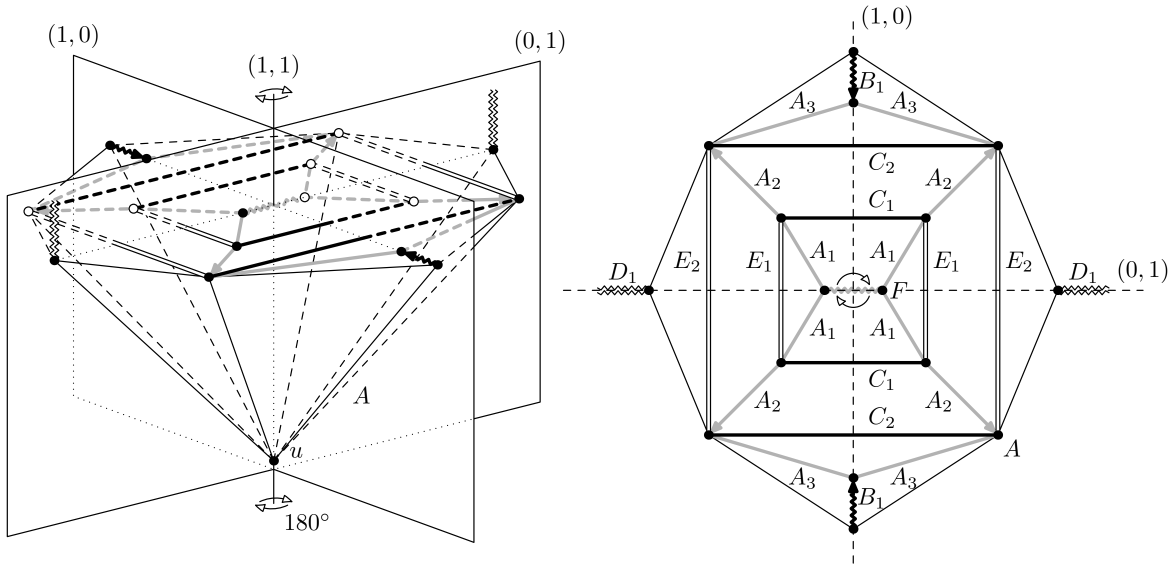


Figure 8.7: On the left, an example of a non-star block atom A with $\mathrm{Fix}(\partial A) \cong \mathbb{C}_2^2$ generated by two depicted reflections. On the right, the view from above, with colored edges labeled by the corresponding atoms, the central edge corresponds to an atom F. Different types of edge-orbits are depicted with different types of edges. We have $\ell_1 = 3$, $\ell_2 = \ell_4 = 1$ and $\ell_3 = \ell_5 = 2$.

  - If they stabilize the center of a face, there is no edge-orbit of size 1.
  - If they stabilize a vertex v, there might be a fixed edge-orbit of size 1 consisting of a single pendant edge attached at v. We deal with it using (b) as in the proofs of Lemmas 8.2.5, 8.2.7 and 8.2.8, and we put $\Psi_6 = \{1\}$ and $\tau_6^* = \mathrm{id}$.
  - If they stabilize the center of an edge e, then $[e]$ is a reflected edge-orbit of size 1. Let F be a symmetric atom corresponding to the colored edge e and we put $\Psi_6 = \mathrm{Fix}(\partial F^*)$. By Lemma 8.1.5, there exists an involution τ_6 exchanging ∂F, and let τ_6^* be a corresponding involution in $\mathrm{Aut}_{\partial F^*}(F^*)$.

- *Edge-orbits of type* **2** *and of type* **2**$_\leftrightarrow$. By Lemma 8.2.6, there are at most two equivariant classes of fixed/reflected edge-orbits of size 2, one class stabilized by $(1, 0)$ and the other one by $(0, 1)$.
  - There are ℓ_2 equivariant edge-orbits of colored edges fixed by the reflection $(1, 0)$. These colored edges correspond to atoms $B_1, \dots, B_{\ell_2}$.
  - There are ℓ_3 equivariant edge-orbits of colored edges reflected by $(1, 0)$. These colored edges correspond to symmetric atoms $C_1, \dots, C_{\ell_3}$. Let $\tau_3^* \in \mathrm{Fix}(\partial C_1^*) \times \cdots \times \mathrm{Fix}(\partial C_{\ell_3}^*)$ be an involution exchanging their boundaries.
  - There are ℓ_4 equivariant edge-orbits of colored edges fixed by the reflection $(0, 1)$. These colored edges correspond to atoms $D_1, \dots, D_{\ell_4}$.
  - There are ℓ_5 equivariant edge-orbits of colored edges reflected by $(0, 1)$. These colored edges correspond to symmetric atoms $E_1, \dots, E_{\ell_3}$. Let $\tau_5^* \in \mathrm{Fix}(\partial E_1^*) \times \cdots \times \mathrm{Fix}(\partial E_{\ell_5}^*)$ be an involution exchanging their boundaries.
- *Edge-orbits of type* **4**. The group $\mathrm{Fix}(\partial A)$ acts regularly on edge-orbits of size four. Suppose we have ℓ_1 equivariant edge-orbits of colored edges of size four,

and these colored edges correspond to atoms $A_1, \dots, A_{\ell_1}$.

We put

$$\Psi_1 = \prod_{i=1}^{\ell_1} \mathrm{Fix}(\partial A_i^*), \qquad \Psi_2 = \prod_{i=1}^{\ell_2} \mathrm{Fix}(\partial B_i^*), \qquad \Psi_3 = \prod_{i=1}^{\ell_3} \mathrm{Fix}(\partial C_i^*),$$
$$\Psi_4 = \prod_{i=1}^{\ell_4} \mathrm{Fix}(\partial D_i^*), \qquad \Psi_5 = \prod_{i=1}^{\ell_5} \mathrm{Fix}(\partial E_i^*).$$

It easily follows that

$$\mathrm{Fix}(\partial A^*) \cong (\Psi_1^4 \times \Psi_2^2 \times \Psi_3^2 \times \Psi_4^2 \times \Psi_5^2 \times \Psi_6) \rtimes_\varphi \mathbb{C}_2^2.$$

Assuming that $(1,0)$ reverses the edge e_6, the homomorphism φ is defined by

$$\begin{aligned}
\varphi(1,0) &= (\pi_1,\pi_1',\pi_1'',\pi_1''',\pi_2,\pi_2',\pi_3,\pi_3',\pi_4,\pi_4',\pi_5,\pi_5',\pi_6) \mapsto \\
&\qquad (\pi_1',\pi_1,\pi_1''',\pi_1'',\pi_2,\pi_2',\tau_3^*\cdot\pi_3,\tau_3^*\cdot\pi_3',\pi_4',\pi_4,\pi_5',\pi_5,\tau_6^*\cdot\pi_6),\\
\varphi(0,1) &= (\pi_1,\pi_1',\pi_1'',\pi_1''',\pi_2,\pi_2',\pi_3,\pi_3',\pi_4,\pi_4',\pi_5,\pi_5',\pi_6) \mapsto \\
&\qquad (\pi_1'',\pi_1''',\pi_1,\pi_1',\pi_2',\pi_2,\pi_3',\pi_3,\pi_4,\pi_4',\tau_5^*\cdot\pi_5,\tau_5^*\cdot\pi_5',\pi_6),
\end{aligned}$$

where $\tau_6^* = \mathrm{id}$ if e_6 does not exist. Therefore, $\mathrm{Fix}(\partial A^*)$ can be constructed using (b) and (f). □

We are ready to prove the Jordan-like characterization of Fix(connected PLANAR).

Proof of Theorem 8.2.1. Lemma 8.2.2 describes constructions. We prove the opposite implication by induction according to the depth of the subtree of a reduction tree. Let A be an atom and suppose that the subtrees rooted at all its children can be realized by (b) to (f). By Lemmas 8.2.3, 8.2.4, 8.2.5, 8.2.7, 8.2.8, and 8.2.9, also $\mathrm{Fix}(\partial A^*)$ can be realized by (b) to (f). □

8.2.3 Composition of Spherical groups with Fixers

It remains to deal with the root of the reduction tree, corresponding to the primitive graph G_r. In comparison with atoms, $\mathrm{Aut}(G_r)$ does not have to stabilize any vertex or edge, unlike $\mathrm{Fix}(\partial A)$ which stabilizes ∂A. Therefore, all spherical groups are available for $\mathrm{Aut}(G_r)$. Now, we are ready to prove the main result of this chapter, classifying Aut(connected PLANAR).

Theorem 8.2.10. *Let H be a planar graph with colored vertices and colored (possibly oriented) edges, which is either 3-connected, or K_1, or K_2, or a cycle C_n. Let $m_1, \dots, m_\ell$ be the sizes of the vertex- and edge-orbits of the action of* $\mathrm{Aut}(H)$*. Then for all choices $\Psi_1, \dots, \Psi_\ell \in$ Fix(connected PLANAR), we have*

$$(\Psi_1^{m_1} \times \cdots \times \Psi_\ell^{m_\ell}) \rtimes \mathrm{Aut}(H) \in \mathrm{Aut}(\text{connected PLANAR}),$$

where $\mathrm{Aut}(H)$ *permutes the factors of $\Psi_1^{m_1} \times \cdots \times \Psi_\ell^{m_\ell}$ following the action on the vertices and edges of H.*

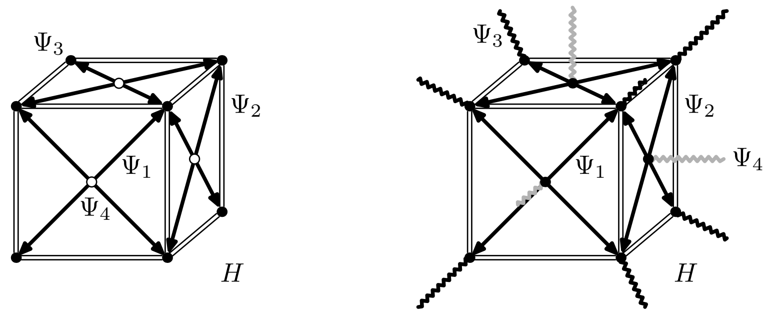


Figure 8.8: On the left, a 3-connected planar graph H obtained from the cube, with only three front faces depicted. We have $\mathrm{Aut}(H) \cong \mathbb{C}_2 \times \mathbb{S}_4$ and its action $*432$ in Table 8.1. Different orbits are shown in different colors: there are two vertex-orbits (of sizes 8 and 6) and two edge-orbits (of sizes 24 and 12).
On the right, H is modified by attaching single pendant edges of different colors for each vertex-orbit. For arbitrary choices of $\Psi_1, \Psi_2, \Psi_3, \Psi_4 \in \mathrm{Fix}(\textsf{connected PLANAR})$, we can expand H to H^* with $\mathrm{Aut}(H^*) \cong (\Psi_1^{24} \times \Psi_2^{12} \times \Psi_3^{8} \times \Psi_4^{6}) \rtimes (\mathbb{C}_2 \times \mathbb{S}_4)$. Notice that some automorphisms of $\mathrm{Aut}(H)$ reflect some white edges and the corresponding automorphism in $\mathrm{Aut}(H^*)$ reflect the expanded atoms corresponding to these edges by τ_2^*.

On the other hand, every group of $\mathrm{Aut}(\textsf{connected PLANAR})$ *can be constructed in the above way as*

$$(\Psi_1^{m_1} \times \cdots \times \Psi_\ell^{m_\ell}) \rtimes \Sigma,$$

where $\Psi_1, \dots, \Psi_\ell \in \mathrm{Fix}(\textsf{connected PLANAR})$ *and* Σ *is a spherical group.*

Proof. Let H be a graph satisfying the assumptions; for an example, see Fig. 8.8. First, we replace colors of the vertices of H with colored single pendant edges attached to them. Using Lemma 8.2.2, we choose arbitrary pairwise non-isomorphic extended atoms $A_1^*, \dots, A_\ell^*$ such that $\mathrm{Fix}(\partial A_i^*) \cong \Psi_i$, and we replace the corresponding colored edges with them. If the edge-orbit replaced by A_i^* consists of undirected edges, we assume that A_i^* are symmetric atoms, and let $\tau_i^* \in \mathrm{Aut}_{\partial A_i^*}(A_i^*)$ be an involution exchanging ∂A_i^*. If it consists of directed edges, we assume that A_i^* are asymmetric atoms placed consistently with the orientation. We denote this modified planar graph by H^*.

Exactly as in the proof of Proposition 7.6.4b, we get that

$$\mathrm{Aut}(H^*) \cong (\Psi_1^{m_1} \times \cdots \times \Psi_\ell^{m_\ell}) \rtimes_\varphi \mathrm{Aut}(H).$$

An automorphism $\pi^* \in \mathrm{Aut}(H^*)$ permutes the extended atoms exactly as $\pi \in \mathrm{Aut}(H)$ permutes the colored edges. If π reflects an edge representing a symmetric atom A_i^*, then π^* reflects A_i^* using τ_i^*.

For the other implication, let G be a planar graph. We apply the reduction series and obtain a primitive graph G_r. By Lemma 8.1.3, we know that $\mathrm{Aut}(G_r)$ is a spherical group. Suppose that we have ℓ edge-orbits of colored edges in the action of $\mathrm{Aut}(G_r)$. Suppose that their sizes are $m_1, \dots, m_\ell$ and their colored edges correspond to expanded atoms $A_1^*, \dots, A_\ell^*$. By Theorem 8.2.1, we know that $\mathrm{Fix}(\partial A_i^*) \in$

Fix(connected PLANAR). Further, for symmetric expanded atoms A_i^*, by Lemma 8.1.5, there exists an involution τ_i^* exchanging ∂A_i^*. We proceed exactly as in the proof of Proposition 7.6.4 and we obtain

$$\mathrm{Aut}(G) \cong \Bigl(\mathrm{Fix}(\partial A_1^*)^{m_1} \times \cdots \times \mathrm{Fix}(\partial A_\ell^*)^{m_\ell}\Bigr) \rtimes_\varphi \mathrm{Aut}(G_r).$$

□

8.2.4 Possible Lengths of Orbits

To describe the groups realizable as automorphism groups of connected planar graphs, we need to understand what are the possible restrictions on sizes m_i of the orbits of $\mathrm{Aut}(G)$ in a 3-connected planar graph G. (When G is K_1, K_2 or a cycle C_n, we get more restricted orbits than in the case of 3-connected planar graphs G. For instance, the wheel W_n is 3-connected and contains all orbits of C_n.) We investigate possible actions of spherical groups Σ realized as groups of isometries of polytopes projected onto the sphere.

In [218], the following characterization of possible equivariance classes of orbits given in Tables 8.1, 8.2, and 8.3 is proved. These tables are organized as follows. Each row of the table corresponds to a distinguished (parametrized) spherical group Σ described using the notation of Conway and Thurston [70]. There are fourteen types of actions, several small special cases are discussed separately. The second column describes Σ as an abstract group and the third column gives the order of Σ.

The fourth column gives the numbers of equivariance classes of *point-orbits* of Σ. By $c \cdot \boldsymbol{a}^b$, we denote that there are c equivariance classes of point-orbits of size a, each class of size b. For instance, in the second row of Table 8.1, the fourth entry contains $\mathbf{48}^\infty, 2\cdot\mathbf{24}^\infty, \mathbf{12}^1, \mathbf{8}^1, \mathbf{6}^1$. This means that there are infinitely many mutually equivariant point-orbits of size 48, three infinite equivariant classes of point-orbits of size 24, and single point-orbits of sizes 12, 8, and 6.

The fifth and sixth columns describe equivariance classes of vertex- and edge-orbits of Σ, respectively. In the second row of Table 8.1, there are two options for sequences of possible vertex- and edge-orbits:

either	$\mathbf{48}$,	$2\cdot\mathbf{24}$,	$\mathbf{12}$,	$\mathbf{8}$,	$\mathbf{6}$	and	$\mathbf{48}$,	$2\cdot\mathbf{24}$,	$2\cdot\mathbf{24}_\leftrightarrow$,	$-_\leftrightarrow$,
or	$\mathbf{48}$,	$2\cdot\mathbf{24}$,	$-$,	$\mathbf{8}$,	$\mathbf{6}$	and	$\mathbf{48}$,	$2\cdot\mathbf{24}$,	$2\cdot\mathbf{24}_\leftrightarrow$,	$\mathbf{12}_\leftrightarrow$.

The multiplicity of orbits in equivariance classes is not displayed. The difference between the two cases comes from the fact that the unique point-orbit of size 12 is either a vertex-orbit or an edge-orbit. The subscript $\leftrightarrow$ means that the edge-orbit is reflexive. Similarly as in the proofs in Section 8.2.2, we distinguish edge-orbits which are fixed (the corresponding half-edges form two orbits of the same size), depicted as $c\cdot\boldsymbol{a}$, and which are reflected, denoted as $c\cdot\boldsymbol{a}_\leftrightarrow$ (the corresponding half-edges form one orbit of the double size).

Action	Σ	$\lvert\Sigma\rvert$	Point-orbits	Vertex-orbits	Edge-orbits
*532	$\mathbb{A}_5 \times \mathbb{C}_2$	120	$\mathbf{120}^\infty, \mathbf{60}^\infty, \mathbf{30}^1, \mathbf{20}^1, \mathbf{12}^1$	$\mathbf{120},\ \mathbf{60},\ \mathbf{30},\ \mathbf{20},\ \mathbf{12}$ $\mathbf{120},\ \mathbf{60},\ -,\ \mathbf{20},\ \mathbf{12}$	$\mathbf{120},\ \mathbf{60},\ \mathbf{60}_\leftrightarrow,\ -_\leftrightarrow$ $\mathbf{120},\ \mathbf{60},\ \mathbf{60}_\leftrightarrow,\ \mathbf{30}_\leftrightarrow$
*432	$\mathbb{S}_4 \times \mathbb{C}_2$	48	$\mathbf{48}^\infty, 2\cdot\mathbf{24}^\infty, \mathbf{12}^1, \mathbf{8}^1, \mathbf{6}^1$	$\mathbf{48},\ 2\cdot\mathbf{24},\ \mathbf{12},\ \mathbf{8},\ \mathbf{6}$ $\mathbf{48},\ 2\cdot\mathbf{24},\ -,\ \mathbf{8},\ \mathbf{6}$	$\mathbf{48},\ 2\cdot\mathbf{24},\ 2\cdot\mathbf{24}_\leftrightarrow,\ -_\leftrightarrow$ $\mathbf{48},\ 2\cdot\mathbf{24},\ 2\cdot\mathbf{24}_\leftrightarrow,\ \mathbf{12}_\leftrightarrow$
*332	$\mathbb{S}_4$	24	$\mathbf{24}^\infty, \mathbf{12}^\infty, \mathbf{6}^1, \mathbf{4}^2$	$\mathbf{24},\ \mathbf{12},\ \mathbf{6},\ \mathbf{4}$ $\mathbf{24},\ \mathbf{12},\ -,\ \mathbf{4}$	$\mathbf{24},\ \mathbf{12},\ \mathbf{12}_\leftrightarrow,\ -_\leftrightarrow$ $\mathbf{24},\ \mathbf{12},\ \mathbf{12}_\leftrightarrow,\ \mathbf{6}_\leftrightarrow$
*22n	$\mathbb{D}_n \times \mathbb{C}_2$, $n \geq 3$, odd	$4n$	$(\mathbf{4n})^\infty, 2\cdot(\mathbf{2n})^\infty, \mathbf{n}^2, \mathbf{2}^1$	$\mathbf{4n}, 2\cdot\mathbf{2n}, \mathbf{n}, \mathbf{2}$	$\mathbf{4n}, 2\cdot\mathbf{2n}, 2\cdot\mathbf{2n}_\leftrightarrow, \mathbf{n}_\leftrightarrow$
*22n	$\mathbb{D}_n \times \mathbb{C}_2$, $n \geq 4$, even	$4n$	$(\mathbf{4n})^\infty, 3\cdot(\mathbf{2n})^\infty, 2\cdot\mathbf{n}^1, \mathbf{2}^1$	$\mathbf{4n},\ 3\cdot\mathbf{2n},\ 2\cdot\mathbf{n},\ \mathbf{2}$ $\mathbf{4n},\ 3\cdot\mathbf{2n},\ \mathbf{n},\ \mathbf{2}$ $\mathbf{4n},\ 3\cdot\mathbf{2n},\ -,\ \mathbf{2}$	$\mathbf{4n},\ 3\cdot\mathbf{2n},\ 3\cdot\mathbf{2n}_\leftrightarrow,\ -_\leftrightarrow$ $\mathbf{4n},\ 3\cdot\mathbf{2n},\ 3\cdot\mathbf{2n}_\leftrightarrow,\ \mathbf{n}_\leftrightarrow$ $\mathbf{4n},\ 3\cdot\mathbf{2n},\ 3\cdot\mathbf{2n}_\leftrightarrow,\ 2\cdot\mathbf{n}_\leftrightarrow$
*222	$\mathbb{D}_2 \times \mathbb{C}_2$ $= \mathbb{C}_2^3$	8	$\mathbf{8}^\infty, 3\cdot\mathbf{4}^\infty, 3\cdot\mathbf{2}^1$	$\mathbf{8},\ 3\cdot\mathbf{4},\ 3\cdot\mathbf{2}$ $\mathbf{8},\ 3\cdot\mathbf{4},\ 2\cdot\mathbf{2}$ $\mathbf{8},\ 3\cdot\mathbf{4},\ \mathbf{2}$ $\mathbf{8},\ 3\cdot\mathbf{4},\ -$	$\mathbf{8},\ 3\cdot\mathbf{4},\ 3\cdot\mathbf{4}_\leftrightarrow,\ -_\leftrightarrow$ $\mathbf{8},\ 3\cdot\mathbf{4},\ 3\cdot\mathbf{4}_\leftrightarrow,\ \mathbf{2}_\leftrightarrow$ $\mathbf{8},\ 3\cdot\mathbf{4},\ 3\cdot\mathbf{4}_\leftrightarrow,\ 2\cdot\mathbf{2}_\leftrightarrow$ $\mathbf{8},\ 3\cdot\mathbf{4},\ 3\cdot\mathbf{4}_\leftrightarrow,\ 3\cdot\mathbf{2}_\leftrightarrow$

Table 8.1: Part 1 of the list of all possible types of lengths of orbits in Theorem 8.2.10.

Action	Σ	$\|\Sigma\|$	Point-orbits	Vertex-orbits	Edge-orbits
532	$\mathbb{A}_5$	60	$\mathbf{60}^\infty, \mathbf{30}^1, \mathbf{20}^1, \mathbf{12}^1$	$\mathbf{60}, \mathbf{30}, \mathbf{20}, \mathbf{12}$ $\mathbf{60}, -, \mathbf{20}, \mathbf{12}$	$\mathbf{60}, -_\leftrightarrow$ $\mathbf{60}, \mathbf{30}_\leftrightarrow$
432	$\mathbb{S}_4$	24	$\mathbf{24}^\infty, \mathbf{12}^1, \mathbf{8}^1, \mathbf{6}^1$	$\mathbf{24}, \mathbf{12}, \mathbf{8}, \mathbf{6}$ $\mathbf{24}, -, \mathbf{8}, \mathbf{6}$	$\mathbf{24}, -_\leftrightarrow$ $\mathbf{24}, \mathbf{12}_\leftrightarrow$
332	$\mathbb{A}_4$	12	$\mathbf{12}^\infty, \mathbf{6}^1, \mathbf{4}^2$	$\mathbf{12}, \mathbf{6}, \mathbf{4}$ $\mathbf{12}, \mathbf{6}, \mathbf{4}$	$\mathbf{12}, -_\leftrightarrow$ $\mathbf{12}, \mathbf{6}_\leftrightarrow$
22n	$\mathbb{D}_n$, $n \geq 3$, odd	$2n$	$(\mathbf{2n})^\infty, \mathbf{n}^2, \mathbf{2}^1$	$\mathbf{2n}, \mathbf{n}, \mathbf{2}$	$\mathbf{2n}_\leftrightarrow, \mathbf{n}_\leftrightarrow$
22n	$\mathbb{D}_n$, $n \geq 4$, even	$2n$	$(\mathbf{2n})^\infty, 2 \cdot \mathbf{n}^1, \mathbf{2}^1$	$\mathbf{2n}, 2 \cdot \mathbf{n}, \mathbf{2}$ $\mathbf{2n}, \mathbf{n}, \mathbf{2}$ $\mathbf{2n}, -, \mathbf{2}$	$\mathbf{2n}, -_\leftrightarrow$ $\mathbf{2n}, \mathbf{n}_\leftrightarrow$ $\mathbf{2n}, 2 \cdot \mathbf{n}_\leftrightarrow$
222	$\mathbb{D}_2 = \mathbb{C}_2^2$	4	$\mathbf{4}^\infty, 3 \cdot \mathbf{2}^1$	$\mathbf{4}, 3 \cdot \mathbf{2}$ $\mathbf{4}, 2 \cdot \mathbf{2}$ $\mathbf{4}, \mathbf{2}$ $\mathbf{4}, -$	$\mathbf{4}, -_\leftrightarrow$ $\mathbf{4}, \mathbf{2}_\leftrightarrow$ $\mathbf{4}, 2 \cdot \mathbf{2}_\leftrightarrow$ $\mathbf{4}, 3 \cdot \mathbf{2}_\leftrightarrow$

Table 8.2: Part 2 of the list of all possible types of lengths of orbits in Theorem 8.2.10.

Action	Σ	$\lvert\Sigma\rvert$	Point-orbits	Vertex-orbits	Edge-orbits
3*2	$\mathbb{A}_4 \times \mathbb{C}_2$	24	$\mathbf{24}^\infty, \mathbf{12}^\infty, \mathbf{8}^1, \mathbf{6}^1$	$\mathbf{24}, \mathbf{12}, \mathbf{8}, \mathbf{6}$ $\mathbf{24}, \mathbf{12}, \mathbf{8}, -$	$\mathbf{24}, \mathbf{12}, \mathbf{12}_\leftrightarrow, -_\leftrightarrow$ $\mathbf{24}, \mathbf{12}, \mathbf{12}_\leftrightarrow, \mathbf{6}_\leftrightarrow$
2*n	$\mathbb{D}_{2n}$, $n \geq 3$	$4n$	$(\mathbf{4n})^\infty, (\mathbf{2n})^\infty, (\mathbf{2n})^1, \mathbf{2}^1$	$\mathbf{4n}, 2 \cdot \mathbf{2n}, \mathbf{2}$ $\mathbf{4n}, \mathbf{2n}, \mathbf{2}$	$\mathbf{4n}, \mathbf{2n}, \mathbf{2n}_\leftrightarrow$ $\mathbf{4n}, \mathbf{2n}, 2 \cdot \mathbf{2n}_\leftrightarrow$
2*2	$\mathbb{D}_4$	8	$\mathbf{8}^\infty, \mathbf{4}^\infty, \mathbf{4}^1, \mathbf{2}^1$	$\mathbf{8}, 2 \cdot \mathbf{4}, \mathbf{2}$ $\mathbf{8}, \mathbf{4}, \mathbf{2}$ $\mathbf{8}, 2 \cdot \mathbf{4}, -$ $\mathbf{8}, \mathbf{4}, -$	$\mathbf{8}, \mathbf{4}, \mathbf{4}_\leftrightarrow, -_\leftrightarrow$ $\mathbf{8}, \mathbf{4}, 2 \cdot \mathbf{4}_\leftrightarrow, -_\leftrightarrow$ $\mathbf{8}, \mathbf{4}, \mathbf{4}_\leftrightarrow, \mathbf{2}_\leftrightarrow$ $\mathbf{8}, \mathbf{4}, 2 \cdot \mathbf{4}_\leftrightarrow, \mathbf{2}_\leftrightarrow$
*nn	$\mathbb{D}_n$, $n \geq 3$, odd	$2n$	$(\mathbf{2n})^\infty, \mathbf{n}^\infty, \mathbf{1}^2$	$\mathbf{2n}, \mathbf{n}, \mathbf{1}$	$\mathbf{2n}, \mathbf{n}, \mathbf{n}_\leftrightarrow$
*nn	$\mathbb{D}_n$, $n \geq 4$, even	$2n$	$(\mathbf{2n})^\infty, 2 \cdot \mathbf{n}^\infty, \mathbf{1}^2$	$\mathbf{2n}, 2 \cdot \mathbf{n}, \mathbf{1}$	$\mathbf{2n}, 2 \cdot \mathbf{n}, 2 \cdot \mathbf{n}_\leftrightarrow$
*22	$\mathbb{D}_2 = \mathbb{C}_2^2$	4	$\mathbf{4}^\infty, 2 \cdot \mathbf{2}^\infty, \mathbf{1}^2$	$\mathbf{4}, 2 \cdot \mathbf{2}, \mathbf{1}$	$\mathbf{4}, 2 \cdot \mathbf{2}, 2 \cdot \mathbf{2}_\leftrightarrow, \mathbf{1}_\leftrightarrow$
nn	$\mathbb{C}_n$, $n \geq 3$	n	$\mathbf{n}^\infty, \mathbf{1}^2$	$\mathbf{n}, \mathbf{1}$	$\mathbf{n}$
22	$\mathbb{C}_2$	n	$\mathbf{2}^\infty, \mathbf{1}^2$	$\mathbf{2}, \mathbf{1}$	$\mathbf{2}, \mathbf{1}_\leftrightarrow$
nx	$\mathbb{C}_{2n}$, $n \geq 3$	$2n$	$(\mathbf{2n})^\infty, \mathbf{2}^1$	$\mathbf{2n}, \mathbf{2}$	$\mathbf{2n}$
2x	$\mathbb{C}_4$	4	$\mathbf{4}^\infty, \mathbf{2}^1$	$\mathbf{4}, \mathbf{2}$ $\mathbf{4}, -$	$\mathbf{4}, -_\leftrightarrow$ $\mathbf{4}, \mathbf{2}_\leftrightarrow$
n*	$\mathbb{C}_n \times \mathbb{C}_2$, $n \geq 3$	$2n$	$(\mathbf{2n})^\infty, \mathbf{n}^\infty, \mathbf{2}^1$	$\mathbf{2n}, \mathbf{n}, \mathbf{2}$	$\mathbf{2n}, \mathbf{n}, \mathbf{n}_\leftrightarrow$
2*	$\mathbb{C}_2 \times \mathbb{C}_2$	4	$\mathbf{4}^\infty, \mathbf{2}^\infty, \mathbf{2}^1$	$\mathbf{4}, 2 \cdot \mathbf{2}$ $\mathbf{4}, \mathbf{2}$	$\mathbf{4}, \mathbf{2}, \mathbf{2}_\leftrightarrow$ $\mathbf{4}, \mathbf{2}, 2 \cdot \mathbf{2}_\leftrightarrow$

Table 8.3: Part 3 of the list of all possible types of lengths of orbits in Theorem 8.2.10.

8.3 Applications of Jordan-like Characterization

In this section, we apply the Jordan-like characterization of Theorems 8.2.1 and 8.2.10 to describe automorphism groups of several important subclasses of planar graphs. First, we determine possible atoms, and primitive graphs and their automorphism groups. Then we determine possible stabilizers similarly as in the proof of Theorem 8.2.1, however, only some of the group products appear. Lastly, we combine these stabilizers together with spherical groups which are representable by primitive graphs, again only some of the (possibly restricted) cases of Table 8.1 may happen. In what follows, for a subclass $\mathcal{C}$ of planar graphs, we set

$$\mathrm{Fix}(\mathcal{C}) = \Big\{\mathrm{Fix}(\partial A^*) : A \text{ is an atom of the reduction tree of a graph in } \mathcal{C}\Big\}.$$

For instance, consider the class of all trees (TREE). The only primitive graph is K_1 (with a single pendant edge attached), so its automorphism group is trivial. All atoms are block atoms (either star block atoms or K_2 with a single pendant edge attached). Therefore, the stabilizers are determined by Lemma 8.2.3. The class is closed under the direct product and the wreath product with symmetric groups. Since K_1 has the trivial automorphism group, we get that the automorphism groups of trees are same as are the vertex-stabilizers of trees, so we get the Jordan's characterization; see Theorem 6.3.1.

8.3.1 Automorphism Groups of 2-connected Planar Graphs

Denote the class of 2-connected planar graphs by 2-connected PLANAR. Consider the reduction tree of a 2-connected planar graph. There are no block atoms since all the atoms are proper or dipoles. Note that Fix(2-connected PLANAR) consists of all point-wise stabilizers of edges in 2-connected planar graphs. The reason is that for a proper atom/dipole A with $\partial A = \{u, v\}$, we may consider the extended atom A^+ constructed from A by adding the edge uv, and the corresponding expanded extended atom $(A^+)^*$. Then $\mathrm{Fix}(\partial A^*)$ is the point-wise stabilizer of the edge uv in $\mathrm{Aut}_{\partial(A^+)^*}((A^+)^*)$.

Lemma 8.3.1. *The class* Fix(2-connected PLANAR) *is defined inductively as follows:*

(a) $\{1\} \in$ 2-Fix(2-connected PLANAR).

(b) If $\Psi_1, \Psi_2 \in$ Fix(2-connected PLANAR)*, then*

$$\Psi_1 \times \Psi_2 \in \mathrm{Fix}(\text{2-connected PLANAR}).$$

(c) If $\Psi \in$ Fix(2-connected PLANAR)*, then*

$$\Psi \wr \mathbb{S}_n \in \mathrm{Fix}(\text{2-connected PLANAR}).$$

(d) If $\Psi_1, \Psi_2, \Psi_3 \in$ Fix(2-connected PLANAR)*, then*

$$(\Psi_1^2 \times \Psi_2) \rtimes \mathbb{C}_2 \in \mathrm{Fix}(\text{2-connected PLANAR}).$$

Proof. The constructions are explained in Fig. 8.2b, c, d. For the other implication, we argue exactly as in the proof of Theorem 8.2.1. We apply induction according to the depth of the reduction tree. Let A be an atom with colored edges corresponding to proper atoms/dipoles $\hat{A}$. We assume that $\mathrm{Fix}(\partial\hat{A}^*)$ can be constructed using the operations (a)–(d). Since A is a proper atom or a dipole, only Lemmas 8.2.3 and 8.2.4 apply, so the operations (b), (c), (d) are sufficient. For (d), we use the simplification described below the proof of Lemma 8.2.4. □

Notice that the operations (c) and (d) are restrictions of (c) and (d) from Theorem 8.2.1 to $\mathbb{S}_n$ and $\mathbb{D}_1 \cong \mathbb{C}_2$, respectively. Also, the class $\mathrm{Fix}(\textsf{2-connected PLANAR})$ is more rich than the class $\mathrm{Aut}(\textsf{TREE})$, characterized by Jordan (Theorem 6.3.1) employing the operations (a)–(c). Therefore,

$$\mathrm{Aut}(\textsf{TREE}) \subsetneq \mathrm{Fix}(\textsf{2-connected PLANAR}) \subsetneq \mathrm{Fix}(\textsf{connected PLANAR}).$$

Finally, we deal with a primitive graph in the root of the reduction tree. We easily modify the characterization in Theorem 8.2.10. There are two key differences. First, we use the class $\mathrm{Fix}(\textsf{2-connected PLANAR})$ instead of $\mathrm{Fix}(\textsf{connected PLANAR})$. Second, we only consider edge-orbits since there are no single pendant edges in primitive graphs, i.e., no expanded block atoms attached to their vertices.

Theorem 8.3.2. *The class* $\mathrm{Aut}(\textsf{2-connected PLANAR})$ *consists of the following groups. Let H be a planar graph with colored (possibly oriented) edges, which is either 3-connected, or K_2, or a cycle C_n. Let $m_1,\dots,m_\ell$ be the sizes of the edge-orbits of the action of* $\mathrm{Aut}(H)$*. Then for all choices* $\Psi_1,\dots,\Psi_\ell \in \mathrm{Fix}(\textsf{2-connected PLANAR})$*, we have*

$$(\Psi_1^{m_1} \times \cdots \times \Psi_\ell^{m_\ell}) \rtimes \mathrm{Aut}(H) \in \mathrm{Aut}(\textsf{2-connected PLANAR}),$$

where $\mathrm{Aut}(H)$ *permutes the factors of* $\Psi_1^{m_1} \times \cdots \times \Psi_\ell^{m_\ell}$ *following the action on the edges of H.*

On the other hand, every group of $\mathrm{Aut}(\textsf{2-connected PLANAR})$ *can be constructed in the above way as*

$$(\Psi_1^{m_1} \times \cdots \times \Psi_\ell^{m_\ell}) \rtimes \Sigma,$$

where $\Psi_1,\dots,\Psi_\ell \in \mathrm{Fix}(\textsf{2-connected PLANAR})$ *and Σ is a spherical group.*

Proof. The reduction tree of a 2-connected planar graph contains only proper atoms and dipoles, and the primitive graph cannot be K_1. The proof proceeds as in Theorem 8.2.10, the only difference is that we use Lemma 8.3.1 instead of Theorem 8.2.1. □

8.3.2 Automorphism Groups of Outerplanar Graphs

Let G be a connected outerplanar graph with the reduction series $G = G_0,\dots,G_r$. All graphs G_i are outerplanar. Since no 3-connected planar graph is outerplanar, G_r is by Lemma 7.4.2 either K_1, K_2, or a cycle C_n (possibly with a single pendant edge attached). So, $\mathrm{Aut}(G_r)$ is a subgroup of a dihedral group.

Next, we describe possible atoms encountered in the reduction:

- *Star block atoms.* We have arbitrary star block atoms.
- *Non-star block atoms.* Each non-star block atom A is outerplanar. By Lemma 7.4.3, A is either K_2 or C_n with single pendant edges attached. Therefore, $\mathrm{Fix}(\partial A)$ is a subgroup of $\mathbb{C}_2$.
- *Proper atoms.* For a proper atom A with $\partial A = \{u, v\}$, the extended proper atom A^+ is an outerplanar graph having an embedding with the edge uv in the outer face. Therefore, by Lemma 7.4.4, A is a non-trivial path, so $\mathrm{Fix}(\partial A) \cong \mathbb{C}_1$.
- *Dipoles.* For a dipole A with $\partial A = \{u, v\}$, the extended dipole A^+ (with the edge uv) has an embedding such that the edge uv belong to the outer face. Therefore, assuming that G contains no parallel edges, A consists of exactly two edges, one corresponding to an edge of G, and the other to a proper atom. Again, $\mathrm{Fix}(\partial A) \cong \mathbb{C}_1$.

Lemma 8.3.3. $\mathrm{Fix}(\textsf{connected OUTERPLANAR}) = \mathrm{Aut}(\textsf{TREE})$.

Proof. By induction, when A is a proper atom or a dipole, we get that $\mathrm{Fix}(\partial A^*)$ is the direct product of $\mathrm{Fix}(\partial \hat{A}^*)$ of the attached extended block atoms. Alternatively, it can be argued that each 2-connected outerplanar graph G has $\mathrm{Aut}(G)$ a subgroup of $\mathbb{D}_n$. Unless G is a cycle, $\mathrm{Aut}(G)$ stabilizes the outer face.

We use the same approach as in the proof of Theorem 8.2.1. Only Lemmas 8.2.3, 7.4.4 for $\mathbb{C}_1$, and 8.2.5 for subgroups of $\mathbb{C}_2$ apply. For a block atom A with $\mathrm{Fix}(\partial A) \cong \mathbb{C}_2$, there might be either an edge-orbit of type **1** consisting of a pendant edge corresponding to a block-atom B, or an edge-orbit of type $\mathbf{1}_{\leftrightarrow}$ consisting of an edge corresponding to a proper atom or a dipole C. In the latter case, $\mathrm{Fix}(\partial C^*)$ is just the direct product of extended block atoms attached in C^*, so the reflection in $\mathrm{Fix}(\partial A)$ just swaps them, while at most one is fixed. Therefore, $\mathrm{Fix}(\textsf{connected OUTERPLANAR})$ is defined inductively by the operations (a)–(c) from Theorem 6.3.1. □

By adapting the proof of Theorem 8.2.10, we get the following:

Theorem 8.3.4. *The class* $\mathrm{Aut}(\textsf{connected OUTERPLANAR})$ *consists of the following groups:*

(i) *If* $\Psi \in \mathrm{Fix}(\textsf{connected OUTERPLANAR})$, *then*

$$\Psi \wr \mathbb{C}_n \in \mathrm{Aut}(\textsf{connected OUTERPLANAR}).$$

(ii) *If* $\Psi_1, \Psi_2 \in \mathrm{Fix}(\textsf{connected OUTERPLANAR})$, *then*

$$(\Psi_1^{2n} \times \Psi_2^n) \rtimes \mathbb{D}_n \in \mathrm{Aut}(\textsf{connected OUTERPLANAR}), \qquad \forall n \text{ odd.}$$

(iii) *If* $\Psi_1, \Psi_2, \Psi_3 \in \mathrm{Fix}(\textsf{connected OUTERPLANAR})$, *then*

$$(\Psi_1^{2n} \times \Psi_2^n \times \Psi_3^n) \rtimes \mathbb{D}_n \in \mathrm{Aut}(\textsf{connected OUTERPLANAR}), \qquad \forall n \text{ even.}$$

Moreover, $\mathrm{Aut}(\textsf{connected OUTERPLANAR}) = \mathrm{Aut}(\textsf{PSEUDOTREE})$.

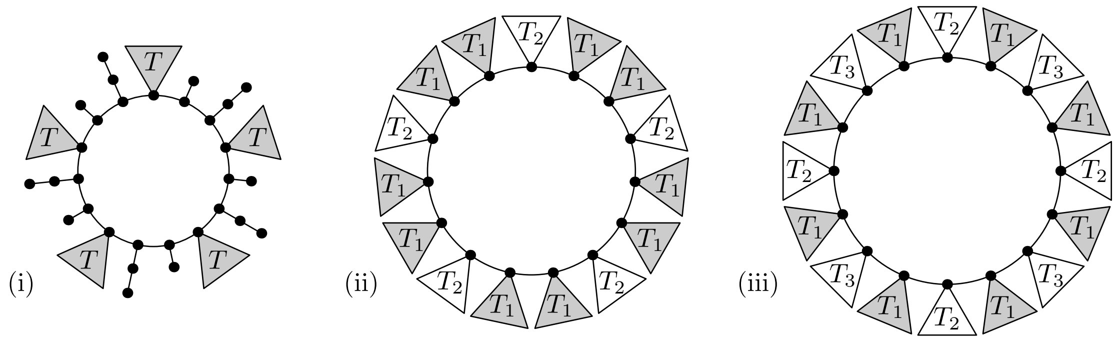


Figure 8.9: Construction of the automorphism groups from Theorem 8.3.4, for $\mathrm{Fix}(\partial T) \cong \Psi$ and $\mathrm{Fix}(\partial T_i) \cong \Psi_i$.

Proof. We have PSEUDOTREE $\subsetneq$ connected OUTERPLANAR and all these automorphism groups are already realized by pseudotrees, see Fig. 8.9. Therefore, we get that $\mathrm{Aut}(\text{connected OUTERPLANAR}) = \mathrm{Aut}(\text{PSEUDOTREE})$.

For the other direction, consider the primitive graph G_r associated to G. We assume that G_r is a cycle, otherwise it is trivial. We get three cases leading to different automorphism groups from the statement: (i) $\mathrm{Aut}(G_r) \cong \mathbb{C}_n$, for $n \neq 2$, (ii) $\mathrm{Aut}(G_r) \cong \mathbb{D}_n$, for n odd, (iii) $\mathrm{Aut}(G_r) \cong \mathbb{D}_n$, for n even.

In the case (i), we have only edge-orbits of type $\boldsymbol{n}$ corresponding to block atoms $A_1, \dots, A_\ell$, so $\Psi = \mathrm{Fix}(\partial A_1^*) \times \cdots \times \mathrm{Fix}(\partial A_\ell^*)$ and $\mathrm{Aut}(G_r) \cong \Psi \wr \mathbb{C}_n$.

In the case (ii), we have edge-orbits of types $\boldsymbol{2n}$, $\boldsymbol{n}$, $\boldsymbol{n}_\leftrightarrow$, each forming one equivariant class of orbits; see Lemma 8.2.6. We define Ψ_1, Ψ_2, and Ψ_3 exactly as in the proof of Lemma 8.2.7. However, since $\mathrm{Fix}(\partial A^*)$ of a proper atom or a dipole A is the direct product of $\mathrm{Fix}(\partial \hat{A}^*)$ of the attached extended block atoms $\hat{A}^*$, we can place these factors into Ψ_1 and Ψ_2. Therefore, $\mathrm{Aut}(G_r) \cong (\Psi_1^{2n} \times \Psi_2^n) \rtimes \mathbb{D}_n$.

In the case (iii), the argument is similar as in (ii), but we have two equivariance classes of edge-orbits of type $\boldsymbol{n}$ and of type $\boldsymbol{n}_\leftrightarrow$. Thus we adapt the proof of Lemma 8.2.8 instead. □

We get $\mathrm{Aut}(\text{OUTERPLANAR}) = \mathrm{Aut}(\text{PSEUDOFOREST})$, and their characterization follows from from Theorem 7.2.1,

8.3.3 Automorphism Groups of Series-Parallel Graphs

Let G be a connected series-parallel graph with the reduction series $G = G_0, \dots, G_r$. All graphs the graphs G_i remain series-parallel since each 1-cut and 2-cut is introduced in the composition of the graph using one or the other operation. The only exception is G_r, where we allow $G_r = K_1$. Since no 3-connected planar graph is series-parallel, G_r is by Lemma 7.4.2 again either K_1, K_2, or a cycle C_n, with attached single pendant edges. So, $\mathrm{Aut}(G_r)$ is a subgroup of a dihedral group.

Next, we describe possible atoms encountered in the reduction:

- *Star block atoms.* Star block atoms may be arbitrary.

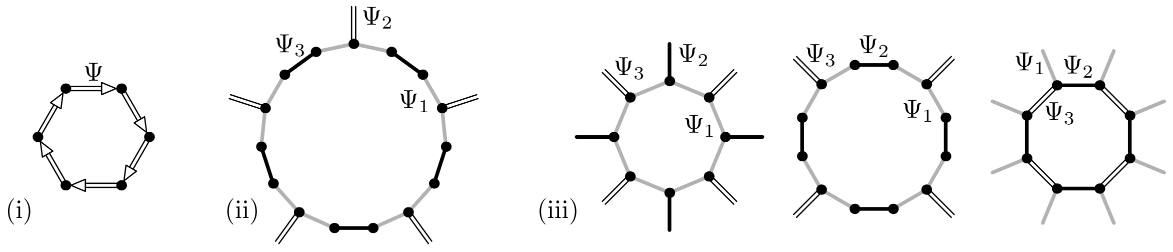


Figure 8.10: Construction of the automorphism groups from Theorem 8.3.6. We get three possible combinations of edge-orbits in (iii), leading to different semidirect products with $\mathbb{D}_n$.

- *Non-star block atoms.* Each non-star block atom A is a series-parallel graph. By Lemma 7.4.3, we get that A is either K_2 or a cycle with attached single pendant edges, so $\mathrm{Fix}(\partial A)$ is again a subgroup of $\mathbb{C}_2$.
- *Proper atoms.* For a proper atom A, the extended proper atom A^+ is a series-parallel graph. Therefore, by Lemma 7.4.4, A is a path, so $\mathrm{Fix}(\partial A) \cong \mathbb{C}_1$.
- *Dipoles.* Dipoles may be arbitrary.

Lemma 8.3.5. Fix(connected SERIES-PARALLEL) = Fix(2-connected PLANAR).

Proof. We use the same approach as in the proof of Theorem 8.2.1. Since the groups $\mathrm{Fix}(\partial A)$ of encountered atoms A are restricted, by Lemmas 8.2.3, 8.2.4, and 8.2.5, the groups $\mathrm{Fix}(\partial A^*)$ can be constructed using (a)–(d) of Lemma 8.3.1. We note that each non-star block atom A with $\mathrm{Fix}(\partial A) \cong \mathbb{C}_2$ has at most one edge-orbit of size 1, which is either of type **1**, or of type $\mathbf{1}_{\leftrightarrow}$. □

By adapting the proof of Theorem 8.2.10, we get the following:

Theorem 8.3.6. *The class* Aut(connected SERIES-PARALLEL) *consists of the following groups:*

(i) If $\Psi \in$ Fix(connected SERIES-PARALLEL)*, then*

$$\Psi \wr \mathbb{C}_n \in \text{Aut(connected SERIES-PARALLEL)}.$$

(ii) If $\Psi_1, \Psi_2, \Psi_3 \in$ Fix(connected SERIES-PARALLEL)*, then*

$$(\Psi_1^{2n} \times \Psi_2^n \times \Psi_3^n) \rtimes \mathbb{D}_n \in \text{Aut(connected SERIES-PARALLEL)}, \qquad \forall n \textit{ odd.}$$

(iii) If $\Psi_1, \Psi_2, \Psi_3 \in$ Fix(connected SERIES-PARALLEL)*, then*

$$(\Psi_1^{2n} \times \Psi_2^n \times \Psi_3^n) \rtimes \mathbb{D}_n \in \text{Aut(connected SERIES-PARALLEL)}, \qquad \forall n \textit{ even.}$$

We note that the semidirect products in (ii) and (iii) are different, see the proof for details.

Proof. Fig. 8.10 depicts the constructions. For the other direction, we deal with the case that the primitive graph G_r is a cycle C_k with attached single pendant edges, otherwise it is trivial. As in the proof of Theorem 8.3.4, we get three cases leading to different automorphism groups from the statement: (i) $\mathrm{Aut}(G_r) \cong \mathbb{C}_n$, for $n \neq 2$, (ii) $\mathrm{Aut}(G_r) \cong \mathbb{D}_n$, for n odd, (iii) $\mathrm{Aut}(G_r) \cong \mathbb{D}_n$, for n even.

The case (i) is exactly the same as in Theorem 8.3.4. In the cases (ii) and (iii), we have edge-orbits of types $\boldsymbol{2n}$, $\boldsymbol{n}$, and $\boldsymbol{n}_{\leftrightarrow}$. The group $\mathrm{Aut}(G_r)$ acts on the edge-orbits of type $\boldsymbol{2n}$ regularly, exactly as in the proofs of Lemmas 8.2.7 and 8.2.8. We argue that the edge-orbits of types $\boldsymbol{n}$ and $\boldsymbol{n}_{\leftrightarrow}$ are more restricted.

In the case (ii), we have exactly one conjugacy class of reflections f_i, recall the notation from the proof of Lemma 8.2.6. Each of the reflections either stabilizes two edges of the cycle or two vertices if n is even; or a vertex and an edge if n is odd. In the first case, we get two equivariant edge-orbits of type $\boldsymbol{n}_{\leftrightarrow}$. In the second case, we get at most two equivariant edge-orbits of type $\boldsymbol{n}$ consisting of pendant-edges attached to stabilized vertices. In the last case, we get an edge-orbit of type $\boldsymbol{n}_{\leftrightarrow}$ and at most one edge-orbit of type $\boldsymbol{n}$ consisting of pendant-edges. The group $\mathrm{Aut}(G)$ can be constructed by the operation (ii) in the same way as in the case (d) in the proof of Lemma 8.2.7.

In the case (iii), we have two conjugacy classes of reflections. Each reflection stabilizes either two edges of the cycle, or two vertices, belonging to the same orbit; and this is same for each conjugacy class. Therefore, each conjugacy class defines either an edge-orbit of type $\boldsymbol{n}$ of attached single pendant edges, or an edge-orbit of type $\boldsymbol{n}_{\leftrightarrow}$, non-equivariant to the edge-orbit defined by the other class. In total, we get three possibilities: two non-equivariant orbits of type $\boldsymbol{n}$, one edge-orbit of type $\boldsymbol{n}$ and one edge-orbit of type $\boldsymbol{n}_{\leftrightarrow}$, or two non-equivariant edge-orbits of type $\boldsymbol{n}_{\leftrightarrow}$. These three possibilities lead to different semidirect products in (iii) which are created by restrictions of the operations (e) from the proof of Lemma 8.2.8. □

8.4 Comparison with Babai's Characterization

In this section, we compare our characterization of automorphism groups of planar graphs with Babai's characterization [11].

Statement of Babai's Characterization. We include the full statement of Babai's characterization of Aut(PLANAR), copied from [11] with an adapted notation.

Theorem 8.4.1 (Babai [11], 8.12 The Main Corollary)**.** *Let Ψ be a finite group. All graphs below are assumed to be finite.*

(A) Ψ is representable by a planar graph if and only if

$$\Psi \cong \Psi_1 \wr \mathbb{S}_{n_1} \times \cdots \times \Psi_t \wr \mathbb{S}_{n_t} \tag{8.2}$$

for some t, $n_1, \dots, n_t$ where the groups Ψ_i are representable by connected planar graphs.

(A') Ψ is representable by a planar graph with a fixed point if and only if Ψ is representable by a planar graph.

(A”) *Ψ is representable by a planar fixed-point free graph if and only if a (8.2) decomposition exists with all groups Ψ_i possessing planar connected fixed-point-free graph representation.*

(B) *For $|\Psi| \geq 3$, Ψ is representable by a connected planar graph if and only if*

$$\Psi \cong (\Psi_1 \wr \mathbb{S}_k) \wr (\Psi_2|_{\mathcal{K}}) \tag{8.3}$$

for some positive integer k, where Ψ_1 should be representable by a connected graph having a fixed point; and either Ψ_2 is representable by a 2-connected planar graph G_2, or $k \geq 2$ and $|\Psi_2| = 1$. In the former case, $\Psi_2|_{\mathcal{K}}$ denotes the not necessarily effective permutation group, acting on some orbit $\mathcal{K}$ of $\mathrm{Aut}(G_2)$.

(B’) *Ψ is representable by a connected planar graph having a fixed point if and only if a (8.3) decomposition exists as described under (B) with either $|\Psi_2| = 1$ or G_2 a 2-connected planar graph with a fixed point.*

(B”) *Ψ is representable by a connected fixed-point-free planar graph if and only if a (8.3) decomposition exists with $|\Psi_2| \neq 1$, G_2 fixed-point free (hence $|\mathcal{K}| \geq 2$).*

(C) *Ψ is representable by a 2-connected planar graph if and only if*

$$\Psi \cong (\Psi_1 \wr \mathbb{S}_k) \wr (\Psi_2|_{\mathcal{K}}) \tag{8.4}$$

where $|\Psi_1| \leq 2$; Ψ_2 is representable by some 2-connected planar graph G_2. If $|\Psi_1| = k = 1$, G_2 should be 3-connected. $\Psi_2|_{\mathcal{K}}$ denotes the action of $\Psi_2 \cong \mathrm{Aut}(G_2)$, as a not necessarily effective permutation group, on $\mathcal{K}$, an orbit of either an ordered pair (a, b) or of an unordered pair $\{a, b\}$ of adjacent vertices $a, b \in \boldsymbol{V}(G_2)$.

(C’) *Ψ is representable by a 2-connected planar graph with a fixed point or with an invariant edge if and only if a (8.4) decomposition exists such that G_2 has a fixed point or an invariant edge.*

(C”) *Ψ is representable by a 2-connected planar fixed-point-free graph if and only if a (8.4) decomposition exists with G_2 fixed-point-free.*

(D) *Ψ is representable by a 3-connected planar graph if and only if Ψ is isomorphic to one of the finite symmetry groups of the 3-space:*

$$\begin{array}{lllll} \mathbb{C}_n, & \mathbb{D}_n, & \mathbb{A}_4, & \mathbb{S}_4, & \mathbb{A}_5, \\ \mathbb{C}_n \times \mathbb{C}_2, & \mathbb{D}_n \times \mathbb{C}_2, & \mathbb{A}_4 \times \mathbb{C}_2, & \mathbb{S}_4 \times \mathbb{C}_2, & \mathbb{A}_5 \times \mathbb{C}_2. \end{array} \tag{8.5}$$

(D’) *Ψ is representable by a 3-connected planar graph with a fixed point if and only if Ψ is a cyclic or a dihedral group.*

(D”) *Ψ is representable by a 3-connected planar fixed-point-free graph if and only if $|\Psi| \geq 2$ and Ψ is one of the groups listed under (8.5).*

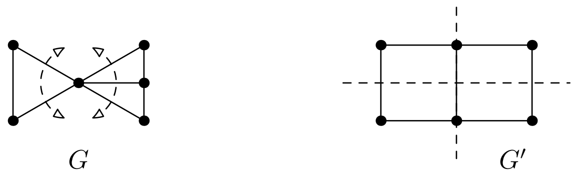


Figure 8.11: Planar graphs G and G' with $\mathrm{Aut}(G) \cong \mathrm{Aut}(G') \cong \mathbb{C}_2^2$, having different actions. For G, we get independent reflections/rotations for each of the blocks. For G', the group $\mathrm{Aut}(G')$ is generated by two reflections.

The characterization is very long and hard to understand, but it works in a nutshell as follows. The automorphism group of a k-connected planar graph ($k \le 2$) is constructed by combining automorphism groups of smaller k-connected planar graphs with stabilizers of k-connected planar graphs and automorphism groups of $(k+1)$-connected graphs.

The part (A) corresponds to Jordan's Theorem 7.2.1. The automorphism groups listed in the part (D) are the spherical groups described in Section 8.1 and are based on the classical results from geometry. Therefore, the novel parts are (B) and (C). Unfortunately, it is not clear which groups are $\Psi_2|_{\mathcal{K}}$, used in (8.3) and (8.4).

In principle, it would be possible to derive Theorem 8.4.1 from the described Jordan-like characterization of Theorems 8.2.1 and 8.2.10, and the reader can work out further details. The opposite is not possible because Jordan-like characterization contains more information about automorphism groups of planar graphs; for instance the one given in Table 8.1. We note that Jordan-like characterizations for all parts of Theorem 8.4.1 exist. For example, the characterization of Aut(2-connected PLANAR) in Section 8.3.1, corresponds to the part (C) of Theorem 8.4.1.

Group Products Instead of Group Extensions. We explain why the simple version of our characterization given in Theorem 6.3.8 already describes the structure more accurately than Theorem 6.3.7.

Recall simple groups from Section 7.2. A consequence of Babai's characterization (Theorem 6.3.7) describes building blocks (simple groups) for the automorphism groups of planar graphs. But Theorem 6.3.7 does not describe how the building blocks for automorphism groups of planar graphs are put together. In certain special cases, the structure of Ψ can be described from Σ and Ψ/Σ by semidirect products; see Section 7.2.

A simple version of our characterization (Theorem 6.3.8) states that the automorphism groups of planar graphs can be build from standard building blocks using a series of semidirect product, so it describes the structure more accurately than Theorem 6.3.7. In Theorems 8.2.1 and 8.2.10, we describe these semidirect products in more detail. As far as we understand Babai's approach, he uses generalized wreath products instead of semidirect products. More important difference between ours and Babai's approach consists in the fact that we deal with the 1- and 2-connected case together. This allows us to apply recursion in a more compact way, thus deriving the Jordan-like characterization of stabilizers of 1-cuts and 2-cuts, established in Theorem 8.2.1.

A group of symmetries of a graph is not fully described just by characterizing it as an abstract group. Figure 8.11 shows two simple graphs with isomorphic abstract automorphism groups, realized by different group actions on the graphs. From Babai's characterization, the structure of this action is not very clear. On the other hand, Theorems 8.2.1 and 8.2.10 reveal the actions of automorphism groups on planar graphs, by describing them with respect to each 1-cut and 2-cut (Theorem 8.2.1), and with respect to the primitive graph (Theorem 8.2.10).

8.5 Quadratic-time Algorithm

In this section, we describe a quadratic-time algorithm which computes the automorphism groups of planar graphs.

Lemma 8.5.1. *Let G be an essentially 3-connected planar graph with colored edges. There exists a quadratic-time algorithm which computes a generating set of* $\mathrm{Aut}(G)$ *or of the stabilizer of a vertex.*

Proof. Consider the unique embedding of G into the sphere. Let n be the number of vertices of G. We work with colored pendant edges as with colored vertices. Let (v,e,e') be an arbitrary angle. For every other angle $(\hat v,\hat e,\hat e')$ there exists at most one automorphism $\pi \in \mathrm{Aut}(M)$ which maps (v,e,e') to $(\hat v,\hat e,\hat e')$. We introduce three involutions ρ,λ,τ on the set of oriented angles by setting:

- $\rho(v,e,e') = (v,e',e)$,
- $\lambda(v,e,e') = (v',e'',e')$, where v' is the other vertex incident to e' and the angles (v,e,e') and (v',e'',e') lie on the same side of e', and
- $\tau(v,e,e') = (v,e,e'')$, where (v,e,e'') is the other angle incident to v and e.

The procedure checking whether the mapping $\pi\colon (v,e,e') \mapsto (\hat v,\hat e,\hat e')$ extends to an automorphism is based on the observation that an automorphism of G commutes with ρ, λ, and τ. It follows that in time $\mathcal{O}(n)$, we can check whether the mapping $\pi\colon (v,e,e') \mapsto (\hat v,\hat e,\hat e')$ extends to an automorphism or not. In the positive case, we get the automorphism π as a byproduct. Moreover, we can easily verify whether the automorphism preserves colors. Therefore, $\mathrm{Aut}(G)$ is computed in time $\mathcal{O}(n^2)$ and the algorithm can easily identify which of the abstract spherical groups is isomorphic to $\mathrm{Aut}(G)$. Note that the number of edges of G is $\mathcal{O}(n)$. For the stabilizer, we just compute the automorphisms which map (v,e,e') to $(v,\hat e,\hat e')$. □

Notice that by the above algorithm, we can compute $\mathrm{Fix}(\partial A)$ of a non-star block atom A.

Lemma 8.5.2. *There is a linear-time algorithm which computes a generating set of* $\mathrm{Fix}(\partial A)$ *and* $\mathrm{Aut}_{\partial A}(A)$ *of a proper atom, dipole, or a star block atom A with colored edges.*

Proof. If A is a dipole or a star block atom, we get $\mathrm{Fix}(\partial A)$ as the direct product of symmetric groups, one for each type and color class of edges. Further, if A is a dipole, it is symmetric if and only if it has the same number of directed edges of each color class in both directions, and then $\mathrm{Aut}(A) \cong \mathrm{Fix}(\partial A) \rtimes \mathbb{C}_2$; otherwise $\mathrm{Aut}_{\partial A}(A) = \mathrm{Fix}(\partial A)$.

Let A be a proper atom with $\partial A = \{u, v\}$. Since A^+ is essentially 3-connected, the reasoning from Lemma 8.5.1 applies. We know that both $\mathrm{Aut}_{\partial A}(A)$ and $\mathrm{Fix}(\partial A)$ are generated by automorphisms which maps the two angles containing u and uv and the two angles containing v and uv between each other. We can easily test in time $\mathcal{O}(n)$ which of these mappings are automorphisms. □

Proof of Theorem 6.3.9. By Lemma 7.7.1, we cane compute the simplified reduction tree T in time $\mathcal{O}(n)$. In the beginning, all nodes of T are unmarked. We process the tree from the leaves to the root, dealing with the nodes which have all children marked, and marking these nodes after. We compute colors and symmetry types of the considered atoms, the groups $\mathrm{Fix}(\partial A)$ and for symmetric atoms also involutions τ exchanging the vertices in the boundaries. Let the colors be integers. Suppose that in some step, we process several atoms whose edges are colored and have computed symmetry types.

Dipoles and star block atoms. To each dipole/star block atom with n edges, we assign the vector $\boldsymbol{v} = (t, c_1, \dots, c_m)$ where t is the type of the atom and $c_1, \dots, c_m$ is the sorted list of colors. By lexicographic sorting of these vectors for all dipoles/star block atoms, we can compute isomorphism classes and assign new colors to them. This runs in linear time.

Non-star block atoms. Let A be a non-star block atom with $\partial A = \{u\}$. We work with single pendant edges as with colors of vertices. Let n be the number of its vertices and m the number of its edges. Consider a map of A. For each choice of an angle (u, e, e'), we compute labellings $1, \dots, n$ of the vertices and $1, \dots, m$ of the edges as they appear in BFS of the map. Starting with u, we visit all its neighbors, from the one incident with e following the rotational scheme. From each neighbor, we visit their unvisited neighbors, and so on.

For a labeling, we compute the vector

$$\boldsymbol{v} = \Big(c_1, \dots, c_n, (x_1, y_1, c'_1), \dots, (x_m, y_m, c'_m)\Big)$$

where c_i is the color of the i-th vertex, and $x_j < y_j$ are the endpoints and c'_j the color of the j-th edge. We compute at most $2n$ of these vectors for all possible choices of (u, e, e') and we choose the one which is lexicographically smallest. Notice that two atoms are isomorphic if and only if their associated vectors are identical. So, by sorting the chosen vectors for all non-star block atoms lexicographically, we compute isomorphism classes and assign new colors to them. This runs in quadratic time since we compute $2n$ vectors for each atom.

Proper atoms. We approach A^+ similarly as a non-star block atom, but we just need to consider the labellings starting from an angle containing uv and either u, or v. We have four choices for each vector, so it runs in linear time.

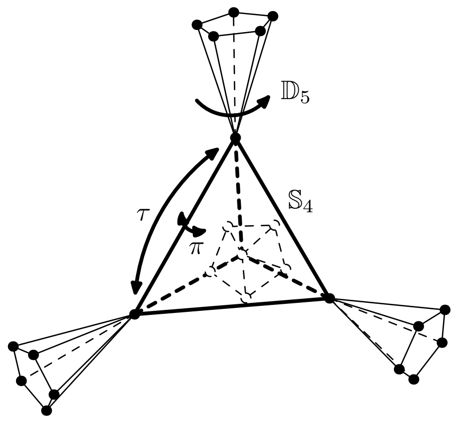


Figure 8.12: The bold edges are symmetric proper atoms A with $\mathrm{Fix}(\partial A) \cong \mathbb{C}_2$, generated by π. We visualize the automorphism group $(\mathbb{D}_5^4 \times \mathbb{C}_2^6) \rtimes \mathbb{S}_4$.

For each considered atom A, we apply one of the algorithms described in Lemmas 8.5.1 and 8.5.2, and we compute $\mathrm{Fix}(\partial A)$. If A is a dipole or a proper atom, we also compute its type, and if A is symmetric, we construct an involution $\tau \in \mathrm{Aut}_{\partial A}(A)$ exchanging the vertices of ∂A.

Following the proof of Theorem 8.2.1, we can compute in quadratic time also $\mathrm{Fix}(\partial A^*)$ and τ^* for every node which is a child of the root node. By Theorem 8.2.10, we can compute $\mathrm{Aut}(G)$ as the semidirect product of these groups $\mathrm{Fix}(\partial A^*)$ with $\mathrm{Aut}(H)$ computed by Lemma 8.5.1. We can output the automorphism groups in terms of permutation generators, or by assigning the computed groups $\mathrm{Fix}(\partial A)$, τ and $\mathrm{Aut}(H)$ to the corresponding nodes of T. □

Corollary 8.5.3. *For a planar graph G, its reduction tree can be computed in quadratic time.*

8.6 Conclusions

Let G be a connected planar graph. We propose the following way of imagining the action of $\mathrm{Aut}(G)$ geometrically, which can be used in a dynamic visualization in 3-space. Suppose that the reduction tree T of G is computed together with the corresponding parts of $\mathrm{Aut}(G)$, assigned to the nodes. For each 2-connected block, we have some 3-connected colored primitive graph which can be visualized by a symmetric polytope, and these polytopes are connected by articulations as in the block tree; see Fig. 8.12.

Onto each polytope, we attach a hierarchical structure of colored atoms given by the decomposition. For a dipole A with $\partial A = \{u, v\}$, we know that independent color classes can be arbitrarily permuted, so we assign symmetric groups to them. For a proper atom A with $\partial A = \{u, v\}$, the non-trivial element of $\mathrm{Fix}(\partial A)$ (if it exists) is the reflection through u and v, and we represent it. These symmetries generate the subgroup of $\mathrm{Aut}(G)$ which fixes all polytopes. Further, for an edge uv representing a symmetric atom A, we also add an involution $\tau \in \mathrm{Aut}_{\partial A}(A)$ exchanging u and v, which is geometrically a reflection through uv in A^+, if τ is used in $\mathrm{Aut}(G)$.

The central block is preserved by $\mathrm{Aut}(G)$, so it is transformed by a spherical group $\mathrm{Aut}(H)$, permuting also the attached polytopes. Multiple polytopes attached at an articulation correspond to a star block atom. So isomorphic subtrees of blocks can be arbitrarily permuted and we assign symmetric groups to them. Consider a polytope attached by the articulation u to its parent in the block tree. Since $\mathrm{Fix}(\partial A)$ of a non-star block atom is either a dihedral or a cyclic group, the polytope can be only rotated/reflected around u.

Problem 8.6.1. *For a planar graph G, is it possible to compute a generating set of* $\mathrm{Aut}(G)$ *in a linear time?*

Our algorithm has two bottlenecks which need to be improved to get a linear-time algorithm: (i) the algorithm of Lemma 8.5.1, (ii) the procedure of finding lexicographically smallest vectors of non-star block atoms. Both of these can likely be solved by modifying the algorithm of [194].

Problem 8.6.2. *What are the automorphism groups of projectively planar or toroidal graphs?*

9 Graph Isomorphism Restricted by Lists

This chapter contains:

- *9.1: Basic Results.* We describe bipartite perfect matchings and give basic algorithms for ListIso.
- *9.2: GI-completeness of GraphIso Implies NP-completeness of ListIso.* We show that reduction for GraphIso using vertex-gadgets can be modified for ListIso.
- *9.3: NP-completeness for 3-regular Colored Graphs.* We modify the reduction of Lubiw [260].
- *9.4, 9.5, and 9.6.* We describe combinatorial algorithms for GraphIso of trees, planar graphs, interval graphs, permutation graphs, and circle graphs, and modify them for ListIso.
- *9.7 and 9.8.* We modify involved algorithms for GraphIso of bounded genus and bounded treewidth graphs to ListIso.
- *9.9: Conclusions.* We describe related topics and open problems.

http://pavel.klavik.cz/orgpad/list_isomorphism.html

9.1 Basic Results

Let ℓ be the total size of all lists. To make the problem non-trivial, we can assume that $\ell \geq n$.

9.1.1 Bipartite Perfect Matchings

As a subroutine, we frequently solve *bipartite perfect matching*:

Lemma 9.1.1 (Hopcroft and Karp [187])**.** *The bipartite perfect matching problem can be solved in time $\mathcal{O}(\sqrt{n}m)$, where n is the number of vertices and m is the number of edges.*

We repeatedly use this subroutine to solve ListIso for many graph classes. Therefore, the running time of many of our algorithms $\mathcal{O}(\sqrt{n}\ell)$ while the input size is $\Omega(n+\ell)$. This cannot be avoided for the following reason.

Lemma 9.1.2. *There exists a linear-time reduction from the bipartite perfect matching problem for n vertices and m edges to* ListIso *of two independent sets with n vertices and $\ell = m$.*

Proof. We have a bipartite graph B. One part X is represented by $\boldsymbol{V}(G)$ and the other part Y by $\boldsymbol{V}(H)$. For every $u \in X$, we put $\mathfrak{L}(u) = \{v : v \in Y, uv \in \boldsymbol{E}(B)\}$. □

Similar reductions work for trees, etc. Therefore, finding bipartite perfect matchings is the bottleneck in many of our algorithms and cannot be avoided: if it cannot be solved in linear time, ListIso for many graph classes cannot be solved in linear time as well.

9.1.2 Basic Complexity Results

In this section, we prove some basic results concerning the complexity of ListIso and ListAut.

Lemma 9.1.3. *Both problems* ListAut *and* ListIso *are polynomially equivalent.*

Proof. To see that ListAut is polynomially reducible to ListIso just set H to be a copy of G and keep the lists for all vertices of G. It is straightforward to check that these two instances are equivalent. For the other direction, we build an instance G' and $\mathfrak{L}'$ of ListAut as follows. Let G' be a disjoint union of G and H. And let $\mathfrak{L}'(v) = \mathfrak{L}(v)$ for all $v \in \boldsymbol{V}(G)$ and set $\mathfrak{L}'(w) = \boldsymbol{V}(G)$ for all $w \in \boldsymbol{V}(H)$. It is easy to see that there exists list-compatible isomorphism from G to H, if and only if there exists a list-compatible automorphism of G'. □

Lemma 9.1.4. *Let $\mathcal{C}$ be a class of graphs for which* GraphIso *can be solved in polynomial time. Then* ListAut *can be solved in time $\mathcal{O}(n^c \cdot g)$ where c is some constant and g is the size of the automorphism group* $\mathrm{Aut}(G)$*.*

Proof. Using [270], we compute the generators of $\mathrm{Aut}(G)$ using $\mathcal{O}(n^3)$ instances of the graph isomorphism problem. Then we generate all g automorphisms and for each automorphism, we test whether it is list-compatible. $\square$

Lemma 9.1.5. *The problem* ListIso *can be solved in time $\mathcal{O}(n+m)$ when all lists are of size at most two.*

Proof. We construct a list-compatible isomorphism $\pi : G \to H$ by solving a 2-Sat formula which can be done in linear time [110, 9]. When $w \in \mathfrak{L}(v)$, we assume that $\deg(v) = \deg(w)$, otherwise we remove w from $\mathfrak{L}(v)$. Notice that if $\mathfrak{L}(u) = \{w\}$, we can set $\pi(u) = w$ and for every $v \in N(u)$, we modify $\mathfrak{L}(v) := L(v) \cap N(w)$. Now, for every vertex u_i with $\mathfrak{L}(u_i) = \{w_i^0, w_i^1\}$, we introduce a variable x_i such that $\pi(u_i) = w_i^{x_i}$. Clearly, the mapping π is compatible with the lists.

We construct a 2-SAT formula such that there exists a list-compatible isomorphism if and only if it is satisfiable. First, if $\mathfrak{L}(u_i) \cap \mathfrak{L}(u_j) \neq \emptyset$, we add implications for x_i and x_j such that $\pi(u_i) \neq \pi(u_j)$. Next, when $\pi(u_i) = w_i^j$, we add implications that every $u_j \in N(u_i)$ is mapped to $N(w_i^j)$. If $\mathfrak{L}(u_j) \cap N(w_i^j) \neq \emptyset$, otherwise u_i cannot be mapped to w_i^j and $x_i \neq j$. Therefore, π obtained from a satisfiable assignment maps $N[u]$ bijectively to $N[\pi(u)]$ and it is an isomorphism. The total number of variables in n, and the total number of clauses is $\mathcal{O}(n+m)$, so the running time is $\mathcal{O}(n+m)$. $\square$

Lemma 9.1.6. *Let $G_1, \dots, G_k$ be the components of G and $H_1, \dots, H_k$ be the components of H. If we can decide* ListIso *in polynomial time for all pairs G_i and H_j, then we can solve* ListIso *for G and H in polynomial time.*

Proof. Let $G_1, \dots, G_k$ be the components of G and $H_1, \dots, H_k$ be the components of H. For each component G_i, we find all components H_j such that there exists a list-compatible isomorphism from G_i to H_j. Notice that a necessary condition is that every vertex in G_i contains one vertex of H_j in its list. So we can go through all lists of G_i and find all candidates H_j, in total time $\mathcal{O}(\ell)$ for all components $G_1, \dots, G_k$. Let $n' = |\boldsymbol{V}(G_i)|$, $m' = |\boldsymbol{E}(G_i)|$, and ℓ' be the total size of lists of G_i restricted to H_j. We test existence of a list-compatible isomorphism in time $\varphi(n', m', \ell')$. Then we form the bipartite graph B between $G_1, \dots, G_k$ and $H_1, \dots, H_k$ such that $G_iH_j \in \boldsymbol{E}(B)$ if and only if there exists a list-compatible isomorphism from G_i to H_j. There exists a list-compatible isomorphism from G to H, if and only if there exists a perfect matching in B. Using Lemma 9.1.1, this can be tested in time $\mathcal{O}(\sqrt{k}\ell)$. The total running time depends on the running time of testing ListIso of the components, and we note that the sum of the lengths of lists in these test is at most ℓ. $\square$

Lemma 9.1.7. *The problem* ListIso *can be solved for cycles in time $\mathcal{O}(\ell)$.*

Proof. We may assume that $|\boldsymbol{V}(G)| = |\boldsymbol{V}(H)|$. Let $u \in \boldsymbol{V}(G)$ be a vertex with a smallest list and let $k = |\mathfrak{L}(u)|$. Since $\ell = \mathcal{O}(kn)$, it suffices to show that we can find a list-compatible isomorphism in time $\mathcal{O}(kn)$. We test all the k possible mappings $\pi \colon G \to H$ with $\pi(u) \in \mathfrak{L}(u)$. For $u \in \boldsymbol{V}(G)$ and $v \in \mathfrak{L}(u)$, there are at most two possible isomorphisms that map u to v. For each of these isomorphism, we test whether they are list-compatible. $\square$

Lemma 9.1.8. *The problem* LISTISO *can be solved for graphs of maximum degree 2 in time $\mathcal{O}(\sqrt{n}\ell)$.*

Proof. Both graphs G and H are disjoint unions of paths and cycles of various lengths. For each two connected components, we can decide in time $\mathcal{O}(\ell')$ whether there exists a list-compatible isomorphism between them, where ℓ' is the total size of lists when restricted to these components: for paths trivially, and for cycles by Lemma 9.1.7. The rest follows from Lemma 9.1.6, where the running time is of each test in $\mathcal{O}(\ell')$ where ℓ' is the total length of lists restricted to two components. □

9.2 GI-completeness of the Graph Isomorphism Implies NP-completeness of List Restricted Graph Isomorphism

Suppose that graph isomorphism is GI-complete for some class of graphs $\mathcal{C}'$. We want to show that in most cases, this translates into NP-completeness of LISTISO for $\mathcal{C}'$.

Vertex-gadget Reductions. Suppose that GRAPHISO is GI-complete for a class $\mathcal{C}$. To show that GRAPHISO is GI-complete for another class $\mathcal{C}'$, one builds a polynomial-time reduction ψ from GRAPHISO of $\mathcal{C}$: given graphs $G, H \in \mathcal{C}$, we construct graphs $G', H' \in \mathcal{C}'$ in polynomial time such that $G \cong H$ if and only if $G' \cong H'$. Such reductions were described for certain graph classes (e.g., chordal graphs [262]) and they were systematically studied in [74].

We say that ψ uses *vertex-gadgets*, if to every vertex $u \in \boldsymbol{V}(G)$ (resp. $u \in \boldsymbol{V}(H)$), it assigns a *vertex-gadget* $\mathcal{V}_u$, and these gadgets are subgraphs of G' (resp. of H'), and satisfy the following two conditions:

1. Every isomorphism $\pi : G \to H$ induces an isomorphism $\pi' : G' \to H'$ such that $\pi(u) = v$ implies $\pi'(\mathcal{V}_u) = \mathcal{V}_v$.
2. Every isomorphism $\pi' : G' \to H'$ maps vertex-gadgets to vertex-gadgets and induces an isomorphism $\pi : G \to H$ such that $\pi'(\mathcal{V}_u) = \mathcal{V}_v$ implies $\pi(u) = v$.

Theorem 9.2.1. *Let $\mathcal{C}$ and $\mathcal{C}'$ be classes of graphs. Suppose that there exists a polynomial-time reduction ψ using vertex-gadgets from* GRAPHISO *of $\mathcal{C}$ to* GRAPHISO *of $\mathcal{C}'$. Then there exists a polynomial-time reduction from* LISTISO *of $\mathcal{C}$ to* LISTISO *of $\mathcal{C}'$.*

Proof. Let $G, H \in \mathcal{C}$ be an instance of LISTISO. Using the reduction ψ, we construct the corresponding graphs $G', H' \in \mathcal{C}'$ with vertex-gadgets. We need to add lists for $\boldsymbol{V}(G')$, we initiate them empty. Let $u \in \boldsymbol{V}(G)$. To all vertices w of $\mathcal{V}_u$, we add $\bigcup_{v \in \mathfrak{L}(u)} \boldsymbol{V}(\mathcal{V}_v)$ to $\mathfrak{L}(w)$. For the vertices of G' outside vertex-gadgets, we set the lists equal to the union of all remaining vertices of H'.

We want to argue that there exists a list-compatible isomorphism $\pi' : G' \to H'$, if and only if there exists a list-compatible isomorphism $\pi : G \to H$. If π exists, by the first assumption of the reduction, it induces π' which is list-compatible by our construction of lists. On the other hand, suppose that there exists a list-compatible isomorphism π'. By the second assumption, π' maps vertex-gadgets to vertex-gadgets and induces an isomorphism $\pi : G \to H$ which is list-compatible by our construction. □

Corollary 9.2.2. *Let $\mathcal{C}$ be a class of graphs with* NP*-complete* LISTISO*. Suppose that there exists a reduction ψ using vertex-gadgets from* GRAPHISO *of $\mathcal{C}$ to* GRAPHISO *of $\mathcal{C}'$. Then* LISTISO *is* NP*-complete for $\mathcal{C}'$.*

Among others, this implies NP-completeness of LISTISO for the following graph classes:

Corollary 9.2.3. *The problem* LISTISO *is* NP*-complete for bipartite graphs, split and chordal graphs, chordal bipartite and strongly chordal graphs, trapezoid graphs, comparability graphs of dimension 4, grid intersection graphs, line graphs, and self-complementary graphs.*

Proof. We use Corollary 9.2.2 together with Theorem 6.5.1. We briefly describe GI-hardness reductions for every mentioned class. It is easy to check that, except for line graphs and self-complementary graphs, all these reductions use vertex-gadgets, where $\mathcal{V}_u = \{u\}$ for every $u \in \boldsymbol{V}(G) \cup \boldsymbol{V}(H)$.

Bipartite graphs. Assuming the graphs are not cycles, we subdivide every edge in the input graphs G and H.

Split and chordal graphs [262]. We subdivide every edge in G and H and add the complete graphs on the original vertices.

Chordal bipartite and strongly chordal graphs [346]. For bipartite graphs G and H, we subdivide all edges e_i twice, by adding vertices a_i and b_i, we add paths of length three from a_i to b_i, and we add the complete bipartite graph between a_i's and b_i's.

Trapezoid graphs [335]. For bipartite graphs G and H, we subdivide every edge and add the complete bipartite graph on the original vertices.

Comparability graphs of dimension at most 4 [225]. Assuming the graphs are not cycles, we replace every edge in G and H by a path of length 8.

Grid intersection graphs [345]. For bipartite graphs G and H, we subdivide every edge twice and add the complete bipartite graph on the original vertices.

Line graphs [358, 175]. Assuming the graphs are not K_3 and $K_{1,3}$, we consider G' and H' being the line graphs of G and H. For every $u \in \boldsymbol{V}(G)$, we put $\mathcal{V}_u = \{e : e \in \boldsymbol{E}(G), u \in e\}$, and similarly for $u \in \boldsymbol{V}(H)$. By Whitney Theorem [358], $G \cong H$ if and only if $G' \cong H'$, and it is easy to observe that it is a reduction using vertex-gadgets.

Self-complementary graphs [69]. A graph H is self-complementary if and only if $H \cong \overline{H}$ where $\overline{H}$ is the complement of H. Notice that the path of length 3 is self-complementary. We first describe a polynomial-time reduction from GRAPHISO of general graphs to GRAPHISO of self-complementary graphs.

For an arbitrary graph G, let $G_1, \dots, G_4$ be four copies of G. We construct G' as depicted in Fig. 9.1 as the disjoint union of G_1, $\overline{G_2}$, $\overline{G_3}$, and G_4. Further, we connect all vertices in $\boldsymbol{V}(G_i)$ with $\boldsymbol{V}(G_{i+1})$. The graph G' is self-complementary; see Fig. 9.1.

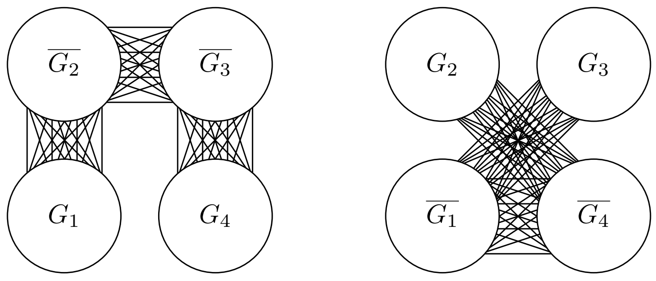


Figure 9.1: On the left, the construction of G' from four copies of G. On the right, $\overline{G'}$ is depicting, showing that G' is a self-complementary graph.

All vertices of G_1 and G_4 have degrees at most $2n-1$ in G' and all vertices of $\overline{G_2}$ and $\overline{G_3}$ have degrees at least $2n$. Since all vertices of G_1 have common neighbors in $\overline{G_2}$, but there are no edges between $\boldsymbol{V}(\overline{G_2})$ and $\boldsymbol{V}(G_4)$, we can find these four copies of G in G'. Therefore, $G \cong H$ if and only if $G' \cong H'$. The reduction is clearly polynomial.

It remains to define vertex-gadgets. For every $u \in \boldsymbol{V}(G)$, we put $\mathcal{V}_u = \{u_1\}$, where $u_1 \in \boldsymbol{V}(G_1)$ is the copy of u. This reduction clearly uses vertex-gadgets.

□

We are not aware of any polynomial-time reduction for graph isomorphism used in the literature which cannot be easily modified to use vertex-gadgets. The reason is that most of the reductions use the following operations:

- Taking the complement of the graph.
- Replacing all vertex by small disjoint isomorphic gadgets.
- Replacing all edge by small disjoint isomorphic gadgets.
- Taking disjoint copies of the graph or its complement. (We can set vertex-gadgets equal to the vertices in one copy only.)
- Adding a universal vertex, adjacent to all vertices.
- Adding a complete subgraph on some vertices or a complete bipartite graph between two sets of vertices.

For instance, all reductions described in [74] can be easily modified to use vertex-gadgets.

9.3 NP-completeness for 3-regular Colored Graphs

Using group theory techniques, graph isomorphism can be solved in polynomial time for graphs of bounded degree [263] and for colored graphs with color classes of bounded size [134]. In this section, we modify the reduction of Lubiw [260] to show that ListIso remains NP-complete even for 3-regular colored graphs with color classes of size at most 8 and each list of size at most 3.

The reduction of Lubiw [260] is from 3-Sat, but we instead use 1-in-3 Sat which is NP-complete by Schaefer [315]: all literals are positive, each clause is of size

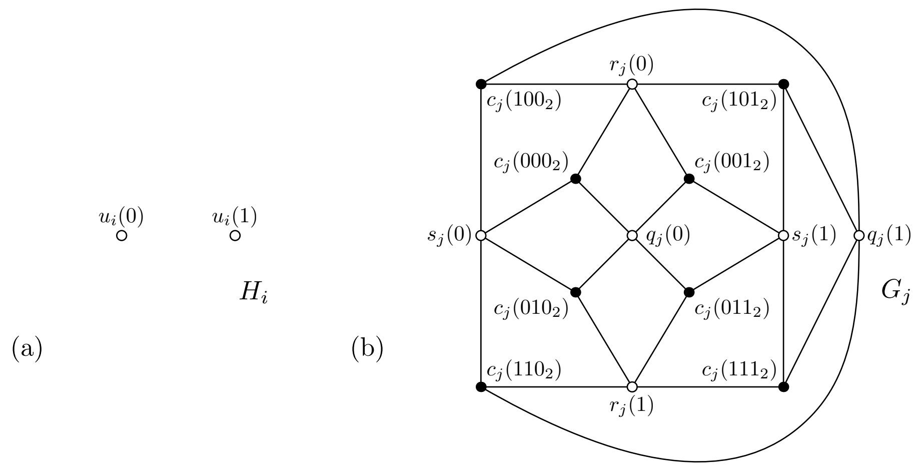


Figure 9.2: (a) The variable gadget H_i. (b) The black vertices form the clause gadget G_j, adjacent to white vertices of variable gadgets.

3 and a satisfying assignment has exactly one true literal in each clause. We show that an instance of 1-IN-3 SAT can be solved using LISTAUT. We further simplify the reduction since a fixed-point free automorphism is not required for LISTAUT.

Variable Gadget. For each variable u_i, we construct the *variable gadget H_i* which consists of two isolated vertices $u_i(0)$ and $u_i(1)$; see Fig. 9.2a. We assign $\mathfrak{L}(u_i(0)) = \mathfrak{L}(u_i(1)) = \big\{u_i(0), u_i(1)\big\}$. There exist two list-compatible automorphisms of H_i: the transposition α_i swapping $u_i(0)$ and $u_i(1)$ and the identity β_i fixing both $u_i(0)$ and $u_i(1)$.

Clause Gadget. Let c_j be a clause with the literals q_j, r_j, and s_j. For every such clause c_j, the *clause gadget G_j* consists of the isolated vertices $c_j(0), \dots, c_j(7)$. For every $k = 0, \dots, 7$, we consider its binary representation $k = abc_2$, for $a, b, c \in \{0, 1\}$. The vertex $c_j(k)$ has three neighbors $q_j(a)$, $r_j(b)$, and $s_j(c)$ belonging to the variable gadgets of its literals; see Fig. 9.2b. We assign the list

$$\mathfrak{L}(c_j(k)) = \{c_j(k \oplus 100_2), c_j(k \oplus 010_2), c_j(k \oplus 001_2)\},$$

where $\oplus$ denotes the bitwise XOR; i.e., $\mathfrak{L}(c_j(k))$ contains all $c_j(k')$ in which k' differs from k in exactly one bit. Let G be the resulting graph consisting of all variable and clause gadgets.

Lemma 9.3.1. *Suppose that π' is a partial automorphism of G obtained by choosing α_i or β_i on each variable gadget H_i. There exists a unique automorphism π extending π' such that $\pi(G_j) = G_j$.*

Proof. Let c_j be a clause with the literals q_j, r_j, and s_j. We claim that $\pi(c_j(k))$ is determined by the images of its neighbors. Recall that β_i preserves the vertices of H_i, but α_i swaps them. Therefore, one neighbor of $\pi(c_j(k))$ is different from the corresponding neighbor of $c_j(k)$ for every application of α_i on q_j, r_j and s_j. Let $p = abc_2$ such that $a = 1$, $b = 1$ and $c = 1$ if and only if α_i is applied on the variable gadget of q_j, r_j, and s_j, respectively. Then $\pi(c_j(k)) = c_j(k \oplus p)$; otherwise π would not be an automorphism. $\square$

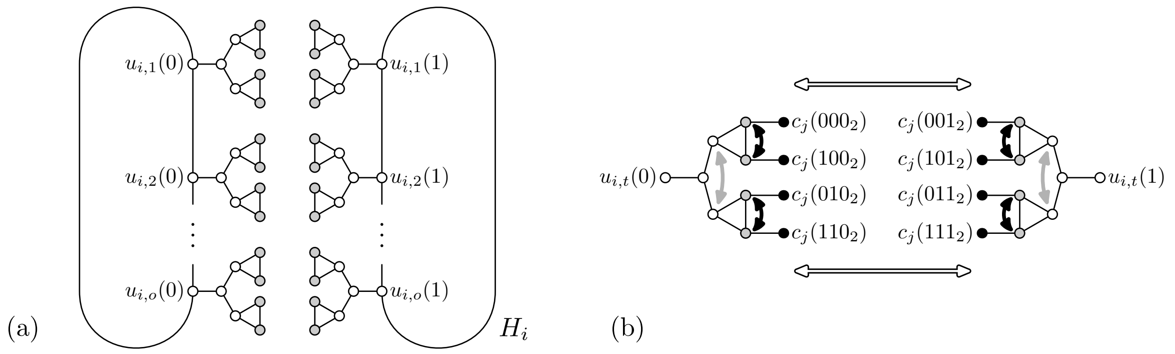


Figure 9.3: (a) The variable gadget H_i. (b) The connection between H_i and G_j. Suppose that the variable u_i has a literal in the clause c_j, so $k = yzx_2$. We connect H_i with G_j as depicted. Suppose that an automorphism π maps $c_j(k)$ to $c_j(k \oplus p)$. We show the action of π on the vertices of H_i when $p = 001$ (in white), $p = 010$ (in gray), and $p = 100$ (in black).

Lemma 9.3.2. *The* 1-IN-3 SAT *formula is satisfiable if and only if there exists a list-compatible automorphism of G.*

Proof. Let T be a truth value assignment satisfying the input formula. We construct a list-compatible automorphism π of G. If $T(u_i) = 1$, we put $\pi|_{H_i} = \alpha_i$, and if $T(u_i) = 0$, we put $\pi|_{H_i} = \beta_i$. By Lemma 9.3.1, this partial isomorphism has a unique extension to an automorphism π of G. It is list-compatible since T satisfies the 1-in-3 condition, so $\pi(c_j(k)) = c_j(k \oplus p)$, for $p \in \{100_2, 010_2, 001_2\}$.

For the other implication, let π be a list-compatible automorphism. Then $\pi|_{H_i}$ is either equal α_i, or β_i, which gives the values $T(u_i)$. By Lemma 9.3.1, $\pi(c_j(k)) = c_j(k \oplus p)$ and since π is a list-compatible isomorphism, we have $p \in \{100_2, 010_2, 001_2\}$. Therefore, exactly one literal in each clause is true, so all clauses are satisfied in T. □

The described reduction clearly runs in polynomial-time, so we have established a proof of Theorem 6.5.1. For colored graphs, we require that automorphisms preserve colors. By altering the above reduction, we get the following:

Theorem 9.3.3. *The problem* LISTISO *is* NP*-complete for 3-regular colored graphs for which each color class is of size at most 8 and each list is of size at most 3.*

Proof. We modify the graph G to a 3-regular graph. For a clause gadget G_j representing c_j, every vertex $c_j(k)$ already has degree 3. On the other hand, suppose that a variable u_i has o literals in the formula. Then both vertices of H_i have degrees $4o$, so we have to modify the variable gadgets.

We replace H_i by two cycles of length o, consisting of the vertices $u_{i,1}(0), \dots, u_{i,o}(0)$ and $u_{i,1}(1), \dots, u_{i,o}(1)$, respectively. To each of these vertices, we attach a small gadget depicted in Fig. 9.3a. We have $\mathfrak{L}(u_{i,t}(0)) = \mathfrak{L}(u_{i,t}(1)) = \Big\{u_{i,t}(0), u_{i,t}(1)\Big\}$. Again, there are two list-compatible automorphisms: α_i exchanging these two cycles by swapping $u_{i,t}(0)$ with $u_{i,t}(1)$, and β_i which is the identity fixing all $2o$ vertices. We note that when $o \le 2$, we get parallel edges or loops; if we want to avoid this, we may replace edges of two cycle by some 3-regular subgraphs.

Consider the attached gadgets to the vertices $u_{i,t}(0)$ and $u_{i,t}(1)$ corresponding to one literal of a clause c_j. Each vertex depicted in gray is adjacent to exactly one $c_j(k)$ of G_j, as depicted in Fig. 9.3b. Each k consists of three bits, denoted x, y and z (in some order). The bit x corresponds to this literal of u_i (i.e, x is the first bit for q_j being a literal of u_i, and so on). The gray vertices of gadgets attached to $u_{i,t}(j)$ are adjacent to $c_j(k)$ with $x = j$. Adjacent pairs of gray vertices are connected to $c_j(k)$ where k differs in the bit y. Non-adjacent pairs of gray vertices in one gadget are connected to $c_j(k)$ where k differs in the bit z.

In Fig. 9.3b, the action of $\mathbb{Z}_2^3$ is depicted. Lemma 9.3.1 translated to the modified definitions of variable gadgets which implies correctness of the reduction. The lists for the vertices of the attached gadgets are created as images of three depicted automorphisms; they clearly are of size at most 3.

The constructed graph G is 3-regular and all lists of G are of size at most 3. We color the vertices by the orbits of all list-compatible automorphisms and their compositions. Notice that each color class is of size at most 8. $\square$

With Lemma 9.1.8, we get a dichotomy for the maximum degree: ListIso can be solved in time $\mathcal{O}(\sqrt{n}\ell)$ for the maximum degree 2, and it is NP-complete for the maximum degree 3. Similarly, Lemma 9.1.5 implies a dichotomy for the list sizes: ListIso can be solved in time $\mathcal{O}(n+m)$ where all lists are of size 2, and it is NP-complete for lists of size at most 3. For the last parameter, the maximum size of color classes, there is a gap. Lemma 9.1.5 implies that ListIso can be solved in time $\mathcal{O}(n+m)$ when all color classes are of size 2 while it is NP-complete for size at most 8.

9.4 Trees

In this section, we modify the standard algorithm for tree isomorphism to solve list restricted isomorphism of trees. We may assume that both trees G and H are rooted, otherwise we root them by their centers (and possibly subdivide the central edges). The algorithm for GraphIso process both trees from bottom to the top. Using dynamic programming, it computes for every vertex possible images using possible images of its children. This algorithm can be modified to ListIso.

Theorem 9.4.1. *The problem* ListIso *can be solved for trees in time* $\mathcal{O}(\sqrt{n}\ell)$.

Proof. We apply the same dynamic algorithm with lists and update these lists as we go from bottom to the top. After processing a vertex u, we compute an updated list $\mathfrak{L}'(u)$ which contains all elements of $\mathfrak{L}(u)$ to which u can be mapped compatibly with its descendants. To initiate, each leaf u of G has $\mathfrak{L}'(u) = \{w : w \text{ is a leaf and } w \in \mathfrak{L}(u)\}$.

Next, we want to compute $\mathfrak{L}'(u)$ and we know $\mathfrak{L}'(u_i)$ of all children $U = \{u_1, \dots, u_k\}$ of u. For each $w \in \mathfrak{L}(u)$ with k children $w_1, \dots, w_k$, we want to decide whether to put $w \in \mathfrak{L}'(u)$. Let $W = \{w_1, \dots, w_k\}$. Each u_i can be mapped to all vertices in $\mathfrak{L}'(u_i) \cap W$. We need to decide whether all u_i's can be mapped simultaneously. Therefore, we form a bipartite graph $B(U, W)$ between U and W: we put an edge u_iw_j if and only if $w_j \in \mathfrak{L}'(u_i)$. Simultaneous mapping is possible if and only if there exists a perfect matching in this bipartite graph.

Let r be the root of G and r' be the root of H. We claim that there is a list-compatible isomorphism $\pi : G \to H$, if and only if $\mathfrak{L}'(r) = \{r'\}$. Suppose that π exists. When $\pi(u) = w$, its children U are mapped to W. Since this mapping is compatible with the lists, $w \in \mathfrak{L}(u)$, and the mapping of $u_1, \dots, u_k$ gives a perfect matching in $B(U, W)$. Therefore, $w \in \mathfrak{L}'(u)$, and by induction $r' \in \mathfrak{L}'(r)$. On the other hand, we can construct π from the top to the bottom. We start by putting $\pi(r) = r'$. When $\pi(u) = w$, we map its children U to W according to some perfect matching in $B(U, W)$ which exists from the fact that $w \in \mathfrak{L}'(u)$.

It remains to argue details of the complexity. We process the tree which takes time $\mathcal{O}(\ell)$ (assuming $n \le \ell$) and we process each list constantly many times which takes $\mathcal{O}(\ell)$. Suppose that we want to compute $\mathfrak{L}'(u)$. We consider all vertices $w^1, \dots, w^p \in \mathfrak{L}(u)$, and let W^j be the children of w^j. We go through all lists of $\mathfrak{L}'(u_1), \dots, \mathfrak{L}'(u_k)$ in linear time, and split them into sublists $\mathfrak{L}'(u_i^j)$ of vertices whose parent is w^j. Only these sublists are used in the construction of the bipartite graph $B(U, W^j)$. Using Lemma 9.1.1, we decide existence of a perfect matching in time $\mathcal{O}(\sqrt{k}\ell_j)$ which is at most $\mathcal{O}(\sqrt{n}\ell_j)$, where ℓ_j is the total size of all sublists $\mathfrak{L}'(u_i^j)$. When we sum this complexity for all vertices u, we get the total running time $\mathcal{O}(\sqrt{n}\ell)$. □

9.5 Planar Graphs

In this section, we describe how to solve ListIso on planar graphs. We use the 3-connected reduction described in Chapter 7. For the purpose of this section, we consider extended graphs introduced in Section 7.1. Every isomorphism maps vertices and half-edges while preserving incidencies. We consider the problem ListIso with lists on both vertices and half-edges.

3-connected Planar Graphs.

Lemma 9.5.1. *The problem* ListIso *(with lists on both vertices and half-edges) can be solved for 3-connected planar graphs in time $\mathcal{O}(\ell)$.*

Proof. We start by computing embeddings of both G and H, in time $\mathcal{O}(n)$. It remains to decide whether there exists a list-compatible isomorphism which has to be a map isomorphism. By Euler Theorem, we know that the average degree is less than six. Consider all vertices of degree at most 5, let u be such a vertex with a smallest list, and let $k = |\mathfrak{L}(u)|$. We have $\ell = \Omega(kn)$ and we show that we can decide existence of a list-compatible isomorphism in time $\mathcal{O}(kn)$.

We test all possible mappings $\pi : G \to H$ having $\pi(u) \in \mathfrak{L}(u)$. For each, we have at most 10 possible ways how to extend this mapping on the neighbors of u, and the rest of the mapping is uniquely determined by the embeddings and can be computed in time $\mathcal{O}(n)$. In the end, we test whether the constructed mapping π is an isomorphism and whether it is list-compatible. □

3-connected Reduction.

Theorem 9.5.2. *Let $\mathcal{C}$ be a class of connected graphs closed under contractions and taking connected subgraphs. Suppose that* ListIso *with lists on both vertices and half-edges can be solved for 3-connected graphs in $\mathcal{C}$ in time $\varphi(n, m, \ell)$. We can solve* ListIso *on $\mathcal{C}$ in time $\mathcal{O}(\sqrt{m}\ell + m + \varphi(n, m, \ell))$.*

Proof. We compute simplified reduction trees T_G and T_H (without colors and edge types) for both G and H in time $\mathcal{O}(n + m)$. We apply the idea of Theorem 9.4.1 to test list-compatible isomorphism of T_G and T_H. We compute the lists $\mathfrak{L}(N)$ for the nodes N of T_G, from the bottom to the root. A node $M \in \mathfrak{L}(N)$ if there exists a list-compatible isomorphism from N to M mapping ∂N to ∂M and there exists list-compatible isomorphism between attached subtrees. (Further, if $|\partial N| = |\partial M| = 2$, we remember in $\mathfrak{L}(N)$ which of both possible mappings of ∂N to ∂M can be extended as list-compatible isomorphisms.)

Suppose that N has the children $N_1, \dots, N_k$ with computed lists and M has the children $M_1, \dots, M_k$. There exists a list-compatible isomorphism mapping the subtree of N_i to the subtree of M_j, if and only if $M_j \in \mathfrak{L}(N_i)$. The difference from Theorem 9.4.1 is these subtrees have to be compatible with a list-isomorphism from N to M; so it depends on the structure of the nodes N and M.

There, we compute $\mathfrak{L}(N)$ differently according to the type of N:

- *Star block atoms or dipoles.* For star block atoms, similarly as in Theorem 9.4.1, we construct a bipartite graph between $N_1, \dots, N_k$ and $M_1, \dots, M_k$ and test existence of a perfect matching using Lemma 9.1.1. For dipoles, we test two possible isomorphisms, construct two bipartite graph and test existence of perfect matchings.
- *Non-star block or proper atoms.* We modify the lists of ∂N to the vertices of ∂M only. (When they are proper atoms, we run this in two different ways.) We encode the lists $\mathfrak{L}(N_1), \dots, \mathfrak{L}(N_k)$ by lists on the corresponding darts of N (depending on which of two possible list-isomorphisms of ∂N_i are possible), and we remove single pendant edges, and intersect their lists with the lists of the incident vertices. For a proper atom, we further consider N^+ and M^+ with added edges e and f such that $\mathfrak{L}(e) = \{f\}$. If the nodes are K_2 or cycles, and we can test existence of a list-compatible isomorphism using Lemma 9.1.7. If both are 3-connected, we can test it by our assumption in time $\varphi(n', m', \ell')$. If this list-compatible isomorphism exists, we add M to $\mathfrak{L}(N)$.
- *The root primitive graphs.* We use the same approach as above, ignoring the part about ∂N and ∂M.

A list-compatible isomorphism from G to H exists, if and only if $M \in \mathfrak{L}(N)$ for the root nodes N and M of T_G and T_H.

The correctness of the algorithm can be argued from the fact that all automorphisms are captured by the reduction trees (see Section 7.6), inductively from the top to the bottom as in Theorem 9.4.1. It remains to discuss the running time. The simplified reduction trees can be computed in linear time using Lemma 7.7.1. When computing $\mathfrak{L}(N)$, we first consider the lists of all vertices and edges of N. A node

M is a candidate for $\mathfrak{L}(N)$, if every vertex and every edge of N has a vertex/edge of M in its list. Therefore, we can find all these candidate nodes by iterating these lists, in linear time with respect to their total size. Let M be one of them, and let $n' = |\boldsymbol{V}(N)|$, $m' = |\boldsymbol{E}(N)|$ and ℓ' be the total size of lists of the vertices and edges of N when restricted only to the vertices and edges of M. Either we construct a bipartite graph and test existence of a perfect matching in time $\mathcal{O}(\sqrt{m'}\ell')$, or we test existence of a list-compatible isomorphism in time $\varphi(n', m', \ell')$. The total running time spent on the tree is $\mathcal{O}(\ell)$, the total running time spent testing perfect matchings is $\mathcal{O}(\sqrt{m}\ell)$, and the total running time testing list-compatible isomorphisms of 3-connected graphs is $\mathcal{O}(\varphi(n, m, \ell))$. □

General Planar Graphs. By putting both results together, we get the following:

Theorem 9.5.3. *The problem* ListIso *can be solved for planar graphs in time $\mathcal{O}(\sqrt{n}\ell)$.*

Proof. If G and H are connected, we use Theorem 9.5.2. By Lemma 9.5.1, the function $\varphi(n, m, \ell)$ is $\mathcal{O}(\ell)$. If G and H are disconnected, we apply Lemma 9.1.6 on all connected components of G and H, and by analysing the proof, the total running time is $\mathcal{O}(\sqrt{n}\ell)$. □

9.6 Interval, Permutation and Circle Graphs

In this section, we prove that the standard algorithms solving GraphIso on interval, circle and permutation graphs can be modified to solve ListIso on them. The key idea is that the structure of these graph classes can be captured by graph-labeled trees which are unique up to isomorphism and which capture the structure of all automorphisms; see [224, 225] and the references therein.

For interval graphs, we use MPQ-tree. For circle graphs, we use split trees. For permutation graphs, we use modular trees. On these trees, we apply a bottom-up procedure similarly as in the proof of Theorem 9.4.1. The key difference is that nodes correspond to either prime, or degenerate graphs. Degenerate graphs are simple and lead to perfect matchings in bipartite graphs. Prime graphs have a small number of automorphisms [224, 225], so all of them can be tested.

9.6.1 Interval Graphs

Recall MPQ-trees from Sections 3.1 and 4.2. To each interval graph G, a unique MPQ-tree T_G is assigned, and $G \cong H$ if and only if T_G and T_H are equivalent. From [225, Lemma 4.3], it follows that every isomorphism $\pi : G \to H$ is obtained by an equivalence transformation of T_G and some permutation of the vertices in identical sections. Therefore, we can apply a bottom-up procedure to test ListIso for MPQ-trees, similarly as in Theorem 9.4.1.

Theorem 9.6.1. *The problem* ListIso *can be solved for interval graphs in time $\mathcal{O}(\sqrt{n}\ell + m)$.*

Proof. We proceed similarly as in Theorem 9.4.1. We compute MPQ-trees T_G and T_H representing the interval graphs G and H in linear time [235]. Then we compute lists $\mathfrak{L}(N)$ for every node N of T_G from the bottom. We distinguish the three types of nodes of an MPQ-tree.

- *Leaf nodes.* Let L_G be a leaf node in T_G and let L_H be a leaf node in T_H. Then $L_H \in \mathfrak{L}(L_G)$ if there exists a list-compatible isomorphism between the induced complete subgraphs $G[\mathrm{sec}(L_G)]$ and $H[\mathrm{sec}(L_H)]$.

- *P-nodes.* Let N and M be P-nodes of T_G and T_H, respectively. We want to decide whether $M \in \mathfrak{L}(N)$. Let $N_1, \dots, N_k$ be the children of N and let $M_1, \dots, M_k$ be the children of M_k. We construct a bipartite graph similarly as in Theorem 9.4.1. Then $M \in \mathfrak{L}(N)$ if there exists a perfect matching in the bipartite graph and a perfect matching between the lists of $G[\mathrm{sec}(N)]$ and $H[\mathrm{sec}(M)]$ (which are complete graphs).

- *Q-nodes.* Let N and M be Q-nodes of T_G and T_H and let $N_1, \dots, N_k$ and $M_1, \dots, M_k$ be their children. Here we have at most two possible isomorphisms. In particular, an isomorphism can either map the subtree of N_i on the subtree of M_i, or in the reversed order, and we can test for both possibilities whether the lists $\mathfrak{L}(N_i)$ are compatible. Moreover, we consider all sets of intervals belonging to exactly the same sections of the Q-node, and we test by perfect matchings between pairs of them whether there exists a list-compatible isomorphism between them.

The MPQ-trees have $\mathcal{O}(n)$ nodes and $\mathcal{O}(n)$ intervals in their sections. For leaf nodes and P-nodes, the analysis is exactly the same as in the proof of Theorem 9.4.1. For Q-nodes, we just test two possible mappings and bipartite matchings for sections. We get the total running time $\mathcal{O}(\sqrt{n}\ell + m)$. □

9.6.2 Permutation Graphs

Recall the modular decomposition from Section 2.7.1. We use it to show that the problem ListIso can be decided in $\mathcal{O}(\sqrt{n}\ell + m)$ time for permutation graphs.

Theorem 9.6.2. *The problem* ListIso *can be solved for permutation graphs in time* $\mathcal{O}(\sqrt{n}\ell + m)$.

Proof. For input graph G and H, we first compute the modular trees T_G and T_H, respectively, in time $\mathcal{O}(n + m)$ [272]. We again apply the idea of Theorem 9.4.1. We compute the list $\mathfrak{L}(N)$ for every node N of T_G. Note that all inner nodes consist only of marker vertices which have no lists. Therefore, we first compute $\mathfrak{L}(L)$, for every leaf node. A leaf node K is in $\mathfrak{L}(L)$ if every non-marker vertex of L has a non-marker vertex of K in its list. These candidate nodes for $\mathfrak{L}(L)$ can be found in linear time in the total size of lists by iterating through the lists of vertices of L.

Let N be a node of the modular tree. It has the children $N_1, \dots, N_k$ with computed lists $\mathfrak{L}(N_1), \dots, \mathfrak{L}(N_k)$. Let M be a node with the children $M_1, \dots, M_k$.

There exist a list-compatible isomorphism mapping the subtree of N_i to the subtree of M_j if and only if $M_j \in \mathfrak{L}(N_i)$. Moreover these subtrees have to be compatible with a list isomorphism from N to M. We compute $\mathfrak{L}(N)$ according to the type of N.

- *Degenerate nodes.* For degenerate nodes, we proceed similarly as for trees in Theorem 9.4.1. We construct a bipartite graph between the nodes nodes $N_1, \dots, N_k$ and $M_1, \dots, M_k$ and test for a perfect matching using Lemma 9.1.1.
- *Prime nodes.* For prime nodes, there are at most four possible isomorphisms mapping N to M [225, Lemma 6.6]. We test for these four possible isomorphisms π whether $\pi(M_i) \in \mathfrak{L}(M_i)$ for every M_i.

A list compatible isomorphism exists if $M \in \mathfrak{L}(N)$, for the root nodes N and M of T_G and T_H. The correctness of the algorithm follows from the fact that all automorphisms of a permutation graph are captured by the modular tree [225]. A similar argument as in the proofs of Theorems 9.4.1 and 9.6.1 gives the running time. □

9.6.3 Circle Graphs

Recall split decomposition and split trees from Section 2.6.

Testing ListIso. Next, we show that the problem ListIso can be solved on circle graphs in polynomial time.

Theorem 9.6.3. *The problem* ListIso *can be solved for circle graphs in polynomial time.*

Proof. For input graph G and H, we first compute the split trees T_G and T_H, in polynomial time using [150, 79]. We assume that the trees T_G and T_H are rooted and we can also assume that the roots are prime or degenerate nodes. We again apply the idea of Theorem 9.4.1.

We compute the list $\mathfrak{L}(N)$ for every node N of T_G. Let M be a leaf node of T_H and let $m_N \in \boldsymbol{V}(N)$ and $m_M \in \boldsymbol{V}(M)$ be the marker vertices incident to a tree edge closer to the root. Then M is in $\mathfrak{L}(N)$ if there is a list-compatible isomorphism from N to M which maps m_N to m_M.

Let N be a node of the split tree. It has the children $N_1, \dots, N_k$ with computed lists $\mathfrak{L}(N_1), \dots, \mathfrak{L}(N_k)$. Let M be a node with the children $M_1, \dots, M_k$. There exist a list-compatible isomorphism mapping the subtree of N_i to the subtree of M_j if and only if $M_j \in \mathfrak{L}(N_i)$. Moreover these subtrees have to be compatible with an isomorphism from N to M. We compute $\mathfrak{L}(N)$ according to the type of N.

- *Degenerate nodes.* For degenerate nodes, we proceed similarly as for trees in Theorem 9.4.1. We construct a bipartite graph between the nodes nodes $N_1, \dots, N_k$ and $M_1, \dots, M_k$ and test for a perfect matching using Lemma 9.1.1.
- *Prime nodes.* For prime nodes, there are at most four possible isomorphisms mapping m_N to m_M [224, Lemma 5.6]. We test those four possible isomorphisms, construct four bipartite graphs and test existence of perfect matchings.

- *The root node.* If it is degenerate, we proceed as above. If it is prime, then its automorphism groups is a subgroup of a dihedral group [224, Lemma 5.5]; essentially it behaves as a cycle. Therefore, we approach it similarly as in Lemma 9.1.7.

A list compatible isomorphism exists if $M \in \mathfrak{L}(N)$, for the root nodes N and M of T_G and T_H.

The correctness of the algorithm follows from the fact that all automorphisms of a circle graph are captured by the split tree [224]. The running time is clearly polynomial. □

9.7 Bounded Genus Graphs

In this section, we describe an FPT algorithm solving ListIso when parameterized by the Euler genus g. We modify the recent paper of Kawarabayashi [206] solving graph isomorphism in linear time for a fixed genus g. The harder part of this paper are structural results, described below, which transfer to list-compatible isomorphisms without any change. Using these structural results, we can build our algorithm.

Theorem 9.7.1. *For every integer g, the problem* ListIso *can be solved on graphs of Euler genus at most g in time $\mathcal{O}(\sqrt{n}\ell)$.*

Proof. See [206, p. 14] for overview of the main steps. We show that these steps can be modified to deal with lists. We prove this result by induction on g, where the base case for $g = 0$ is Theorem 9.5.3. Next, we assume that both graphs G and H are 3-connected, otherwise we apply Theorem 9.5.2. By [206, Theorem 1.2], if G and H have no polyhedral embeddings, then the face-width is at most two.

Case 1: G and H have polyhedral embeddings. Following [206, Theorem 1.2], we have at most $f(g)$ possible embeddings of G and H. We choose one embedding of G and we test all embeddings of H. It is known that the average degree is $\mathcal{O}(g)$. Therefore, we can apply the same idea as in the proof of Lemma 9.5.1 and test isomorphism of all these embeddings in time $\mathcal{O}(\ell)$.

Case 2: G and H have no polyhedral embedding, but have embeddings of face-width exactly two. Then we split G into a pair of graphs (G', L). The graph L are called *cylinders* and the graph G' correspond to the remainder of G. The following properties hold [206, p. 5]:

- We have $G = G' \cup L$ and for $\partial L = \boldsymbol{V}(G' \cap L)$, we have $|\partial L| = 4$.
- The graph G' can be embedded to a surface of genus at most $g - 1$, and L is planar [206, p. 4].
- This pair (G', L) is canonical, i.e., every isomorphism from G to H maps (G', L) to another pair (H', L') in H.

It is proved [206, Theorem 5.1] that there exists some function $q'(g)$ bounding the number of these pairs both in G and H, and can be found in time $\mathcal{O}(n)$. We fix a pair (G', L) in G and iterate over all pairs (H'_i, L'_i) in H. Following [206, p. 36], we get that

$G \cong H$, if and only if there exists a pair (H'_i, L'_i) in H such that $G' \cong H'_i$, $L \cong L'_i$, and $G' \cap L$ is mapped to $H'_i \cap L'_i$. To test this, we run at most $2q'(g)$ instances of ListIso on smaller graphs with modified lists.

Suppose that we want to test whether $G' \cong H'_i$ and $L \cong L'_i$. First, we modify the lists: for $u \in \boldsymbol{V}(G')$, put $\mathfrak{L}'(u) = \mathfrak{L}(u) \cap H'_i$, and for $v \in \boldsymbol{V}(L)$, put $\mathfrak{L}'(v) = \mathfrak{L}(v) \cap L'_i$, and similarly for lists of darts. Further, for all vertices $u \in \partial L$ in both G' and L, we put $\mathfrak{L}'(u) = \mathfrak{L}(u) \cap \partial L$. We test existence of list-compatible isomorphisms from G' to H'_i and from L to L'_i. There exists a list-compatible isomorphism from G to H, if and only if these list-compatible isomorphisms exist at least for one pair (H'_i, L'_i).

We note that when $g = 2$, a special case is described in [206, Theorem 5.3], which is slightly easier and can be modified similarly.

Case 3: G and H have no polyhedral embedding and have only embeddings of face-width one. Let V be the set of vertices in G such that for each $u \in V$, there exists a non-contractible curve passing only through u. By [206, Lemma 6.3], $|V| \le q(g)$ for some function q. For u, the non-contractible curve divides its edges to two sides, so we can cut G at u, and split the incident edges. We obtain a graph G' which can be embedded to a surface of genus at most $g - 1$.

By [206, Lemma 6.3], we can find all these vertices V and V' in G and H in time $\mathcal{O}(n)$. We choose $u \in V$ arbitrarily, and we test all possible vertices $v \in V'$. Let G' be constructed from G by splitting u into new vertices u' and u'', and similarly H' be constructed from H by splitting v into new vertices v' and v''. In [206, p. 36], it is stated that $G \cong H$, if and only if there exists a choice of $v \in V'$ such that $G' \cong H'$ and $\{u', u''\}$ is mapped to $\{v', v''\}$. Therefore, we run at most $q(g)$ instances of ListIso on smaller graphs with modified lists.

If $v \notin L(u)$, clearly a list-compatible isomorphism is not possible for this choice of $v \in V'$. If $v \in L(u)$, we put $L'(u') = L'(u'') = \{v', v''\}$. Then there exists a list-compatible isomorphism from G to H, if and only if there exists a list-compatible isomorphism from G' to H'.

The correctness of our algorithm follows from [206]. It remains to argue the complexity. Throughout the algorithm, we produce at most $w(g)$ subgraphs of G and H, for some function w, for which we test list-compatible isomorphisms. Assuming the induction hypothesis, the reduction of graphs to 3-connected graphs can be done in time $\mathcal{O}(\sqrt{n}\ell)$. Case 1 can be solved in time $\mathcal{O}(\ell)$. Case 2 can be solved in time $\mathcal{O}(\sqrt{n}\ell)$. Case 3 can be solved in time $\mathcal{O}(\sqrt{n}\ell)$. □

9.8 Bounded Treewidth Graphs

In this section, we prove that ListIso can be solved in FPT with respect to the parameter treewidth $\mathrm{tw}(G)$. Unlike in Sections 9.5 and 9.6, the difficulty of graph isomorphism on bounded treewidth graphs raises from the fact that tree decomposition is not uniquely determined. We follow the approach of Bodlaender [37] which describes an XP algorithm for GraphIso of bounded treewidth graphs, running in time $n^{\mathcal{O}(\mathrm{tw}(G))}$. Then we show that the recent breakthrough by Lokshtanov et al. [256], giving an FPT algorithm for GraphIso, translates as well.

Definition 9.8.1. A *tree decomposition* of a graph G is a pair $\mathcal{T} = (\{B_i \colon i \in I\}, T = (I, F))$, where T is a rooted tree and $\{B_i \colon i \in I\}$ is a family of subsets of V, such that

1. for each $v \in \boldsymbol{V}(G)$ there exists an $i \in I$ such that $v \in B_i$,
2. for each $e \in \boldsymbol{E}(G)$ there exists an $i \in I$ such that $e \subseteq B_i$,
3. for each $v \in \boldsymbol{V}(G), I_v = \{i \in I \colon v \in B_i\}$ induces a subtree of T.

We call the elements B_i the *nodes*, and the elements of the set F the decomposition edges.

We define the width of a tree decomposition $\mathcal{T} = (\{B_i \colon i \in I\}, T)$ as $\max_{i \in I} |B_i| - 1$ and the *treewidth* $\operatorname{tw}(G)$ of a graph G as the minimum width of a tree decomposition of the graph G.

Nice Tree Decompositions. It is common to define a *nice tree decomposition* of the graph [227]. We naturally orient the decomposition edges towards the root and for an oriented decomposition edge (B_j, B_i) from B_j to B_i we call B_i the *parent* of B_j and B_j a *child* of B_i. If there is an oriented path from B_j to B_i we say that B_j is a *descendant* of B_i.

We also adjust a tree decomposition such that for each decomposition edge (B_i, B_j) it holds that $\big||B_i| - |B_j|\big| \le 1$ (i.e. it joins nodes that differ in at most one vertex). The in-degree of each node is at most 2 and if the in-degree of the node B_k is 2 then for its children B_i, B_j holds that $B_i = B_j = B_k$ (i.e. they represent the same vertex set).

We classify the nodes of a nice decomposition into four classes—namely *introduce nodes*, *forget nodes*, *join nodes* and *leaf nodes*. We call the node B_i an introduce node of the vertex v, if it has a single child B_j and $B_i \setminus B_j = \{v\}$. We call the node B_i a forget node of the vertex v, if it has a single child B_j and $B_j \setminus B_i = \{v\}$. If the node B_k has two children, we call it a join node (of nodes B_i and B_j). Finally we call a node B_i a leaf node, if it has no child.

Bodlaender's Algorithm. A graph G has treewidth at most k if either $|\boldsymbol{V}(G)| \le k$, or there exists a cut set $U \subseteq \boldsymbol{V}(G)$ such that $|U| \le k$ and each component of $G \setminus U$ together with U has treewidth at most k. The set U corresponds to a bag in some tree decomposition of G. Bodlaender's algorithm [37] enumerates all possible cut sets U of size at most k in G (resp. H), we denote these C_i (resp. D_i). Furthermore, it enumerates all connected components of $G \setminus C_i$ as C_i^j (resp. of $H \setminus D_i$ as D_i^j). We denote by $G[U, W]$ the graph induced by $U \mathbin{\dot\cup} W$. The set W is either a connected component or a collection of connected components. We call U the *border set*.

Lemma 9.8.2 ([8, 37])**.** *A graph $G[U, W]$ with at least k vertices has a treewidth at most k with the border set U if and only if there exists a vertex $v \in W$ such that for each connected component A of $G[W \setminus v]$, there is a k-vertex cut $C_s \subseteq U \cup \{v\}$ such that no vertex in A is adjacent to the (unique) vertex in $(U \cup \{v\}) \setminus C_s$, and $G[C_s, A]$ has treewidth at most k.*

Lemma 9.8.3. *The problem* ListIso *can be solved in* XP *with respect to the parameter treewidth.*

Proof. We modify the algorithm of Bodlaender [37]. Let $k = \text{tw}(G) = \text{tw}(H)$. We compute the sets C_i, C_i^j for G and the sets $D_{i'}, D_{i'}^{j'}$ for H; there are $n^{\mathcal{O}(k)}$ pairs (C_i, C_i^j). The pair (C_i, C_i^j) is *compatible* if C_i^j is a connected component of $G' \setminus C_i$ for some $G' \subseteq G$ that arises during the recursive definition of treewidth. Let $f\colon C_i \to D_{i'}$ be an isomorphism. We say that $(C_i, C_i^j) \equiv_f (D_{i'}, D_{i'}^{j'})$ if and only if there exists an isomorphism $\varphi\colon C_i \cup C_i^j \to D_{i'} \cup D_{i'}^{j'}$ such that $\varphi|_{C_i} = f$. In other words, φ is a partial isomorphism from G to H. The change for ListIso is that we also require that both f and φ are list-compatible.

The algorithm resolves $(C_i, C_i^j) \equiv_f (D_{i'}, D_{i'}^{j'})$ by the dynamic programming, according to the size of $D_{i'}^{j'}$. If $|C_i^j| = |D_{i'}^{j'}| \le 1$, we can check it trivially in time $k^{\mathcal{O}(k)}$. Otherwise, suppose that $|C_i^j| = |D_{i'}^{j'}| > 1$, and let m be the number of components of C_i^j (and thus $D_{i'}^{j'}$). We test whether $f : C_i \to D_{i'}$ is a list-compatible isomorphism. Let $v \in C_i^j$ be a vertex given by Lemma 9.8.2 (with $U = C_i$ and $W = C_i^j$) and let C_s be the corresponding extension of v to a cut set. We compute for all $w \in D_{i'}^{j'} \cap \mathfrak{L}(v)$ all connected components B_q. From the dynamic programming, we know for all possible extensions D' of w to a cut set whether $(C_m, A_p) \equiv_{f'} (D', B_q)$ with $f'(x) = f(x)$ for $x \in C_i$ and $f'(v) = w$. Finally, we decide whether there exists a perfect matching in the bipartite graph between (C_m, A_p)'s and (D', B_q)'s where the edges are according to the equivalence. □

Reducing The Number of Possible Bags. Otachi and Schweitzer [290] proposed the idea of pruning the family of potential bags which finally led to an FPT algorithm [256]. A family $\mathcal{B}(G)$, whose definition depends on the graph, is called *isomorphism-invariant* if for an isomorphism $\phi : G \to G'$, we get $\mathcal{B}(G') = \phi(\mathcal{B}(G))$, where $\phi(\mathcal{B}(G))$ denotes the family $\mathcal{B}(G)$ with all the vertices of G replaced by their images under ϕ.

For a graph G, a pair (A, B) with $A \cup B = V$ is called a *separation* if there are no edges between $A \setminus B$ and $B \setminus A$ in G. The order of (A, B) is $|A \cap B|$. For two vertices $u, v \in \boldsymbol{V}(G)$, by $\mu(u, v)$ we denote the minimum order of separation (A, B) with $u \in A \setminus B$ and $v \in B \setminus A$. We say a graph G is *k-complemented* if $\mu_G(u, v) \ge k) \implies uv \in \boldsymbol{E}(G)$ holds for every two vertices $u, v \in V$. We may canonically modify the input graphs G and H ListIso, by adding these additional edges and making them k-complemented.

Theorem 9.8.4 ([256], Theorem 5.5)**.** *Let k be a positive integer, and let G be a graph on n vertices that is connected and k-complemented. There exists an algorithm that computes in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^3$ an isomorphism-invariant family of bags $\mathcal{B}$ with the following properties:*

1. $|B| \le \mathcal{O}(k^4)$ *for each* $B \in \mathcal{B}$*,*
2. $|\mathcal{B}| \le 2^{\mathcal{O}(k^5 \log k)} \cdot n^2$*,*

3. *Assuming* $\mathrm{tw}(G) < k$*, the family* $\mathcal{B}$ *captures some tree decomposition of* G *that has width* $\mathcal{O}(k^4)$*.*

4. *The family* $\mathcal{B}$ *is closed under taking subsets.*

Theorem 9.8.5. *The problem* ListIso *can be solved in* FPT *time* $2^{\mathcal{O}(k^5 \log k)} n^5$ *where* $k = \mathrm{tw}(G)$*.*

Proof. We use the algorithm of Lemma 9.8.3, where C_i's and D_i's are from the collection $\mathcal{B}$ of Theorem 9.8.4. The total number of pairs (C_i, C_i^j) and $(D_{i'}, D_{i'}^{j'})$ is bounded by $2^{\mathcal{O}(k^5 \log k)} n^3$ [256, p. 20]. The dynamic programming in [256, Theorem 6.2] is done according to the potential function $\Phi(D_{i'}, D_{i'}^{j'}) = 2|D_{i'}^{j'}| + |D_{i'}|$. We use nice tree decompositions, so in each step, the dynamic programming either introduces a new node into the bag $D_{i'}$, or moves a node from the bag $D_{i'}$ to $D_{i'}^{j'}$, or joins several pairs with the same bag $D_{i'}$. In all these operations, we check existence of a list-compatible isomorphism, using dynamic programming, exactly as in Lemma 9.8.3. □

9.9 Conclusions

We conclude this chapter with description of related results and open problems.

Forbidden Images. We note that Lubiw [260] used a different definition of ListIso: for every vertex $u \in \boldsymbol{V}(G)$, we are given a *list of forbidden images* $\mathcal{F}(u) \subseteq \boldsymbol{V}(H)$ and we want to find an isomorphism $\pi : G \to H$ such that $\pi(u) \notin \mathcal{F}(u)$. The advantage of forbidden lists is that we can express GraphIso in space $\mathcal{O}(n + m)$, but the input for ListIso is of size $\mathcal{O}(n^2)$. On the other hand, we consider lists of allowed images more natural (for instance, list coloring is defined similarly) and also such a definition appears naturally in [118]. Both statements are clearly polynomially equivalent, and the main focus of our paper is to distinguish between tractable and intractable cases for ListIso.

Group Reformulation. Luks [263] described the following group problem which generalizes computing automorphism groups of graphs. Let Ω be a ground set and let Γ be a group acting on Ω. Further, let Ω be colored. We want to compute the subgroup of Γ which is color preserving. When Γ is the symmetric group acting on all pairs of vertices $\boldsymbol{V}(G)$ which are colored by two colors (corresponding to edges and non-edges in G), then the computed subgroup is $\mathrm{Aut}(G)$. To generalize graph isomorphism in this language, we have two colorings and we want to find a color-preserving permutation $g \in \Gamma$. We note that Babai [14] calls these generalizations *string automorphism/isomorphism problems.*

A similar generalization of ListIso was suggested to us by Ponomarenko. We are given a group Γ acting on a ground set Ω and for every $x \in \Omega$, we have a list $\mathfrak{L}(x) \subseteq \Omega$. We ask whether there exists a permutation $g \in \Gamma$ such that $g(x) \in \mathfrak{L}(x)$ for every $x \in \Omega$. We obtain ListIso either similarly as above, or when $\Omega = \boldsymbol{V}(G)$ and $\Gamma = \mathrm{Aut}(G)$.

We may interpret our results for ListIso using this group reformulation. The robust combinatorial algorithms work because the groups Γ are highly restricted. In particular, for trees, Jordan [202] proved that Aut(G) is formed by a series of direct products and wreath products with symmetric groups, to it has a tree structure. Therefore, the algorithm of Theorem 9.4.1 solves ListIso on Aut(G) by a bottom-up dynamic algorithm. Similar characterizations were recently proved for interval, permutation and circle graphs [224, 225] and for planar graphs [217], and are used in the algorithms of Theorems 9.5.3, 9.6.1, 9.6.2, and 9.6.3. For graphs of bounded genus or bounded treewidth, no such detailed description of the automorphism groups is yet known, but they are likely restricted as well. On the other hand, for cubic graphs, the automorphism groups Aut(G) may be arbitrary, so this approach fails, and actually ListIso is NP-complete by Theorem 9.3.3.

To attack ListIso from the point of this group reformulation, instead of different graph classes, we may study it for different combinations of Γ and the lists $\mathfrak{L}$. First, for which groups Γ, it can be solved efficiently for all possible lists $\mathfrak{L}$? Second, for which lists $\mathfrak{L}$, the problem can be solved efficiently? We did not try to attack the problem much in the second direction, aside Lemma 9.1.5 and Theorem 9.3.3. For instance, when $\mathfrak{L}$ is a partitioning of Ω, the problem is easy since we get the usual color-preserving isomorphism problem.

***k*-dimensional Weisfieler-Leman refinement (*k*-WL).** The classical 2-WL [355, 356] colors vertices of a graph and it initiates with different colors for each degree. In each step, it takes vertices of one color class and partitions them by different numbers of neighbors of other color classes. It stops when no partitioning longer occurs. Its generalization k-WL [12, 196] colors and partitions $(k-1)$-tuples according to their adjacencies.

Certainly, when G and H are isomorphic graphs, they are partitioned and colored the same. So when $G \ncong H$, k-WL distinguishes G and H, for a suitable value of k. If we prove for a graph class that k is small enough, we obtain a combinatorial algorithm for GraphIso. For instance, Grohe [162] proves that for every graph X, there exists a value k such that two X-minor free graphs are either isomorphic, or distinguished by k-WL. This does not translate to ListIso since k-WL applied on G only estimates the orbits of Aut(G). When $G \cong H$, and we may test this, assuming that GraphIso can be decided efficiently for G and H, we obtain two identical partitions. They may be used to reduce sizes of the lists, but we still end up with the question whether there exists a list-compatible isomorphism. In Section 9.3, we show that it is NP-complete to decide ListIso even when sizes of all lists are bounded by 3.

Excluded Minors. Another major open problem for ListIso is its complexity for graphs with excluded minors. As described in Introduction, the original polynomial-time algorithm of Ponomarenko [298] for GraphIso is based heavily on group theory, and his technique unlikely translates to ListIso. But it seems doubtful that the problem will be NP-complete, since new combinatorial structural and algorithmic results may be applied.

Robertson and Seymour [307] proved that every graph G with an excluded minor can be decomposed into pieces which are "almost embeddable" to a surface of genus g, where g depends on the minor. The recent book of Grohe [162] describes the seminal idea of automorphism-invariant treelike decompositions. A *treelike decomposi-*

tion generalizes a classical tree decomposition by replacing a tree of bags by a directed acyclic graph of bags. Unlike tree decompositions, a treelike decomposition D of G can be constructed with two additional properties. Firstly, it is *automorphism-invariant*, meaning that every automorphism of G induces an automorphism of D. Secondly, it is *canonical*, meaning that for two isomorphic graphs G and G', isomorphic treelike decompositions D and D' are constructed.

The main structural result of Grohe [162] is that every graph G with an excluded minor has a canonical automorphism-invariant treelike decomposition for which the graphs induced by bags called *torsos* are "almost embeddable" to a surface of genus g, where g depends on the minor. Therefore, to solve ListIso on graphs with excluded minors, we need to prove the following:

1. We need to show that ListIso can be solved on almost embeddable graphs in polynomial time. We may use the results of Theorems 9.7.1 and 9.8.5 to do so.
2. We need to prove Lifting Lemma for ListIso, stating the following: If we can compute a canonical automorphism-invariant treelike decomposition of a class $\mathcal{C}$ in polynomial time and we can solve ListIso for its torsos $\mathcal{A}$ in polynomial time, then we can solve ListIso for $\mathcal{C}$ in polynomial time as well. Here, we might modify the algorithm for lifting of canonization by Grohe [162].

We note that it is quite difficult to understand and describe everything in detail. The book of Grohe [162] is very extensive (almost 500 pages) and described in the language of graph logics. Unfortunately, no purely combinatorial description of the results is available, and we believe that such a description of a combinatorial algorithm for solving graph isomorphism of graphs with excluded minors would be desirable. A combinatorial description of treelike decompositions is described by Grohe and Marx [163], but an algorithm for the graph isomorphism problem of graphs with excluded minors is only used as a black box.

Bounded Rankwidth. Rankwidth generalizes treewidth in the way that bounded treewidth implies bounded rankwidth, but not the opposite. Very recently, the first XP algorithm for graph isomorphism of graphs of bounded rankwidth was described [164]. The approach is by computing automorphism-invariant tree decomposition (which should translate to ListIso), but then further group theory is applied to test whether the decompositions are isomorphic. It is a very interesting question whether group theory can be avoided and the problem can be solved in a purely combinatoric way. Therefore, determining the complexity of ListIso for graphs of bounded rankwidth is one of major open problems and it might give an insight into this question as well.

Also, rankwidth is closely related to cliquewidth: when one parameter is bounded, the other is bounded as well. The graphs of cliquewidth at most 2 are called *cographs* and can be represented by *cotrees*. Their isomorphism can therefore be reduced to isomorphism of cotrees and solved in polynomial time [76], and this approach should translate to ListIso. Very recently, a combinatorial polynomial-time algorithm for graph isomorphism of graphs of cliquewidth at most 3 was described [85] which might translate to ListIso as well.

Bounded Eigenvalue Multiplicity. The polynomial-time algorithms for GraphIso of graphs of bounded eigenvalue multiplicity [15, 108] are heavily based on group theory. Actually, already the case of multiplicity one is non-trivial. It seems unlikely that these results will translate to ListIso, but constructing an NP-hardness reduction with bounded eigenvalue multiplicity seems non-trivial.

Forbidden Subgraphs, Induced Subgraphs and Induced Minors. There are several papers dealing with GraphIso for classes of graphs with excluded subgraphs, induced subgraphs and induced minors, and again the question is which results translate to ListIso. Otachi and Schweitzer [289] prove a dichotomy for excluded subgraphs. The GI-complete cases translate by Theorem 9.2.1, but the polynomial cases follow from [163] which does not seem to translate. More complicated characterizations are known for forbidden induced subgraphs [40, 242, 320]. Belmonte et al. [22] describe dichotomy for forbidden induced minors.

Logspace Results. For some graph classes, GraphIso is known to be solvable in LogSpace. It is a natural question to ask whether these results translate to ListIso. For instance, graph isomorphism of trees [254] can be solved in LogSpace, with a similar bottom-up procedure as in the proof of Theorem 9.4.1. The celebrated result of Reingold [301], stating that undirected reachability can be solved in LogSpace, allowed many other graph algorithms to be translated to LogSpace. In particular, GraphIso is known to be solvable in LogSpace for planar graphs [86, 87], k-trees [228], interval graphs [229], and bounded treewidth graphs [105].

10 3-connected Reduction for Regular Graph Covers

This chapter contains:

- *10.1: Definition of Regular Graph Covering.* We define semiregular subgroups, regular covering projections and regular quotients.
- *10.2: Regular Projections and Quotients of Atoms.* We introduce halvable atoms and describe three types of quotients of atoms.
- *10.3: Quotient Expansions.* We revert the reductions in quotients and describe all quotients H_0 of G_0 rising from the quotients H_r of G_r.
- *10.4: Quotients of Planar Graphs and Negami's Theorem.* We apply the results to planar graphs. We describe all quotients of planar atoms and primitive graphs, and thus describe all quotients of planar graphs.
- *10.5: Concluding Remarks.* We conclude with remarks and open problems.

`http://pavel.klavik.cz/orgpad/regular_covers.html`

10.1 Definition of Regular Graph Covering

A graph G *covers* a graph H (or G is a *cover* of H) if there exists a locally bijective homomorphism p called a *covering projection.* A homomorphism p from G to H is given by a mapping $p_h : \boldsymbol{H}(G) \to \boldsymbol{H}(H)$ preserving edges and incidences between half-edges and vertices. It induces two mappings $p_v : \boldsymbol{V}(G) \to \boldsymbol{V}(H)$ and $p_e : \boldsymbol{E}(G) \to \boldsymbol{E}(H)$ such that $p_e(uv) = p_v(u)p_v(v)$ for every $uv \in \boldsymbol{E}(G)$. The property to be local bijective states that for every vertex $u \in \boldsymbol{V}(G)$ the mapping p_h restricted to the half-edges incident with u is a bijection. Figure 10.1 contains two examples of graph covers. We mostly omit subscripts and just write $p(u)$ or $p(e)$.

Fibers. A *fiber* over a vertex $v \in \boldsymbol{V}(H)$ is the set $p^{-1}(v)$, i.e., the set of all vertices $\boldsymbol{V}(G)$ that are mapped to v, and similarly for fibers over edges and half-edges. We adopt the standard assumption that both G and H are connected. It follows that all fibers of p are of the same size. In other words, $\boldsymbol{h}(G) = k \cdot \boldsymbol{h}(H)$ and $\boldsymbol{v}(G) = k \cdot \boldsymbol{v}(H)$ for some $k \in \mathbb{N}$ which is the size of each fiber, and we say that G is a *k-fold cover* of H.

Regular Coverings. We aim to consider regular coverings which are highly symmetric. From the two examples from Fig. 10.1, the regular covering p is more symmetric than the non-regular covering p'.

An action of a group of automorphism is called *semiregular* if it has no non-trivial (i.e., non-identity) stabilizers of half-edges and vertices. Further, we require the stabilizer of an edge in a semiregular action to be trivial, unless it is a halvable edge, when it may contain an involution transposing the two half-edges. We say that a group is *semiregular* if its action is semiregular. Through the paper, the letter Γ is reserved for semiregular subgroups of $\mathrm{Aut}(G)$. We say that $\pi \in \mathrm{Aut}(G)$ is semiregular if the subgroup $\langle\pi\rangle$ is semiregular. (Note that this is equivalent to the fact that π has all cycles of the same length.)

Let Γ be any semiregular subgroup of $\mathrm{Aut}(G)$. It defines a graph G/Γ called a *regular quotient* (or simply *quotient*) of G as follows: The vertices of G/Γ are the orbits of the action Γ on $\boldsymbol{V}(G)$, the half-edges of G/Γ are the orbits of the action Γ on $\boldsymbol{H}(G)$. A vertex-orbit $[v]$ is incident with a half-edge-orbit $[h]$ if and only if the vertices

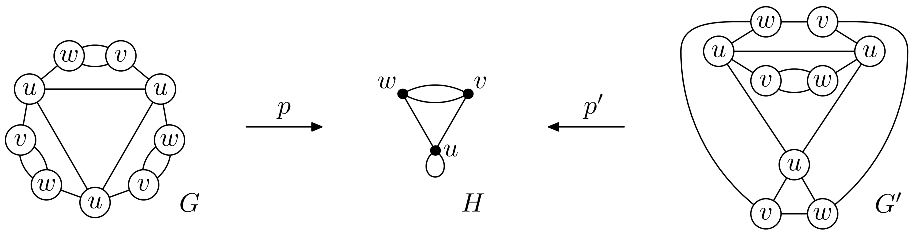


Figure 10.1: Two covers of H. The projections p_v and p'_v are written inside of the circles, and the projections p_e, p_h, p'_e, and p'_h are omitted. Notice that each loop is realized by having two neighbors labeled the same, and parallel edges are realized by having multiple neighbors labeled the same. Also covering projections preserve degrees.

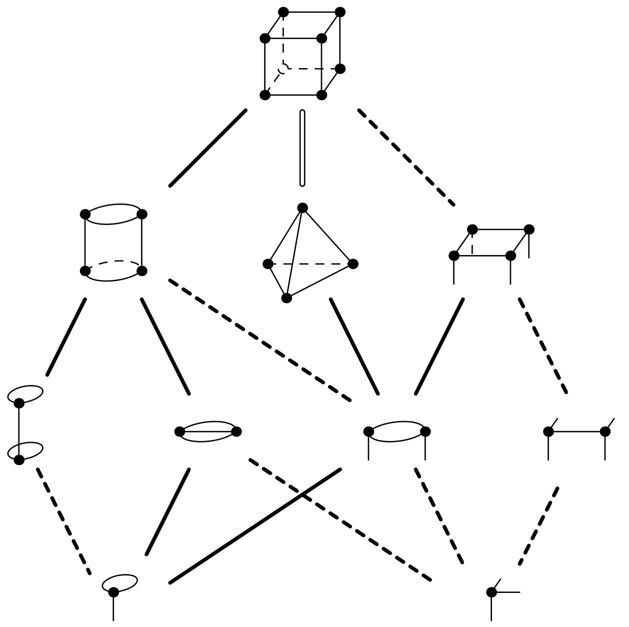


Figure 10.2: The Hasse diagram of all quotients of the cube graph depicted in a geometric way. When semiregular actions fix edges, the quotients contain half-edges. The quotients connected by bold edges are obtained by 180 degree rotations. The quotients connected by dashed edges are obtained by reflections. The tetrahedron is obtained by the antipodal symmetry of the cube, and its quotient is obtained by a 180 degree rotation with the axis going through the centers of two non-incident edges of the tetrahedron.

of $[v]$ are incident with the half-edges of $[h]$. (Because the action of Γ is semiregular, each vertex of $[v]$ is incident with exactly one half-edge of $[h]$, so this is well defined.) We say that G *regularly covers* H if there exists a regular quotient of G isomorphic to H.

We naturally construct the *regular covering projection* $p : G \to G/\Gamma$ by mapping the vertices to its vertex-orbits and half-edges to its half-edge-orbits. Concerning an edge $e \in \boldsymbol{E}(G)$, it is mapped to an edge of G/Γ if the two half-edges belong to different half-edge-orbits of Γ. If they belong to the same half-edge-orbits, it corresponds to a half-edge of G/Γ with free end. The projection p is a $|\Gamma|$-fold regular covering.

For the graphs G and H of Fig. 10.1, we get $H \cong G/\Gamma$ for $\Gamma \cong \mathbb{C}_3$ which "rotates the cycle by three vertices". As a further example, Fig. 10.2 depicts all quotients of the cube graph.

Block trees and Regular Coverings.

Lemma 10.1.1. *If G has a non-trivial semiregular automorphism, then G has a central block.*

Proof. For contradiction, suppose that G has a central articulation u. Every automorphism of a tree preserves its center, so $\mathrm{Aut}(T)$ preserves u. Also, all automorphisms of $\mathrm{Aut}(G)$ preserve u since every automorphism of $\mathrm{Aut}(G)$ induces an automorphism of $\mathrm{Aut}(T)$. This contradicts existence of a non-trivial semiregular automorphism. □

Let u be an articulation contained in the central block. By T_u we denote the subtree of T defined by u and all its predecessors, and let G_u be the graph induced by all vertices of the blocks of T_u.

Lemma 10.1.2. *Let Γ be a semiregular subgroup of* $\mathrm{Aut}(G)$*. If u and v are two articulations of the central block and of the same orbit of Γ, then $G_u \cong G_v$. Moreover there is a unique $\pi \in \Gamma$ which maps G_u to G_v.*

Proof. Notice that either $G_u = G_v$, or $G_u \cap G_v = \emptyset$. Since u and v are in the same orbit of Γ, there exists $\pi \in \Gamma$ such that $\pi(u) = v$. Consequently $\pi(G_u) = G_v$. Suppose that there exist $\pi, \sigma \in \Gamma$ such that $\pi(G_u) = \sigma(G_u) = G_v$. Then $\pi \cdot \sigma^{-1}$ is an automorphism of Γ fixing u. Since Γ is semiregular, $\pi = \sigma$. □

In the language of quotients, it means that G/Γ consists of C/Γ together with the graphs G_u attached to C/Γ, one for each orbit of Γ.

In the following, we shall assume that G contains a central block.

Why not just 2-connected?? We use the 3-connected reduction described in Chapter 7 since it behaves well with respect to automorphism groups. The behaviour of regular covering with respect to 1-cuts in G is very simple, so a natural question follows: why do we not restrict ourselves to 2-connected graphs G? The issue is that the quotient C/Γ might not be 2-connected (see Fig. 10.5 on the right), so it may consists of many blocks in H. When H contains subtree of blocks isomorphic to G_u, it may correspond to G_u, or it may correspond to a quotient of a subgraph C/Γ, together with some other G_v attached. Therefore, we work with 1-cuts together with 2-cuts and we define 3-connected reduction for 1-cuts in G as well, unlike in [340, 344, 191, 78, 352, 11]. This is essential for the algorithm for regular covering testing described in Chapter 11.

10.2 Regular Projections and Quotients of Atoms

In this section, we review the changes in atoms and the 3-connected reduction of Chapter 7 in order to capture semiregular subgroups of the automorphism groups.

Halvable Atoms. For the purpose of studying regular covering projection, we distinguish an additional symmetry type of an atom. Aside asymmetric atoms, we divide symmetric atoms into two types:

1. *A halvable atom A.* There exits a semiregular involutory automorphism $\tau \in \mathrm{Aut}(A)$ which exchanges u and v.
2. *A symmetric atom A.* The atom A is not halvable, but there exists an automorphism in $\mathrm{Aut}(A)$ which exchanges u and v.

When a halvable atom is reduced, we replace it by a halvable edge. Again, we naturally consider only the automorphisms which preserve these edge types and of course the orientation of directed edges, and we use this generalized definition to define symmetry types of their atoms. In the definition of a halvable atom, the automorphism

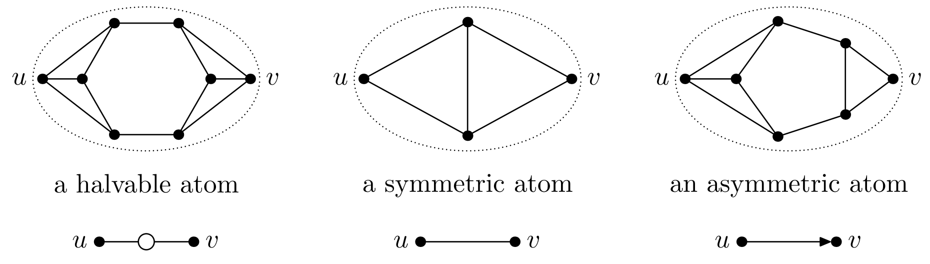


Figure 10.3: The three types of atoms and the corresponding edge types which we use in the reduction. We denote halvable edges by small circles in the middle.

τ fixes no vertices and no directed and undirected edges, but some halvable edges may be fixed. Figure 10.3.

We note that the symmetry type of atoms depends on colors and types of edges the atom contains; see Fig. 10.4 for an example. Also, the figure depicts a quotient G_2/Γ_2 of G_2, and its expansions to G_1/Γ_1 and G_0/Γ_0. The resulting quotients G_1/Γ_1 and G_2/Γ_2 contain half-edges because Γ_1 and Γ_2 fix some halvable edges but G_0/Γ_0 contains no half-edges. This example shows that in reductions and expansions we need to consider half-edges even both graphs G and G/Γ are simple.

Three Types of Projections Let Γ be a semiregular subgroup of $\mathrm{Aut}(G)$, which defines a regular covering projection $p : G \to G/\Gamma$. Negami [288, p. 166] investigated possible projections of proper atoms, and we investigate this question in more detail. Let A be an atom and $p|_A$ be its regular projection. We distinguish three different cases

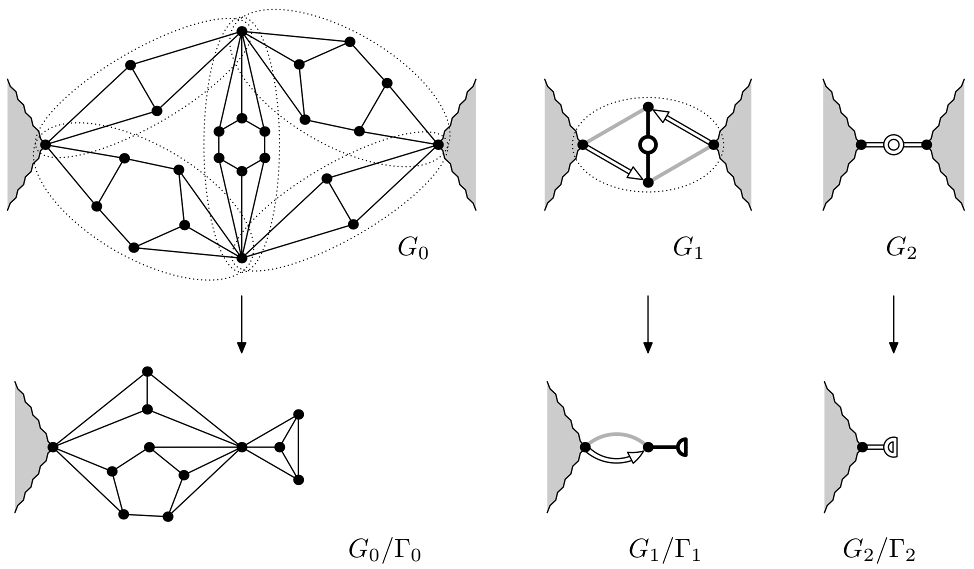


Figure 10.4: We reduce a part of a graph in two steps. In the first step, we replace five atoms by five edges of different types. As the result we obtain one halvable atom which we further reduce to one halvable edge. Notice that without considering edge types, the resulting atom in G_1 would be just symmetric. In the bottom, we show a part of the corresponding quotient graphs when Γ_i contains a semiregular involutory automorphism π from the definition of a half-projection.

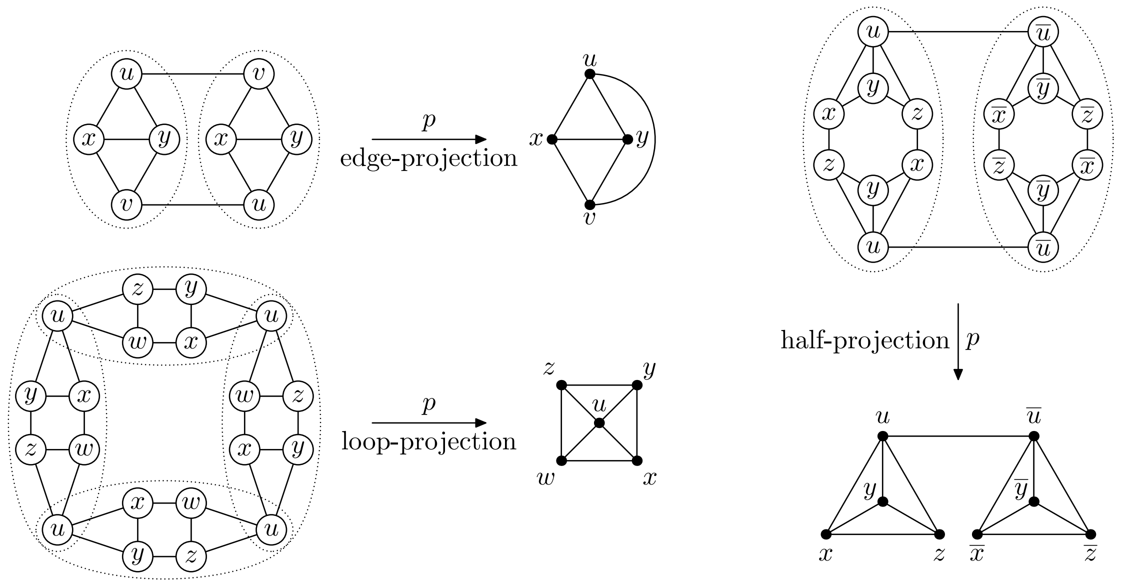


Figure 10.5: The three cases for projectios of atoms. Notice that for the third graph, an edge-projection can also be applied which gives a different quotient.

illustrated in Fig. 10.5. For a block atom A, we have exclusively an *edge-projection.* For a proper atom or a dipole A with $\partial A = \{u, v\}$, we get the following three types of projections $p|_A$:

- *An edge-projection.* The atom A is preserved in G/Γ, meaning $p(A) \cong A$. Notice that $p(A)$ is a subgraph of G/Γ, not necessarily induced. For instance for a proper atom, it can happen that $p(u)p(v)$ is adjacent, even through $uv \notin \boldsymbol{E}(G)$, as in Fig. 10.5.
- *A loop-projection.* The interior $\mathring{A}$ is preserved and the vertices u and v are identified, i.e., $p(\mathring{A}) \cong \mathring{A}$ and $p(u) = p(v)$.
- *A half-projection.* The covering projection p is a $2k$-fold cover. There exists an involutory permutation π in Γ which exchanges u and v and preserves A. Then $p(u) = p(v)$ and $p(A)$ is a halved atom A, consisting of orbits of π on A. This projection can only occur when A is a halvable atom.

Lemma 10.2.1. *For an atom A and a regular covering projection p, we have $p|_A$ either an edge-projection, a loop-projection, or a half-projection.*

Proof. Let Γ be the semiregular subgroup of $\mathrm{Aut}(G)$ defining $p : G \to G/\Gamma$. For a block atom A, Lemma 10.1.2 implies that $p(A) \cong A$, so only an edge-projection occurs. It remains to deal with A being a proper atom or a dipole, and let $\partial A = \{u, v\}$. According to Lemma 7.4.8b every automorphism π either preserves $\mathring{A}$, or $\mathring{A}$ and $\pi(\mathring{A})$ are disjoint.

Suppose that there exists a non-trivial automorphism $\pi \in \Gamma$ preserving $\mathring{A}$. By Lemma 7.4.8a, we know $\pi(\partial A) = \partial A$, and by semiregularity, π is uniquely determined and exchanges u and v. Then the fiber containing u and v has to be of an even size, with π being an involution reflecting k copies of A, and so p is a $2k$-fold covering projection. Therefore, $p|_A$ is a half-projection.

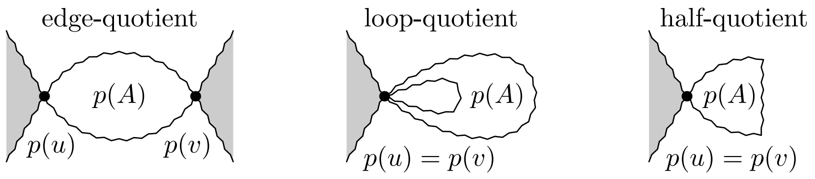


Figure 10.6: How can the quotient $p(A)$ look in G/Γ, depending on type of $p|_A$.

Suppose that there is no non-trivial automorphism which preserves $\mathring{A}$. The only difference between an edge- and a loop-projection is whether u and v are contained in one fiber of $p|_A$, or not. First, suppose that for every non-trivial $\pi \in \Gamma$ we get $A \cap \pi(A) = \emptyset$. Then no fiber contains more than one vertex of A, and $p_| A$ is an edge-projection, i.e, $A \cong p(A)$. Next, suppose that there exists $\pi \in \Gamma$ such that $A \cap \pi(A) \neq \emptyset$. By Lemma 7.4.8c, we get $A \cap \pi(A) = \partial A \cap \partial\pi(A)$, so u and v belong to one fiber of $p|_A$, which makes $p|_A$ a loop-projection. $\square$

Figure 10.6 shows the corresponding quotients $p(A)$ in G/Γ. For an edge-, a loop- and a half-projection $p|_A$, we get three types of quotients $p(A)$ of A which we call an *edge-quotient*, a *loop-quotient* and a *half-quotient*, respectively. The following lemma allows to say "the" edge- and "the" loop-quotient of an atom.

Lemma 10.2.2. *For every atom A, there is the unique edge-quotient and the unique loop-quotient up to isomorphism.*

Proof. In both cases, we have $\mathring{A} \cong \mathring{p}(A)$, so these quotients are uniquely determined. $\square$

For half-quotients, this uniqueness does not hold. First, an atom A with $\partial A = \{u, v\}$ has to be halvable to admit a half-quotient. Then each half-quotient is determined by an involutory automorphism τ exchanging u and v; here τ is the restriction of π from the definition of a half-projection. There is a one-to-many relation between non-isomorphic half-quotients and automorphisms τ, i.e., several different automorphisms τ may give the same half-quotient.

Lemma 10.2.3. *A dipole A has at most $\left\lfloor \frac{e(A)}{2} \right\rfloor + 1$ pairwise non-isomorphic half-quotients, and this bound is achieved.*

Proof. Figure 10.7 shows a construction which achieves the bound. It remains to show that it is an upper bound. Without loss of generality, we can assume that all edges of this dipole are halvable. Let τ be a semiregular involution. Edges which are fixed in τ

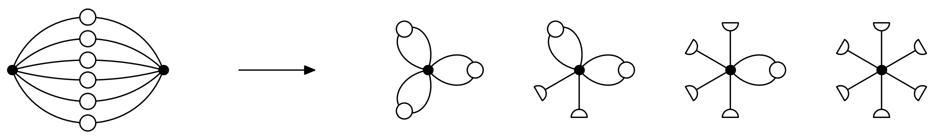


Figure 10.7: Assuming that quotients can contain half-edges, the depicted dipole has four non-isomorphic half-quotients.

correspond to half-edges in the half-quotient $A/\langle\tau\rangle$. Pairs of edges interchanged by τ give rise to loops in $A/\langle\tau\rangle$. In the quotient, we have ℓ loops and h half-edges attached to a single vertex such that $2\ell + h = \boldsymbol{e}(A)$. Since ℓ is between 0 and $\left\lfloor \frac{\boldsymbol{e}(A)}{2} \right\rfloor$, the upper bound is established. □

For planar proper atoms, we prove in Lemma 10.4.3 that there are at most two non-isomorphic half-quotients. This non-uniqueness of half-quotients is one of the main algorithmic difficulties for regular covering testing of planar graphs studied in Chapter 11.

10.3 Quotient Expansions

Suppose that $H_r = G_r/\Gamma_r$ is a quotient of G_r. The reductions applied to reach G_r are reverted on H_r and produce an *expansion series* $H_r, H_{r-1}, \dots, H_0$ of H_r. We obtain a series of semiregular subgroups $\Gamma_r, \dots, \Gamma_0$ such that $H_i = G_i/\Gamma_i$ and Γ_i extends Γ_{i+1}. The entire process is depicted in the diagram in Fig. 6.11.

The problem is that expansions are unlike reductions not uniquely determined. From H_{i+1}, we can construct multiple H_i. In this section, we characterize all possible ways how H_i can be constructed from H_{i+1}, and thus establish Theorem 6.6.1.

Reduction Epimorphism and Semiregular Subgroups. Recall the reduction epimorphism $\boldsymbol{\Phi}_i$ defined in Section 7.6. We show that it behaves nicely with respect to semiregular subgroups of $\mathrm{Aut}(G_i)$.

Lemma 10.3.1. *Let $\boldsymbol{\Phi}_i : \mathrm{Aut}(G_i) \to \mathrm{Aut}(G_{i+1})$ be the reduction epimorphism. For a semiregular subgroup Γ of $\mathrm{Aut}(G_i)$, the restriction $\boldsymbol{\Phi}_i|_\Gamma$ is an isomorphism. Moreover, the subgroup $\boldsymbol{\Phi}_i(\Gamma)$ remains semiregular.*

Proof. Recall that the kernel $\mathrm{Ker}(\boldsymbol{\Phi}_i)$ is the set of all π such that $\boldsymbol{\Phi}_i(\pi) = \mathrm{id}$ and it is a normal subgroup of $\mathrm{Aut}(G_i)$. It has the following structure: $\pi \in \mathrm{Ker}(\boldsymbol{\Phi}_i)$ if and only if it fixes everything except for the interiors of the atoms. Further, $\pi(\mathring{A}) = \mathring{\pi}(A)$, so π can non-trivially act only inside the interiors of the atoms.

For any subgroup Γ, the restricted mapping $\boldsymbol{\Phi}_i|_\Gamma$ is a group homomorphism with $\mathrm{Ker}(\boldsymbol{\Phi}_i|_\Gamma) = \mathrm{Ker}(\boldsymbol{\Phi}_i) \cap \Gamma$. If Γ is semiregular, then we show that $\mathrm{Ker}(\boldsymbol{\Phi}_i) \cap \Gamma$ is trivial. We know that G_i contains at least one atom A. The boundary ∂A is fixed by $\mathrm{Ker}(\boldsymbol{\Phi}_i)$, so by semiregularity of Γ the intersection with $\mathrm{Ker}(\boldsymbol{\Phi}_i)$ is trivial. Hence $\boldsymbol{\Phi}_i|_\Gamma$ is an isomorphism.

For the semiregularity of $\boldsymbol{\Phi}_i(\Gamma)$, let $\pi' \in \boldsymbol{\Phi}_i(\Gamma)$. Since $\boldsymbol{\Phi}_i|_\Gamma$ is an isomorphism, there exists the unique $\pi \in \Gamma$ such that $\boldsymbol{\Phi}_i(\pi) = \pi'$. If π' fixes a vertex u, then π fixes u as well, so it is the identity, and $\pi' = \boldsymbol{\Phi}_i(\mathrm{id}) = \mathrm{id}$. If π' only fixes an edge $e = uv$, then π' exchanges u and v. If π also fixes e, this edge is halvable and so π' can fix it as well. Otherwise there is an atom A in G_i replaced by e in G_{i+1}. Then $\pi|_A$ is an involutory semiregular automorphism exchanging u and v, so A is halvable. But then e is a halvable edge, and thus π' is allowed to fix it. □

Reduction Preserves Central Block. We show that the reduction preserves the central block.

Lemma 10.3.2. *Let G admit a non-trivial semiregular automorphism π. Then each G_{i+1} has a central block which is obtained from the central block of G_i by replacing its atoms by colored edges.*

Proof. By Lemma 10.3.1, semiregular automorphisms are preserved during the reduction. By Lemma 10.1.1, each G_i has a central block. Since we replace only proper atoms and dipoles in the central block, it remains to be a block after the reduction. We argue by induction that it remains central as well.

Let B be the central block of G_i and let B' be this block in G_{i+1}. Consider the subtree T'_u of the block tree T' of G_{i+1} attached to B' in u containing the longest path in T' from B'. This subtree corresponds to the subtree T_u attached at u in G_i. Let π be a non-trivial semiregular automorphism in G_i. Then $\pi(u) = v$, and by Lemma 10.1.2 we have $T_v \cong T_u$. Then T'_v corresponds in G_{i+1} to T_v after reduction and $T'_u \cong T'_v$. Therefore B' is the central block of G_{i+1}. □

If G has a non-trivial semiregular automorphism, then its central block is preserved in the primitive graph G_r. By Lemma 7.4.2, G_r is either 3-connected, or C_n, or K_2, or can be made from these graphs by attaching single pendant edges to at least two vertices.

10.3.1 Quotients and Their Expansion

Let $G_0, \dots, G_r$ be the reduction series of G and let Γ_0 be a semiregular subgroup of $\mathrm{Aut}(G_0)$. By repeated application of Lemma 10.3.1, we get the uniquely determined semiregular subgroups $\Gamma_1, \dots, \Gamma_r$ of $\mathrm{Aut}(G_1), \dots, \mathrm{Aut}(G_r)$ such that $\Gamma_{i+1} = \mathbf{\Phi}_i(\Gamma_i)$, each isomorphic to Γ_0. Let $H_i = G_i/\Gamma_i$ be the quotients with preserved colors of edges, and let p_i be the corresponding covering projection from G_i to H_i. Recall that H_i can contain edges, loops and half-edges; depending on the action of Γ_i on the half-edges corresponding to the edges of G_i.

Lemma 10.3.3. *Every semiregular subgroup Γ_i of $\mathrm{Aut}(G_i)$ corresponds to a unique semiregular subgroup Γ_{i+1} of $\mathrm{Aut}(G_{i+1})$ such that $\Gamma_{i+1} = \mathbf{\Phi}_i(\Gamma_i)$.*

Quotients Reductions. Consider $H_i = G_i/\Gamma_i$ and $H_{i+1} = G_{i+1}/\Gamma_{i+1}$. We investigate relations between these quotients. Let A be an atom of G_i represented by a colored edge e in G_{i+1}. According to Lemma 10.2.1, $p_i|_A$ is either an edge, a loop or a half-projection. It is easy to see that $\mathbf{\Phi}_i$ is defined exactly in the way that $p_{i+1}(e)$ corresponds to an edge for an edge-projection, to a loop for a loop-projection, and to a half-edge for a half-projection. (This explains these names of projections and quotients.) Figure 10.8 shows examples. We get the following commuting diagram:

$$\begin{tikzcd} G_i \arrow[r, "\text{red.}"] \arrow[d, "\Gamma_i"'] & G_{i+1} \arrow[d, "\Gamma_{i+1}"] \\ H_i \arrow[r, "\text{red.}"] & H_{i+1} \end{tikzcd} \qquad (10.1)$$

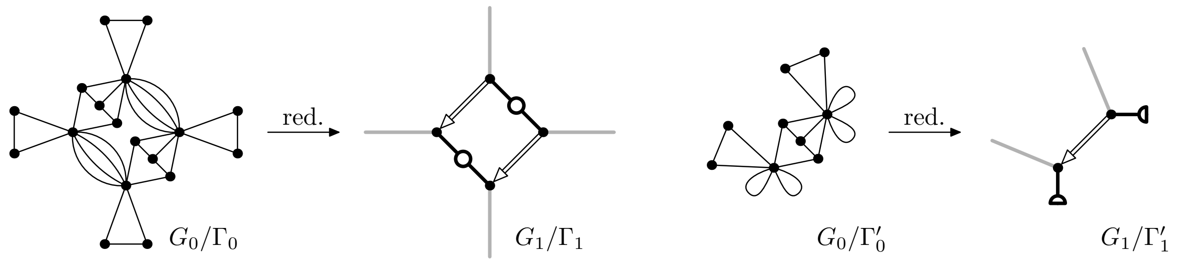


Figure 10.8: An example of two quotients of the graph G_0 from Fig. 7.14 with the corresponding quotients of the reduced graph G_1. Here $\Gamma_1 = \mathbf{\Phi}_1(\Gamma_0)$ and $\Gamma'_1 = \mathbf{\Phi}_1(\Gamma'_0)$.

So we can construct the graph H_{i+1} from H_i by replacing the projections of atoms in H_i by the corresponding projections of the edges replacing the atoms.

Overview of Quotients Expansions. Our goal is to reverse the horizontal edges in Diagram (10.1), i.e, to understand:

$$\begin{tikzcd} G_i \arrow[d, "\Gamma_i"'] & G_{i+1} \arrow[l, "\text{exp.}"'] \arrow[d, "\Gamma_{i+1}"] \\ H_i & H_{i+1} \arrow[l, "\text{exp.}"'] \end{tikzcd} \tag{10.2}$$

Let Γ_i and Γ_{i+1} be semiregular groups such that $\mathbf{\Phi}_i(\Gamma_i) = \Gamma_{i+1}$. Then we call Γ_{i+1} a *reduction* of Γ_i, and Γ_i an *extension* of Γ_{i+1}. There are two fundamental questions we address in this section in full detail:

- *Question 1.* Given a group Γ_{i+1}, which semiregular groups Γ_i are its extensions? Notice that all these groups Γ_i are isomorphic to Γ_{i+1} as abstract groups, but they correspond to different actions on G_i.
- *Question 2.* Let Γ_i and Γ'_i be two semiregular groups extending Γ_{i+1}. Under which conditions are the quotients $H_i = G_i/\Gamma_i$ and $H'_i = G_i/\Gamma'_i$ different?

Extensions of Group Actions. We first deal with Question 1. Our approach is similar as in the proof of Proposition 7.6.4, but because of semiregularity we do not need an extra assumption for existence of involutory automorphisms exchanging boundaries of symmetric proper atoms.

Lemma 10.3.4. *For every semiregular group $\Gamma_{i+1} \le \mathrm{Aut}(G_{i+1})$, there exists an extension $\Gamma_i \le \mathrm{Aut}(G_i)$ such that Diagram (10.1) commutes.*

Proof. First notice that Γ_{i+1} determines the action of Γ_i everywhere on G_i except for the interiors of the atoms of G_i, so we just need to define it there.Let $e = uv$ be one edge of G_{i+1} replacing an atom A in G_i. Let $|\Gamma_{i+1}| = k$. We distinguish three cases, see Fig. 7.17:

Case 1: The atom A is a block atom. Then the orbit $[e]$ contains exactly k edges. Let $\partial A = \{u\}$, $[e] = \{e_1, \dots, e_k\}$, and $u_i = \pi'(u)$ for the unique $\pi' \in \Gamma_{i+1}$ mapping e to e_i. (We know that π' is unique because Γ_{i+1} is semiregular.) Let $A_1, \dots, A_k$ be the atoms of G_i corresponding to $e_1, \dots, e_k$ in G_{i+1}. The edges $e_1, \dots, e_k$ have the same color and type, and thus the block atoms A_i are pairwise isomorphic.

We define the action of Γ_i on the interiors of $A_1, \dots, A_k$ as follows. As in the proof of Proposition 7.6.4, we choose arbitrarily isomorphisms $\sigma_{1,i}$ from A_1 to A_i such that $\sigma_{1,i}(u_1) = u_i$, and put $\sigma_{1,1} = \mathrm{id}$ and $\sigma_{i,j} = \sigma_{1,j}\sigma_{1,i}^{-1}$. If $\pi'(e_i) = e_j$, we set $\pi|_{\mathring{A}_i} = \sigma_{i,j}|_{\mathring{A}_i}$. By (7.2), the composition of the extensions π_1 and π_2 of π'_1 and π'_2 is defined on the interiors of $A_1, \dots, A_\ell$ exactly as the extension of $\pi_2\pi_1$. Also, (7.2) implies that an identity $\pi'_k\pi'_{k-1} \cdots \pi'_1 = \mathrm{id}$ extends to the identity. Hence the extended action remains semiregular.

Case 2: The atom A is a proper atom or a dipole and the orbit $[e]$ contains exactly k edges. Let $e = uv$ and $[e] = \{e_1, \dots, e_k\}$. We define $u_i = \pi'(u)$ and $v_i = \pi'(v)$ similarly as above. The rest of the argument is similar as in Case 1, we just require that $\sigma_{1,i}(u_1) = u_i$ and $\sigma_{1,i}(v_1) = v_i$.

Case 3: The atom A is a proper atom or a dipole and the orbit $[e]$ contains exactly $\ell = \frac{k}{2}$ edges. Then we have k half-edges in one orbit, so in H_{i+1} we get one half-edge. Let $[e] = \{e_1, \dots, e_\ell\}$. They have to be halvable, and consequently the corresponding atoms $A_1, \dots, A_\ell$ are halvable. Let u_i be an arbitrary endpoint of e_i and let v_i be the second endpoint of e_i. Again, we arbitrarily choose isomorphisms $\sigma_{1,i}$ from A_1 to A_i such that $\sigma_{1,i}(u_1) = u_i$ and $\sigma_{1,i}(v_1) = v_i$, and define $\sigma_{i,j} = \sigma_{1,j}\sigma_{1,i}^{-1}$.

Since A_1 is a halvable atom, we further consider a semiregular involution τ_1 of A_1 which exchanges u_1 and v_1. Again, τ_1 defines a semiregular involution of A_i by conjugation as $\tau_i = \sigma_{1,i}\tau_1\sigma_{1,i}^{-1}$. We put $\hat\sigma_{i,j} = \sigma_{i,j}\tau_i = \tau_j\sigma_{i,j}$ which is an isomorphism mapping A_i to A_j such that $\hat\sigma_{i,j}(u_i) = v_j$ and $\hat\sigma_{i,j}(v_i) = u_j$. In the extension, we put $\pi|_{\mathring{A}_i} = \sigma_{i,j}|_{\mathring{A}_i}$ if $\pi'(u_i) = u_j$, and $\pi|_{\mathring{A}_i} = \hat\sigma_{i,j}|_{\mathring{A}_i}$ if $\pi'(u_i) = v_j$. As in the proof of Proposition 7.6.4, it follows that the composition $\pi_2\pi_1$ is correctly defined as above, and it maps identities to identities.

To conclude the proof, it is easy to observe that by semiregularity of Γ_{i+1} the constructed group Γ_i acts semiregularly on G_i, as well. □

Corollary 10.3.5. *The construction in the proof of Lemma 10.3.4 gives all possible extensions of Γ_{i+1} such that Diagram (10.1) commutes.*

Proof. Recall that the atoms of G_i form blocks in the action of any extension Γ_i, and the action on the blocks is prescribed by Γ_{i+1}. An extension Γ_i of Γ_{i+1} gives in Cases 1 and 2 the isomorphisms $\sigma_{1,i}$, for $i = 1, \dots k$, and in Case 3 the isomorphisms $\sigma_{1,i}$, $\hat\sigma_{1,i}$ and τ_i, for $i = 1, \dots \frac{k}{2}$. However, by semiregularity, given these isomorphisms the extension is uniquely determined. □

Quotient Expansion. Recall the description of quotients of atoms from Section 10.2. We are ready to establish the main theorem of this chapter. It states that every quotient H_i of G_i can be created from some quotient H_{i+1} of G_{i+1} by replacing edges, loops and half-edges of atoms replaced in the reduction from G_i to G_{i+1} with corresponding edge-, loop- and half-quotients.

Proof of Theorem 6.6.1. Let $H_{i+1} = G_{i+1}/\Gamma_{i+1}$ and let H_i be constructed in the above way. We first argue that H_i is a quotient of G_i, i.e., it is equal to G_i/Γ_i for some Γ_i extending Γ_{i+1}. To see this, it is enough to construct Γ_i in the way described in the proof of Lemma 10.3.4. We choose $\sigma_{1,i}$ arbitrarily, and the involutory permutations τ are prescribed by chosen half-quotients replacing half-edges. It is easy to see that the resulting graph is the constructed H_i. We note that only the choices of τ matter, for arbitrary choices of $\sigma_{1,i}$ we get isomorphic quotients H_i.

On the other hand, if H_i is a quotient, it replaces the edges, loops and half-edges of H_{i+1} by some quotients, so we can generate H_i in this way. The reason is that according to Corollary 10.3.5, we can generate all Γ_i extending Γ_{i+1} by some choices $\sigma_{1,i}$ and τ. □

We say that two quotients H_i and H_i' extending H_{i+1} are *different* if there exists no isomorphism of H_i and H_i' which fixes the vertices and edges common with H_{i+1}. (But H_i and H_i' still might be isomorphic.) According to Lemma 10.2.2, the edge and loop-quotients are uniquely determined, so we are only free in choosing half-quotients. For non-isomorphic choices of half-quotients, we get different graphs H_i. For instance suppose that H_{i+1} contains a half-edge corresponding to the dipole from Fig. 10.9. Then in H_i we can replace this half-edge by one of the four possible half-quotients of this dipole.

Corollary 10.3.6. *If H_{i+1} contains no half-edge, then H_i is uniquely determined. Thus, for an odd order of Γ_r, the quotient H_r uniquely determines H_0.*

Proof. This is implied by Theorem 6.6.1 and Lemma 10.2.2 which states that edge- and loop-quotients are uniquely determined. If the order of Γ_r is odd, no half-edges are constructed in H_r, so no half-quotients ever appear. □

Half-quotients of Dipoles. In Lemma 10.2.3, we describe that a dipole A without colored edges can have at most $\left\lfloor \frac{e(A)}{2} \right\rfloor + 1$ pairwise non-isomorphic half-quotients. This statement can be easily altered to dipoles with colored edges which admit a much larger number of half-quotients:

Lemma 10.3.7. *Let A be a dipole with colored edges. Then the number of pairwise non-isomorphic half-quotients is bounded by $2^{\lfloor \frac{e(A)}{2} \rfloor}$ and this bound is achieved.*

Proof. Figure 10.9 shows an example. It can be easily generalized to exponentially many pairwise non-isomorphic quotients by introducing more pairs of halvable edges of additional colors. It remains to argue correctness of the upper bound.

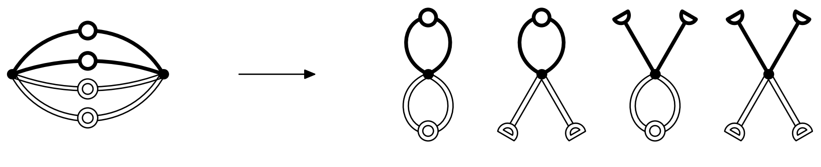


Figure 10.9: An example of a dipole with a pair of black halvable edges and a pair of white halvable edges. There exist four pairwise non-isomorphic half-quotiens.

First, we derive the structure of all involutory semiregular automorphisms τ acting on $\mathring{A}$. We have no freedom concerning the non-halvable edges of A: The undirected edges of each color class has to be paired by τ together. Further, each directed edge has to be paired with a directed edges of the opposite direction and the same color. It remains to describe possible action of τ on the remaining at most $\boldsymbol{e}(A)$ halvable edges of A. These edges belong to c color classes having $m_1, \dots, m_c$ edges. Each automorphism τ has to preserve the color classes, so it acts independently on each class.

We concentrate only on one color class having m_i edges. We bound the number $f(m_i)$ of pairwise non-isomorphic quotients of this class. Then we get the upper bound

$$\prod_{1 \le i \le c} f(m_i) \tag{10.3}$$

for the number of non-isomorphic half-quotients of A.

The rest of the proof is similar to the proof of Lemma 10.2.3. An edge e fixed in τ is mapped into a half-edge of the given color in the half-quotient $A/\langle\tau\rangle$. If τ maps e to $e' \ne e$, then we get a loop in the half-quotient $A/\langle\tau\rangle$. The resulting half-quotient only depends on the number of fixed edges and fixed two-cycles in the considered color class. We can construct at most $f(m_i) = \lfloor \frac{m_i}{2} \rfloor + 1$ pairwise non-isomorphic quotients, since we may have zero to $\lfloor \frac{m_i}{2} \rfloor$ loops with the complementing number of half-edges.

The bound (10.3) is maximized when each class contains exactly two edges. (Except for one class containing either three edges, or one edge if $\boldsymbol{e}(A)$ is odd.) $\square$

Assume that H_{i+1} contains a half-edge corresponding to a half-quotient of a dipole in H_i. By Theorem 6.6.1, the number of non-isomorphic expansions H_i of H_{i+1} can be exponential in the size difference of H_i and H_{i+1}.

The Block Structure of Quotients. We show how the block structure is preserved during expansions. A block atom A of G_i is always projected by an edge-projection, and so it corresponds to a block atom of H_i. Suppose that A is a proper atom or a dipole, and let $\partial A = \{u, v\}$.

- For an edge-projection $p|_A$, we get $p(u) \ne p(v)$, and $p(A)$ is isomorphic to an atom in H_i.
- For a loop- or a half-projection $p|_A$, we get $p(u) = p(v)$ and $p(u)$ is an articulation of H_i. If A is a dipole, then $p(A)$ is a pendant star of half-edges and loops attached to $p(u)$. By Lemma 7.4.4, if A is a proper atom, then $p(A)$ is either a path ending with a half-edge and with attached single pendant edges (when A^+ is essentially a cycle), or a pendant block with attached single pendant edges and half-edges (when A^+ is essentially 3-connected). (The reason is that the fiber of an articulation in a 2-fold cover is a 2-cut.)

Lemma 10.3.8. *The block structure of H_{i+1} is preserved in H_i, possibly with some new subtrees of blocks attached.*

Proof. By Theorem 6.6.1, edges inside blocks are replaced by edge-quotients of block atoms, proper atoms and dipoles which preserves 2-connectivity. New subtrees of blocks in H_i are created by replacing pendant edges with block atoms, loops by loop-quotients, and half-edges by half-quotients. $\square$

10.4 Quotients of Planar Graphs and Negami's Theorem

In this section, we show implication of our theory to planar graphs. We geometrically characterize the quotients of planar graphs which results in a direct proof of Negami's Theorem [288]. Using Theorem 6.6.1, it only remains to understand the quotients of planar primitive graphs and the half-quotients of planar proper atoms.

Recall from Section 8.1 that the automorphism groups of 3-connected planar graphs are spherical groups. The key point is that regular covering projections behave geometrically for them. Table 10.1 shows the number of conjugacy classes of subgroups of spherical groups $\mathbb{S}_4$, $\mathbb{C}_2 \times \mathbb{S}_4$ and $\mathbb{C}_2 \times \mathbb{A}_5$ which are isometry groups of platonic solids. Note that conjugate subgroups Γ determine isomorphic quotients G/Γ.

Geometry and Quotients. Recall from Section 8.1 that automorphisms of a 3-connected planar graph are either orientation preserving, or orientation reversing. This allows to use geometry to study regular quotients. Let τ be an orientation reversing involution of an orientable surface. The involution τ is called *antipodal* if it is a semiregular automorphism of a closed orientable surface S such that $S/\langle\tau\rangle$ is a non-orientable surface. Otherwise τ is called a *reflection.* A reflection of the sphere fixes a circle. An orientation reversing involution of a 3-connected planar graph is called *antipodal* if the respective isometry is antipodal and it is called a *reflection* if

$\mathbb{S}_4$ of the order 24			
Order	**Number**	**Order**	**Number**
1	1	6	1
2	2	8	1
3	1	12	1
4	3		

$\mathbb{C}_2 \times \mathbb{S}_4$ of the order 48			
Order	**Number**	**Order**	**Number**
1	1	8	7
2	5	12	2
3	1	16	1
4	9	24	3
6	3		

$\mathbb{C}_2 \times \mathbb{A}_5$ of the order 120			
Order	**Number**	**Order**	**Number**
1	1	8	1
2	3	10	3
3	1	12	2
4	3	20	1
5	1	24	1
6	3	60	1

Table 10.1: The number of conjugacy classes of the subgroups of the isometry groups of platonic solids.

the respective isometry is a *reflection.* A reflection of a map on the sphere fixes always either an edge, or a vertex.

The quotient of the sphere by an orientation preserving group of automorphisms is again the sphere. The half-quotient of the sphere by a reflection is the disk and and the half-quotient by an antipodal involution is the projective plane. See Fig. 10.10.

Quotients of Primitive Graphs. By Lemma 7.4.2, we know that every primitive graph G_r is either 3-connected with attached single pendant edges, or K_2 or C_n with attached single pendant edges. These attached single pendant edges only make $\mathrm{Aut}(G_r)$ smaller, which restricts the possible quotients. Therefore it is sufficient to understand how possible quotients can look for 3-connected planar graphs, K_2 and C_n.

Lemma 10.4.1 ([331])**.** *Let G be a 3-connected planar graph and Γ be a semiregular subgroup of* $\mathrm{Aut}(G)$*. There are three types of quotients of G:*

(a) Rotational quotients – *The action of Γ is orientation preserving and the quotient G/Γ is planar.*
(b) Reflectional quotients – *The action of Γ is orientation reversing but does not contain an antipodal involution. Then the quotient G/Γ is planar and necessarily contains half-edges.*
(c) Antipodal quotients – *The action of Γ is orientation reversing and contains an antipodal involution. Then G/Γ is projective planar.*

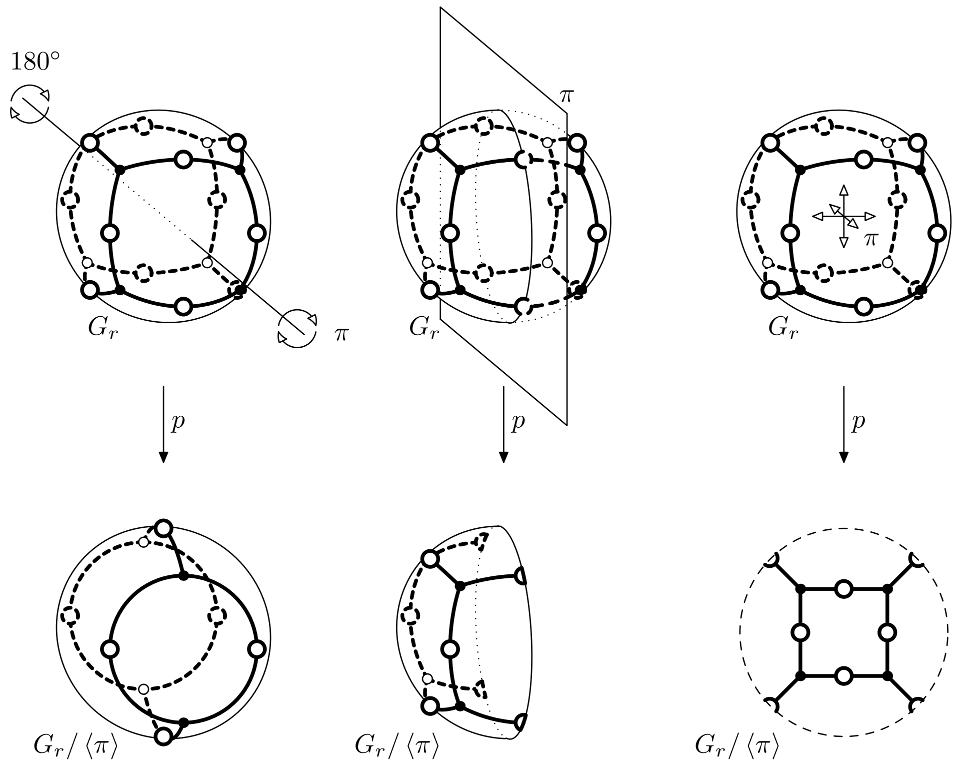


Figure 10.10: From left to right, a rotational quotient, a reflectional quotient and an antipodal quotient of the cube; also see Fig. 10.2.

Figure 10.10 shows examples of these types of quotients. We note that an antipodal quotient can be planar, but not necessarily; for an example, see Fig. 6.7.

The quotients of K_2 are straightforward. Next, we characterize quotients of cycles, which completes the description of possible quotients of primitive graphs:

Lemma 10.4.2. *Let Γ be a semiregular subgroup of* $\mathrm{Aut}(C_n)$. *Then C_n/Γ is either a cycle, or a path with two half-edges attached to its ends (only for n even).*

Half-quotients of Proper Atoms. Next, we characterize the half-quotients of planar proper atoms. There are further restrictions compared to the quotients of primitive graphs since the involution has to exchange the vertices of the boundary:

Lemma 10.4.3. *Let A be a planar proper atom and let $\partial A = \{u, v\}$. There are at most two half-quotients $A/\langle\tau\rangle$ where $\tau \in \mathrm{Aut}_{\partial A}(A)$ is an involutory semiregular automorphism transposing u and v:*

(a) The rotational half-quotient – *The involution τ is orientation preserving and $A/\langle\tau\rangle$ is planar with at most one half-edge.*
(b) The reflectional half-quotient – *The involution τ is a reflection and $A/\langle\tau\rangle$ is planar with at least two half-edges.*

Proof. The graph A^+ (obtained from A by adding the edge uv) is an essentially 3-connected planar graph with a unique embedding into the sphere. By Lemma 8.1.4c, $\mathrm{Aut}_{\partial A}(A)$ is a subgroup of $\mathbb{C}_2^2$. An involution τ exchanging u and v corresponds to a

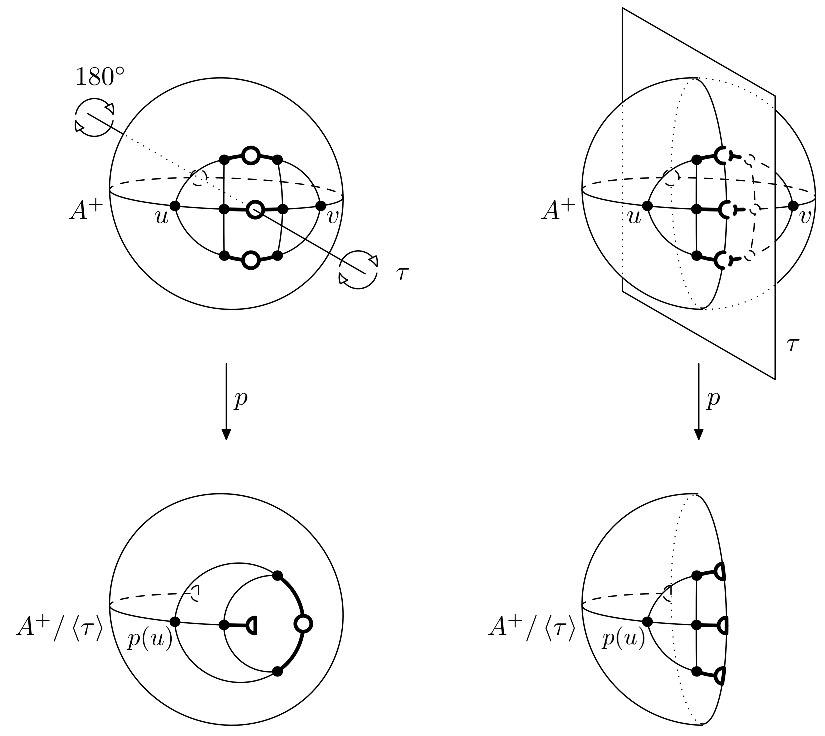


Figure 10.11: The rotational quotient and reflectional quotient of a planar proper atom A with the added edge uv.

map automorphism of A^+ fixing uv. Either τ is a 180° rotation around the centre of uv which gives the rotational half-quotient, or it is a reflection which gives the reflectional half-quotient; see Fig. 10.11. According to Lemma 10.4.1, both possible half-quotients are planar. $\square$

Direct Proof of Negami's Theorem. Using the above statements, we give a direct proof of Negami's Theorem. This theorem states that a graph H has a finite planar regular cover G (i.e, $G/\Gamma \cong H$ for some semiregular $\Gamma \leq \mathrm{Aut}(G)$), if and only if H is projective planar. For a given projective planar graph H, the construction of a planar graph G is easy: by embedding H into the projective plane and taking the double cover of this embedding, we get the graph G embedded to the sphere. Below, we prove the harder implication:

Theorem 10.4.4 (Negami [288]). *Let G be a planar graph. Then every (regular) quotient of G is projective planar.*

Proof. We apply the reduction series on G which produces graphs $G = G_0, G_1, \dots, G_r$ such that G_r is primitive. If G_r is essentially 3-connected, then by Lemma 10.4.1 every quotient $H_r = G_r/\Gamma_r$ is projective planar. If G_r is K_2 or C_n with single pendant edges attached, then by Lemma 10.4.2 every quotient $H_r = G_r/\Gamma_r$ is even planar.

By Theorem 6.6.1, every quotient $H = G/\Gamma$ can be constructed from some H_r by an expansion series in which we replace edges, loops and half-edges by edge-quotients, loop-quotients and half-quotients, respectively. All edge- and loop-quotients are clearly planar. By Lemma 10.4.3, every half-quotient of a proper atom is planar, and by Lemma 10.3.7 every half-quotient of a dipole is a set of loops and half-edges attached to a single vertex, which is also planar. Therefore, these replacements can be done in a way that the underlying surface of H_r is not changed, so H is also projective planar. $\square$

We note that deciding whether $H = G/\Gamma$ is planar or non-planar projective is done on the primitive graph G_r. It is non-planar if and only if Γ_r contains a semiregular antipodal involution and the resulting quotient $H_r = G_r/\Gamma_r$ is non-planar.

10.5 Concluding Remarks

We recall the main points addressed in this chapter:

- We show that the 3-connected reduction described in Chapter 7 behave well with respect to semiregular subgroups and regular graph covering. For a prescribed quotient $H_r = G_r/\Gamma_r$, we describe all possible expansions $H_0 = G_0/\Gamma_0$ which revert the reductions. Theorem 6.6.1 states that every quotient $H \cong G/\Gamma$ can be obtained in this way, and different quotients H_0 are constructed by non-isomorphic quotients H_r and non-isomorphic choices of half-quotients in the expansions.

- Since the quotients of 3-connected planar graphs can be understood using geometry, we give a direct proof of Negami's Theorem [288] (Theorem 10.4.4). The reason is that a quotient $H_r = G_r/\Gamma_r$ is due to geometry always planar or projective planar. By Theorem 6.6.1, the expansions create H from H_r while preserving the underlying surface of H_r.
- Our results have as well algorithmic implications for regular covering testing, described in Chapter 11.

More General Graphs. Our structural results also work for more general graphs. We have assumed that the graphs G and H are without loops and free half-edges. We can work with loops and half-edges in G in the same way as with pendant edges (of different colors). Since we assume that H contains no half-edges, we set the reductions and expansions in the way that half-edges can appear in the expansion series but no expanded quotient H_0 contains half-edges. This is done by having all edges of G_0 as undirected edges. To admit quotients H_0 with half-edges, it is sufficient to change all edges of G_0 to halvable edges. Also, all the results can be used when G and H contain colored edges, vertices, some edges oriented, etc.

Harmonic Regular Covers. There is a generalization of regular graph covering for which it would be interesting to find out whether our techniques can be modified. Consider geometric regular covers of surfaces, like in Fig. 10.10 and 10.11. The orbits of the 180° rotations are of size two, with the exception of two points lying on the axis of the rotation. These exceptional points are called *branch points*. In general, a regular covering projection is locally homeomorphic around a branch point to the complex mapping $z \mapsto z^\ell$ for some integer $\ell \le k$, and ℓ is called the *order* of the branch point.

Assume that G is a 3-connected planar graph embedded onto the sphere, $\Gamma \le \mathrm{Aut}(G)$ is a semiregular subgroup of automorphisms of the sphere, and $p : G \to H = G/\Gamma$ is the regular covering projection. When H is a standard graph (with no free half-edges), all branch points of p belong to faces of the embedding. If a branch point (of order two) is placed in the center of an edge of G, this edge is projected to a half-edge in H. It is possible to consider covering projections between surfaces in which branch points can be placed in vertices of G which gives *harmonic regular covering* [17]. If a branch point of order ℓ is placed in a vertex $v \in \boldsymbol{V}(G)$, then the vertex $p(v) \in \boldsymbol{V}(H)$ has the degree equal $\deg v/\ell$ and for an edge $e \in \boldsymbol{E}(H)$ incident with $p(v)$, the fiber $p^{-1}(e)$ has exactly ℓ edges incident with v.

11 Algorithmic Aspects of Regular Graph Covers

This chapter contains:

- *11.1: Complexity of Regular Graph Covering.* We prove that RegularCover belongs to NP and it is GI-hard.
- *11.2: Atoms, Reduction and Expansion.* We describe how to compute symmetry types and quotients of atoms.
- *11.3: Meta-algorithm.* We describe the FPT algorithm solving RegularCover for graph classes satisfying (P1), (P2), and (P3). It is an involved dynamic programming on the reduction tree of H using two subroutines: a generalization of the perfect matching problem (the bottleneck) and ListIso (fast).
- *11.4: Star Blocks Atoms with Lists.* We discuss details concerning the only slow subroutine in the meta-algorithm.
- *11.5: Applying the Meta-algorithm to Planar Graphs.* We show that planar graphs satisfy (P1), (P2), and (P3).
- *11.6: Concluding Remarks.* We describe the time analysis of the algorithm for regular covering testing of planar graphs and discuss open problems.

`http://pavel.klavik.cz/orgpad/regular_covers.html`

11.1 Complexity of Regular Graph Covering

We establish fundamental complexity properties of regular covering. Our goal is to highlight similarities with the graph isomorphism problem.

Belonging to NP. The H-Cover problem clearly belongs to NP since one can just test in polynomial time whether a given mapping is a locally bijective homomorphism. Not so obviously, the same holds for RegularCover:

Lemma 11.1.1. *The* RegularCover *problem belongs to* NP.

Proof. By definition, G regularly covers H if and only if there exists a semiregular subgroup Γ of $\mathrm{Aut}(G)$ such that $G/\Gamma \cong H$. As a certificate, we give k permutations, one for each element of Γ, and an isomorphism between G/Γ and H. We check that these permutations define a group Γ acting semiregularly on G. The given isomorphism allows to check that the constructed G/Γ is isomorphic to H. Clearly, this certificate is polynomially large and can be verified in polynomial time. □

One can prove even a stronger result:

Lemma 11.1.2. *For a mapping $p : G \to H$, we can test whether it is a regular covering projection in polynomial time.*

Proof. Testing whether p is a covering projection can clearly be done in polynomial time. It remains to test regularity. Choose an arbitrary spanning tree T of H. Since p is a covering, then $p^{-1}(T)$ is a disjoint union of k isomorphic copies $T_1, \dots, T_k$ of T. We number the vertices of the fibers according to the spanning trees, i.e., $p^{-1}(v) = \{v_1, \dots, v_k\}$ such that $v_i \in T_i$. This induces a numbering of the half-edges of each fiber over a half-edge of $\boldsymbol{H}(H)$, following the incidences between half-edges and vertices. For every half-edge $h \notin \boldsymbol{H}(T)$, we define the permutation σ_h of $\{1, \dots, k\}$ taking i to j if there is a half-edge $h' \in p^{-1}(h)$ incident with a vertex of T_i and paired with a half-edge incident with a vertex of T_j.

Let Θ be the group generated by all σ_h, where $h \notin \boldsymbol{H}(T)$. We assume that G is connected. By Orbit-Stabilizer Theorem, we have $|\Theta| = |\Theta_v| \cdot |[v]|$, and from the connectivity, it follows that $|[v]| = k$. Therefore, the action of Θ is regular if and only if $|\Theta| = k$ which can be checked in polynomial time. □

The constructed permutations σ_h associated with p are known in the literature [165] as *permutation voltage assigments* associated with p.

GI-hardness. When $k = |G|/|H| = 1$, the problem RegularCover exactly corresponds to GraphIso. Let GI be the class of decision problems polynomial-time reducible to GraphIso. Bodlaender [36] proved the following for general covers (and his reduction works for regular covers as well):

Lemma 11.1.3. *For every fixed k, the k-*FoldCover *and k-*FoldRegularCover *problems are* GI*-hard.*

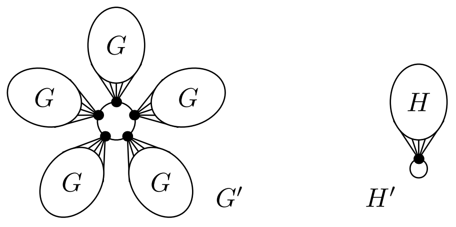


Figure 11.1: The graph G' is constructed by k copies of G with attached universal vertices connected into a cycle while H' is constructed by attaching a universal vertex with a loop to H.

Proof. For input graphs G and H of the graph isomorphism problem, we construct the graphs G' and H' depicted in Fig. 11.1. The reduction works since the universal vertices in G' must be mapped to the universal vertex in H' (since covering projection preserves degrees). Therefore, G' (regularly) covers H' if and only if $G \cong H$. $\square$

11.2 Atoms, Reduction and Expansion

Recall algorithmic results from Section 7.7. We show that we can algorithmically compute with new definitions of Section 10.2.

Computing Symmetry Types of Atoms.

Lemma 11.2.1. *For a dipole A, we can determine its symmetry type (halvable, symmetric, asymmetric) in polynomial time.*

Proof. For a dipole A, we can determine in polynomial time using Lemma 7.7.4 whether it is asymmetric or not. It is halvable if and only if it is balanced and it has an even number of undirected edges, which can be easily tested in polynomial time as well. $\square$

Lemma 11.2.2. *For a proper atom A of $\mathcal{C}$ satisfying (P1), (P2), and (P3*), we can determine its symmetry type (halvable, symmetric, asymmetric) in polynomial time.*

Proof. Let $\partial A = \{u, v\}$. By Lemma 7.7.5, we can determine using (P1) and (P3*) whether A is asymmetric or not. If it is not asymmetric, we use (P2) to generate polynomially many semiregular involutions of order two acting on B. For each semiregular involution, we check whether it transposes u to v, and whether it preserves the colors of $\boldsymbol{V}(B)$ coding pendant edges. If such a semiregular involution exists, then A is halvable, otherwise it is just symmetric. $\square$

Computing Half-quotients of Proper Atoms.

Lemma 11.2.3. *Let A be a proper atom of $\mathcal{C}$ satisfying (P1) and (P2). Then there are polynomially many non-isomorphic half-quotients of A which can be computed in polynomial time.*

Proof. By Lemma 7.4.4, A^+ is either essentially a cycle (where it holds trivially), or it is an essentially 3-connected graph. We construct B^+ from A^+ be replacing pendant edges with colored vertices, by (P1) both A^+ and B^+ belong to $\mathcal{C}$. According to (P2), the number of different semiregular subgroups of order two is polynomial in the size of B^+. Each half-quotient is defined by one of these semiregular involutions which fixes the edge uv, transposes u and v, and preserves colors. □

Computing Reduction Series and Reduction Tree with Halvable Edges.

Lemma 11.2.4. *If a graph G belongs to $\mathcal{C}$ satisfying (P1), (P2) and (P3*), then the reductions series $G = G_0, \dots, G_r$ and the reduction tree can be computed in polynomial time.*

Proof. We work exactly as in Lemma 7.7.6 but we use Lemma 11.2.2 instead of Lemma 7.7.5. □

11.3 Meta-algorithm

In this section, we establish the meta-algorithm from Theorem 6.6.3, solving RegularCover for G belonging to $\mathcal{C}$ satisfying (P1) to (P3) in time $\mathcal{O}^*(2^{e(H)/2})$.

Let $k = |G|/|H|$, and we assume that $k \geq 2$. (If k is not an integer, then clearly G does not cover H. If $k = 1$, then it is equivalent to the graph isomorphism problem and we can test it using Lemma 7.7.7.) The algorithm consists of the following major parts:

1. *Reduction Part:* We construct the reduction series for $G = G_0, \dots, G_r$ terminating with the unique primitive graph G_r. Throughout the reduction the central block is preserved, otherwise according to Lemma 10.3.2 there exists no semiregular automorphism of G and we output "no". According to (P1), the reduction preserves the class $\mathcal{C}$, and also every atom belongs to $\mathcal{C}$.
2. *Quotient Part:* Using (P2), we construct the list of all subgroups Γ_r of $\mathrm{Aut}(G_r)$ of the order k acting semiregularly on G_r. The number of subgroups in the list is polynomially large by (P2).
3. *Expansion Part:* For each Γ_r in the list, we compute $H_r = G_r/\Gamma_r$. We say that a graph H_r is *expandable* if there exists a sequence of extensions repeatedly applying Theorem 6.6.1 which constructs H_0 isomorphic to H. We test the expandability of H_r using dynamic programming while using (P3).

It remains to explain details of Expansion Part, and prove the correctness of the algorithm.

Outline. In Section 11.3.1, we give an overview of Expansion Part. In Section 11.3.2, we describe a catalog which stores all atoms and their quotients discovered during reductions. In Section 11.3.3, we describe reductions with lists, used in expandibility testing. Last, in Section 11.3.4, we conclude with a proof of Theorem 6.6.3.

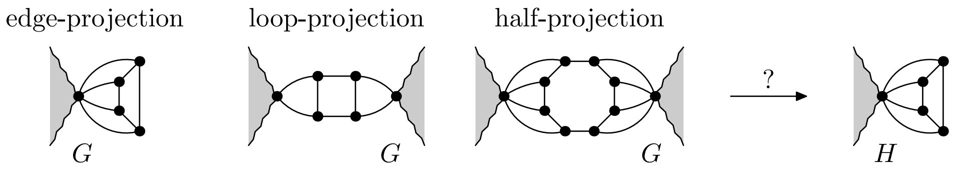


Figure 11.2: For a pendant block of H, there are three possible preimages in G. It could be a block atom mapped by the edge-projection, or a proper atom mapped by the loop-projection, or another proper atom mapped by a half-projection (where the half-quotient is created by 180° rotation τ).

11.3.1 Overview of Testing Expandability

In this section, we explain how to test expandability of H_r. We start by illustrating the fundamental difficulty, for simplicity on pendant blocks. Suppose that H has a pendant block as in Fig. 11.2. From the local information, there is no way to know whether this block corresponds in G to the edge-quotient of a block atom, or to the loop-quotients of some proper atoms, or to half-quotients of some other proper atoms. It can easily happen that the all these atoms appear in G. So without exploiting some additional information from H, there is no way to know what is the preimage of this pendant block.

In our approach, we revert the problem of expandability of H_r by reducing H towards H_r. But since it is not clear which atoms of H correspond to which parts of G, we do not decide it during the reductions, instead we just remember lists of all possibilities. The dynamic programming deals with these lists and computes further lists for larger parts of H. Figure 11.3 illustrates the overview of our algorithm.

Reductions of Quotients and Cores. Notice that H_r might not be primitive; see Fig. 11.4 for an example. It would be difficult to match it to a reduction series in H, so in Step 3, we further reduce H_r to a primitive graph H_s.

We define atoms in the quotient graphs similarly as in Section 7.4 with only one difference. We choose one arbitrary block/articulation called the *core* in H_r; for instance, we can choose the central block/articulation. The core plays the role of the

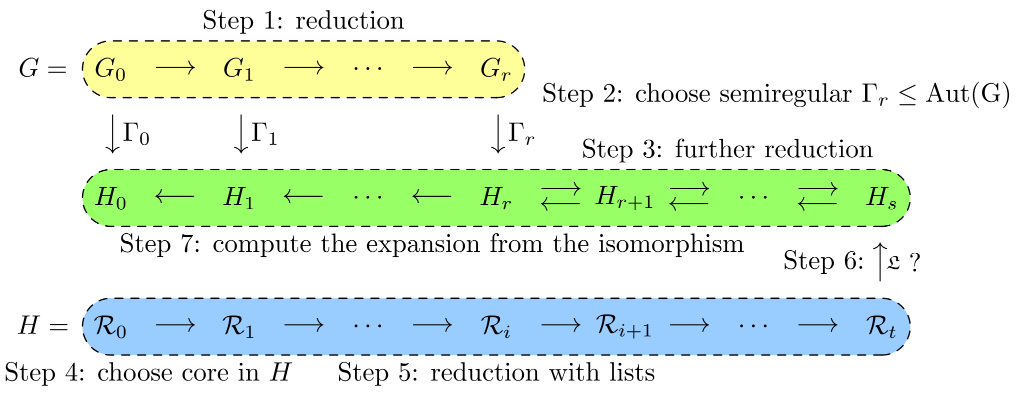

Step 1: reduction
$G = G_0 \longrightarrow G_1 \longrightarrow \cdots \longrightarrow G_r$
Step 2: choose semiregular $\Gamma_r \leq \mathrm{Aut(G)}$
Γ_0
Γ_1
Γ_r
Step 3: further reduction
$H_0 \longleftarrow H_1 \longleftarrow \cdots \longleftarrow H_r \rightleftarrows H_{r+1} \rightleftarrows \cdots \rightleftarrows H_s$
Step 7: compute the expansion from the isomorphism
Step 6: $\cong$?
$H = \mathcal{R}_0 \longrightarrow \mathcal{R}_1 \longrightarrow \cdots \longrightarrow \mathcal{R}_i \longrightarrow \mathcal{R}_{i+1} \longrightarrow \cdots \longrightarrow \mathcal{R}_t$
Step 4: choose core in H
Step 5: reduction with lists

Figure 11.3: The metaalgorithm proceeds in the following seven steps. We iterate over all possible choices in Steps 2 and 4.

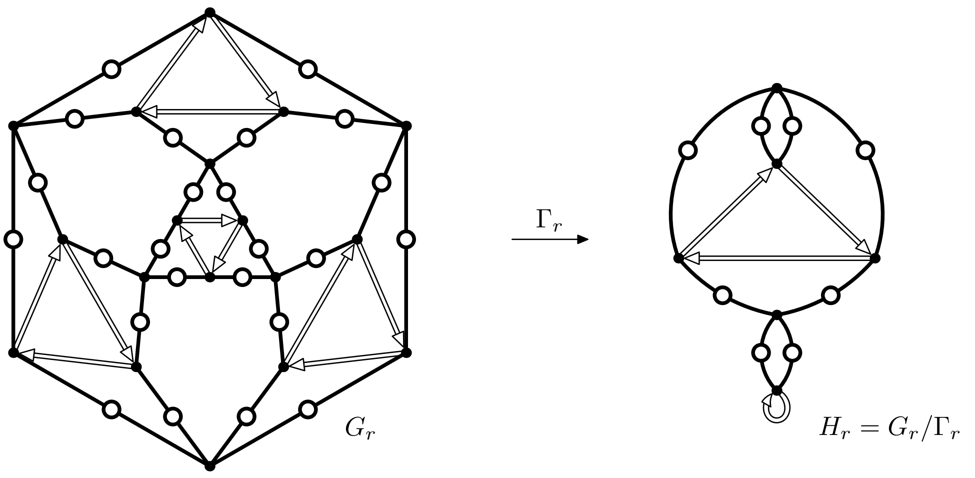


Figure 11.4: A primitive graph G_r which is 3-connected. Let Γ_r be the semiregular subgroup of $\mathrm{Aut}(G_r)$ generated by a 120° rotation. It defines the quotient $H_r = G_r/\Gamma_r$ which is not primitive (contains articulations and 2-cuts).

central block in the definition of parts and atoms. Also, in the definition we consider half-edges and loops as pendant edges, so they do not form block atoms. We proceed with the reductions in H_r further till we obtain a primitive quotient graph H_s, for some $s \geq r$; see Fig. 11.5.

Let $H_0, \dots, H_{s-1}$ be the graphs obtained by an expansion series of H_s using Theorem 6.6.1.

Lemma 11.3.1. *The graph H_s is expandable to H, if and only if H_r is expandable to H.*

Proof. It follows from the fact that the graphs $H_{s-1}, \dots, H_r$ are uniquelly determined, since no half-edges are expanded till H_r. □

Lemma 11.3.2. *If H_s is expandable to H, then the core of H_s is expanded to some block or articulation of H.*

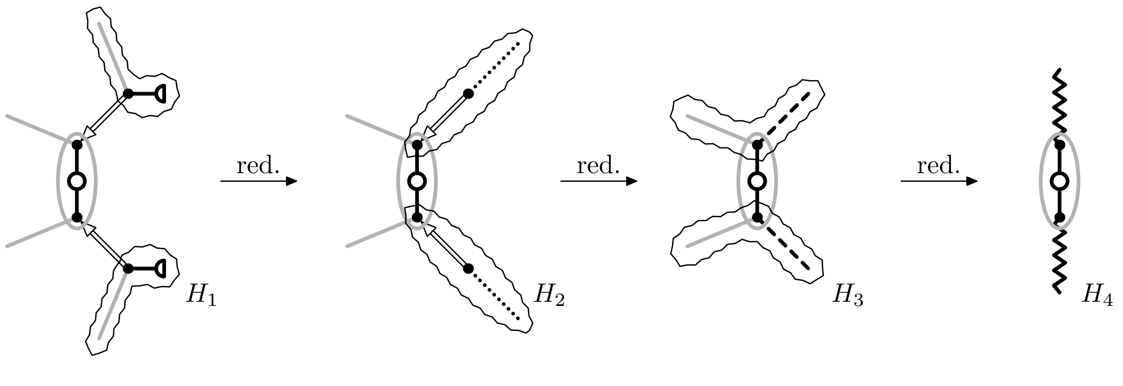


Figure 11.5: The graph H_1 is one quotient of G_1 from Fig. 7.14. We further reduce it to H_3 with respect to the core block depicted in gray. Notice that H_1 and H_2 only contain block atoms.

Proof. The graph H_s consists of the core together with some pendant edges, loops and half-edges. By Lemma 10.3.8, the core is preserved as an articulation/block in all graphs $H_s, \dots, H_0$. The core can be only changed by replacing of its colored edges by edge-quotients. Since H_s is expandable to H_0, the expanded core is isomorphic to some block or articulation of H. □

In Step 4, we test all possible positions of the core in H. (We have $\mathcal{O}(n)$ possibilities, so we run the dynamic programming algorithm multiple times.) In what follows, we have the core fixed in H as well.

In Step 5, we apply on H reductions with lists, described in Section 11.3.3. In Step 6, we test whether some choices from these lists are compatible with the graph H_s.

11.3.2 Catalog of Atoms

During the reduction phase of the algorithm, we construct the following *catalog of atoms* forming a database of all discovered atoms. These atoms arise in three ways: atoms of $G_0, \dots, G_{r-1}$, atoms in half-quotients of these atoms, and atoms in the reductions of the quotients $H_r = G_r/\Gamma_r$. We are not very concerned with a specific implementation of the algorithm, so the purpose of this catalog is to simplify description.

For each isomorphism class of atoms represented by an atom A, we store the following information in the catalog:

- The atom A.
- The corresponding colored edge of a given type representing the atom in the reduction.
- If A is an atom of $G_0, \dots, G_{r-1}$, the unique edge- and loop-quotients of A and information about its half-quotients.

For an overview of adding an atom A into the catalog, see Algorithm 4.

Storing Star Block Atoms. Let A be a star block atom. We store it in the catalog *partially expanded* which works as follows. By the definition, A consists of a vertex with attached edges, loops and half-edges. If some edge corresponds to a star block atom S, we replace it with the edges of S. Similarly, if some loop corresponds to the loop-quotient Q of a dipole D, we replace it by the loops of Q. We repeat this till all pendant edges of A correspond to block atoms and all loops of A correspond to loop-quotients of proper atoms. On the other hand, the half-edges of A may correspond to half-quotients of both proper atoms and dipoles.

Storing Dipoles. Let A be a dipole in $G_0, \dots, G_{r-1}$. By Lemma 10.3.7, it can have exponentially many non-isomorphic half-quotients. On the other hand, they are well described in Section 10.3, so we can generate all of them from the dipole when needed.

We store this dipole A in the catalog *partially expanded* which works as follows. Almost all edges of A correspond to proper atoms, while at most one edge corresponds to a dipole D. (At most one since from the definition, a dipole D with $\partial D = \{u, v\}$ consists of all edges between u and v.) If one edge corresponds to D, we replace it in A with the edges of the dipole D. And if one of these edges of D again correspond to some dipole D', we proceed further with the expansion.

Notice that by the definition of the reduction, all colored edges of D have different colors than the edges of A. Therefore the half-quotients of the original dipole A are exactly the same as the half-quotients of the partially expanded dipole A. The reason for this expansion is that every half-quotient of the partially expanded dipole A consists of loops and half-edges attached to one vertex, where each loop and each half-edge is expanded into one block (with attached single pendant edges, half-edges and loops).

Algorithm 4: The subroutine for adding an atom into the catalog

Require: An atom A.

Ensure: If A is not contained in the catalog, then it is added. A colored halvable/undirected/directed edge corresponding to A is given.

```
 1: if $A$ is a star block atom then
 2:    while $A$ contains a pendant edge $e'$ of a star block atom $S$ do
 3:       Replace $e'$ with the edges of $S$.
 4:    while $A$ contains a loop $e'$ of the loop-quotient of a dipole $D$ do
 5:       Replace $e'$ with the loops of the loop-quotient of $D$.
 6: if $A$ is a dipole then
 7:    while $A$ contains an edge $e'$ corresponding to a dipole $D$ do
 8:       Replace $e'$ with the edges of $D$.
 9: We test whether $A$ is contained in the catalog using Lemma 11.3.4.
10: if $A$ is contained in the catalog then
11:    return The corresponding colored edge representing $A$.
12: We determine the symmetry type of $A$ using Lemmas 7.7.4 and 7.7.5.
13: We assign an edge $e$ of a new color of the corresponding type to $A$.
14: if $A$ is an atom of $G_0, \dots, G_r$. then
15:    We compute the edge-quotient of $A$ and the loop-quotient of $A$ (if $A$ is not a
       block atom).
16:    if $A$ is a dipole consisting of exactly two halvable edges of the same color then
17:       We add the half-quotient of $A$ with one loop to the list of half-quotients.
18:    if $A$ is a halvable proper atom then
19:       We compute all half-quotients $Q$ of $A$ by Lemma 11.2.3.
20:       for each half-quotient $Q$ do
21:          Apply the reduction series on $Q$ with respect to the block containing $\partial Q$,
             constructing a primitive graph $Q'$.
22:          Add all detected atoms to the catalog and replace them by the correspond-
             ing colored edges.
23:          Add $Q'$ to the catalog, as a half-quotient of $A$.
24: return The assigned colored edge $e$ corresponding to $A$.
```

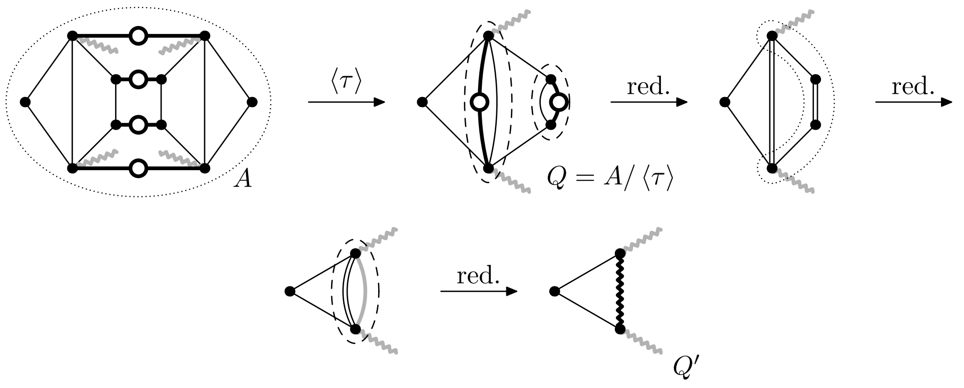


Figure 11.6: A proper atom A with a half-quotient Q generated by 180° rotation τ. A reduction series is applied on Q which adds further atoms to the catalog and the primitive graph Q'.

Further, if a halvable dipole A consists of exactly two edges of the same color, we compute its half-quotient consisting of just the single loop attached, and we add this quotient to the catalog. The reason is that this quotient behaves exactly as the loop-quotient of some proper atom.

Storing Proper Atoms. If A is not a dipole, we compute the list of all its pairwise non-isomorphic half-quotients, and store them in the catalog in the following way. A half-quotient Q of A might not be primitive. Therefore, we apply a reduction series on Q, and add all atoms discovered by the reduction to the catalog. (We do not compute their half-quotients. They are never realized unless these atoms are directly found in G as well.) When the reduction series finishes, this half-quotient is reduced to a primitive graph Q'. Naturally, the block containing ∂Q, being a single vertex of the half-quotient, behaves like the central block in the definition of atoms, i.e., it is never reduced. The reduced half-quotient Q' is either essentially 3-connected, a cycle with attached single pendant edges, or K_2 with a single pendant edge or half-edge attached. See Fig. 11.6 for an example.

Total Size of Catalog. Next, we prove that the catalog is not too large.

Lemma 11.3.3. *Assuming (P2), the catalog contains polynomially many atoms and half-quotients.*

Proof. First we deal with the number of atoms in $G_0, \dots, G_r$. Notice that by replacing an interior of an atom, the total number of vertices and edges is decreased; the interiors of atoms in each G_i contain at least two vertices and edges in total and are pairwise disjoint (see Lemma 7.4.7). Thus we add a linear number of atoms of $G_0, \dots, G_r$ to the catalog, of total linear size.

By (P2), there are polynomially many possible quotients H_r, in each we encounter linearly many atoms when reducing to H_s. So we add polynomially many atoms to the catalog.

By (P2), each proper atom A has polynomially many half-quotients, for different semiregular involutions of $\mathrm{Aut}(A)$. So, we have in total polynomially many half-quotients, each containing at most linearly many atoms in its reduction series. And by Lemma 10.2.2 we have the unique edge- and loop-quotient. So again, the total number of atoms and quotients added to the catalog is polynomial. □

Catalog Queries. Throughout the algorithm, we repeatedly ask *queries* whether some atom or some of its quotients is contained in the catalog, and if so, we retrieve the corresponding colored edge/loop/half-edge.

Lemma 11.3.4. *Assuming (P3*), each catalog query can be answered in polynomial time.*

Proof. By Lemma 11.3.3, we need to test graph isomorphism for the input atom/quotient and polynomially many atoms/quotients in the catalog. If the input is an atom of an edge-quotient, we use Lemma 7.7.3. If it is a loop- or a half-quotient, then it is a primitive graph and we use Lemma 7.7.2. □

11.3.3 Reductions with Lists

In this section, we describe Steps 5 and 6 of the diagram in Fig. 11.3. By Lemma 11.3.1, we need to test whether H_s is expandible to H. We approach this in the opposite way, by applying a reduction series on H with respect to the core defining $\mathcal{H}_0, \dots, \mathcal{H}_t$. As already discussed in Section 11.3.1, we do not know which parts of G project to different parts of H. Therefore each $\mathcal{H}_i$ is a set of graphs, and $\mathcal{H}_t$ is a set of primitive graphs. We then determine expandability of H_s by testing whether $H_s \in \mathcal{H}_t$.

Since each set $\mathcal{H}_i$ can contain a huge number of graphs, we represent it implicitly in the following manner. Each $\mathcal{H}_i$ is represented by one graph $\mathcal{R}_i$ with some colored edges and with so-called pendant elements attached to some vertices.

Pendant Elements with Lists. A pendant element x in $\mathcal{R}_i$ corresponds to a block atom in $\mathcal{R}_j$ for some $j < i$, which is reduced in $\mathcal{R}_{j+1}$. When pendant elements are fully expanded, they correspond to block part of H with pairwise disjoint interiors. We use the name pendant element since it may represent a pendant edge of some color, several loops of some other colors, and several half-edges of some other colors.

Each pendant element x is equipped with a *list* $\mathfrak{L}(x)$ whose *members* are possible realizations of the corresponding block atom by the quotients from the catalog. Each graph of $\mathcal{H}_i$ is created for $\mathcal{R}_i$ by replacing the pendant elements by some choices of edges, loops and half-edges from their lists. The list $\mathfrak{L}(x)$ of a pendant element x contains an edge/loop/half-edge if and only if it is possible to expand this edge/loop/half-edge to the graph isomorphic to the block part corresponding to x. For an example, see Fig. 11.7. According to Lemma 11.3.3, we have polynomially many atoms, and so the size of each list is polynomial in size.

Lemma 11.3.5. *Each list $\mathfrak{L}(x)$ contains at most one edge. Further, if two lists share an edge or a loop, their pendant elements correspond to isomorphic block parts in H.*

Proof. Two atoms have the isomorphic edge-quotients if and only if they are isomorphic. Therefore each list $\mathfrak{L}(x)$ contains at most one edge.

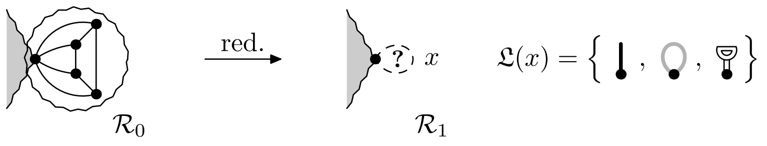


Figure 11.7: Let x be the pendant element corresponding to the pendant block of H depicted in Fig. 11.2. Then $\mathfrak{L}(x)$ contains three different members if all three atoms depicted in Fig. 11.2 are contained in the catalog.

If a pendant element x is fully expanded, it corresponds to one block part of H. Suppose that an edge- or a loop-quotient belongs to $\mathfrak{L}(x)$. If it is fully expanded, then it has to be isomorphic to this block part. But according to Lemma 10.2.2, the expansions of edge- and loop-quotients are deterministic since half-edges are never encountered. Therefore the corresponding block part in H is uniquely determined. □

One list may contain several loops, for which identifying of the vertices of the boundaries constructs identical graphs; see Fig. 11.8. Similarly, a list may contain several half-edges; see Fig. 11.9. Because of the second part of Lemma 11.3.5, the loops pose no problem. On the other hand, one half-edge may be contained in lists of several different pendant elements which are expanded to non-isomorphic subgraphs in H; see Fig. 11.8. This creates the main difficulty for our algorithm, leading to the bottleneck in form of a slow subroutine requiring time $\mathcal{O}^*(2^{e(H)/2})$.

Reductions with Lists. We want to compute the reduction series with lists $H = \mathcal{R}_0, \mathcal{R}_1, \dots, \mathcal{R}_t$ ending with a primitive graph $\mathcal{R}_t$ with attached pendant elements with computed lists. We construct $\mathcal{R}_0$ by replacing all pendant edges and loops by pendant elements with singleton lists.

Suppose that we know $\mathcal{R}_i$, and we want to apply one step of the reduction and

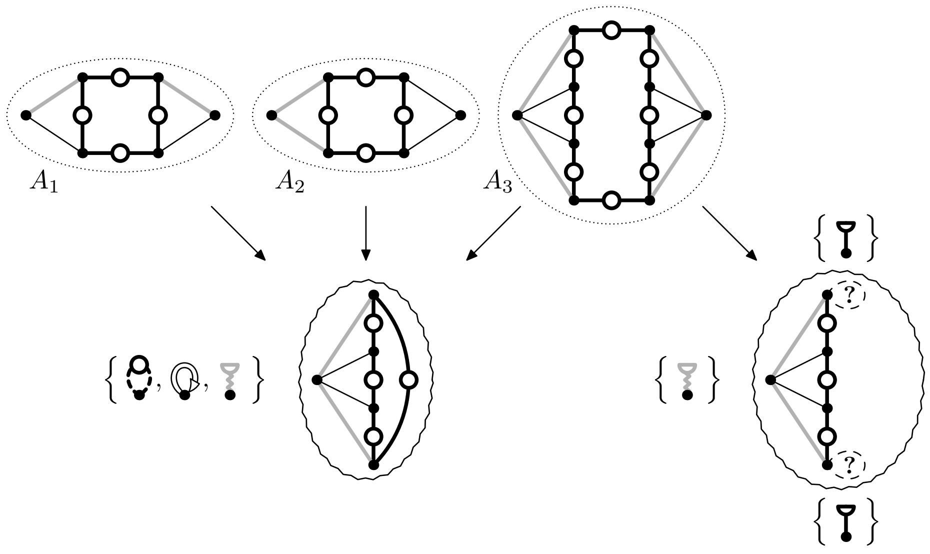


Figure 11.8: Two block atoms corresponding to pendant elements with depicted lists. On the left, the list has the loops corresponding to A_1 and A_2 and the half-edge corresponding to A_3. On the right, the list only contains the half-edge of A_3.

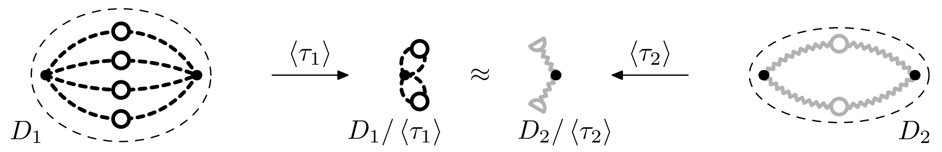


Figure 11.9: An example of two dipoles D_1 and D_2 having isomorphic half-quotients. Consider the atoms A_1 and A_3 from Fig. 11.8, for which the loop-quotient of A_1 is isomorphic to a half-quotient of A_3. Let D_1 consist of four edges corresponding to A_1 and let D_2 consist of two edges corresponding to A_3. Then the half-quotient $D_1/\langle\tau_1\rangle$ can be expanded to a graph isomorphic to an expansion of the half-quotient $D_2/\langle\tau_2\rangle$.

compute $\mathcal{R}_{i+1}$. We find all atoms in $\mathcal{R}_i$. We define atoms with respect to the chosen core in H, and we work with pendant elements as with pendant edges. Further, we consider only star block atoms consisting only of an articulation with all its descendants attached in form of reduced pendant elements. This means that we postpone reduction of star block atoms till all their descendants are reduced first. (The same modification could be applied in all reductions as well, but it is important here.)

To construct $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$, we proceed with the following:

- We replace dipoles and proper atoms by edges of the corresponding colors from the catalog. A proper atom might have pendant elements attached to its interior, but these pendant elements are always realized by edges corresponding to the edge-quotients of some block atoms. Therefore, we can replace the pendant elements with lists by the unique edges from these lists, and if some list contains no edge, we stop the reduction. We run a catalog query and if the dipole or proper atom is not contained in the catalog, we halt the reduction procedure.
- We replace block atoms by pendant elements with constructed lists. If some list is empty, we again halt the reduction.

It remains to describe the construction of the lists for the created pendant elements.

Computing Lists. Let A be a block atom in $\mathcal{R}_i$, replaced by a pendant element x in $\mathcal{R}_{i+1}$, and we want to compute $\mathfrak{L}(x)$. We compute $\mathfrak{L}(x)$ from the lists of the pendant elements attached to A. Suppose that A has pendant elements $y_1, \dots, y_p$ attached. For each member of $\mathfrak{L}(x)$, we remember which members of $\mathfrak{L}(y_1), \dots, \mathfrak{L}(y_p)$ have to be chosen for its expansion.

Lemma 11.3.6. *Let A be a non-star block atom in $\mathcal{R}_i$. Assuming (P3), we can compute the list $\mathfrak{L}(x)$ of the pendant element x corresponding to A in polynomial time.*

Proof. We iterate over quotients in the catalog which are K_2 with a single pendant edge, essentially cycles, or essentially 3-connected graphs by Lemma 7.4.3. Let Q be such a quotient. For $u \in \partial A$, we put $\mathfrak{L}(u) = \partial Q$. For each single pendant element y of A attached at u, we construct $\mathfrak{L}(u)$ consisting of all vertices $v \in \boldsymbol{V}(Q)$ such that the pendent edge/loop/half-edge attached at v belongs to $\mathfrak{L}(y)$. We remove all pendant elements attached at A and all pendant edges/loops/half-edges attached at Q.

By definition of pendant elements, it is possible to expand Q to the block part corresponding to A if and only if there exists a list-compatible isomorphism $A \xrightarrow{\mathfrak{L}} Q$.

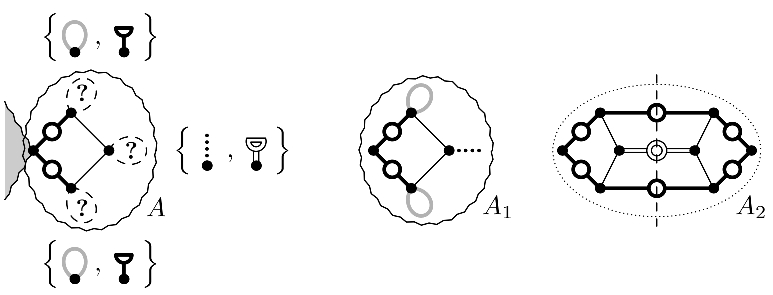


Figure 11.10: On the left, a non-star block atom A in $\mathcal{R}_i$ with depicted lists of its pendant elements. On the right, two possible atoms from the catalog having a quotient for which there exists a list-compatible isomorphism from A. So the list of the pendant element replacing A in $\mathcal{R}_{i+1}$ contains the pendant edge corresponding to the edge-quotient of the block atom A_1 and the half-edge corresponding to a half-quotient of the proper atom A_2.

If Q is K_2 or a cycle, we test it trivially. If Q is 3-connected, we test it using (P3). If $A \xrightarrow{\mathfrak{L}} Q$, we add the pendant edge/loop/half-edge representing this quotient to the list $\mathfrak{L}(x)$, and we remember the constructed isomorphism $A \xrightarrow{\mathfrak{L}} Q$. See Fig. 11.10 for an example. □

On the other hand, if A is a star block atom, we compute its list by a slow subroutine. If this slow subroutine can be avoided and the list for A can be computed in polynomial time, the entire meta-algorithm of Theorem 6.6.3 runs in polynomial time.

Lemma 11.3.7. *Let A be a star block atom in $\mathcal{R}_i$. We can compute the list $\mathfrak{L}(x)$ of the pendant element x corresponding to A in time $\mathcal{O}^*(2^{e(H)/2})$.*

Proof. Each star block atom of $\mathcal{R}_i$ corresponds either to the edge-quotient of a star block atom, or to the loop- or a half-quotient of a dipole. Lemma 10.3.7 states that a dipole can have exponentially many pairwise non-isomorphic half-quotients, we iterate over all of them which gives $2^{e(H)/2}$ part in the complexity bound. Since we postpone reduction of star block atoms, all pendant elements of A necessarily correspond to non-star block atoms in some $\mathcal{R}_j$, for $j < i$, so each pendant element corresponds in H to one subtree of blocks attached at the vertex of A.

Case 1: Dipoles. We iterate over all partially expanded dipoles in the catalog and try to add them to the list $\mathfrak{L}(x)$. Let D be a partially expanded dipole, recall that all edges of D correspond to proper atoms.

We test whether the lists of the pendant elements attached to the star block atom A are compatible with the loop-quotient of D. Each loop of this loop-quotient corresponds to the loop-quotient of some proper atom which is either a cycle with attached single pendant edges, or essentially 3-connected by Lemma 7.4.4. Therefore, it corresponds to exactly one pendant element in A. By Lemma 11.3.5, each loop belongs only to lists of pendant elements of type. Therefore, we just need to compare the number of loops in each color class with the number of lists containing this colored loop. If these numbers match, we add the loop representing the loop-quotient of D to $\mathfrak{L}(x)$.

Then we iterate over all half-quotients of D. By Lemma 10.3.7, let Q be one of its at most $2^{\mathbf{e}(H)/2}$ possible quotients. Recall from Section 11.2 that an edge of D projects either to a half-edge, or together with another edge of D of the same color and type to one loop. So each Q consists of loops and half-edges attached to a vertex. Since all edges of D correspond to proper atoms, each loop and each half-edge has to be matched to one pendant element of A.

Therefore, we test existence of a perfect matching in the following bipartite graph: One part is formed by the loops and the half-edges of Q, and the other part is formed by the pendant elements of A. A loop/half-edge is adjacent to a pendant element, if and only if the corresponding list contains this loop/half-edge. Each perfect matching defines one assignment of the loops and half-edges of Q to the pendant elements of A. See Fig. 11.11 for an example. We add the half-edge corresponding to a half-quotient of D to $\mathfrak{L}(x)$ if and only if there exists a perfect matching for at least one half-quotient Q of D.

Case 2: Star Block Atoms. We iterate over all partially expanded star block atoms of the catalog, let S be one of them. The star block atom S consists of one vertex with attached pendant edges (corresponding to non-star block atoms), loops (corresponding to the loop-quotients of proper atoms) and half-edges. Some of these half-edges correspond to dipoles, and some to proper atoms. Let $h_1, \dots, h_d$ be the half-edges corresponding to partially expanded dipoles $D_1, \dots, D_d$ from the catalog. We construct all expanded edge-quotients Q of S by replacing $h_1, \dots, h_d$ by all possible choices of half-quotients $Q_1, \dots, Q_d$ of $D_1, \dots, D_d$. In total, we have at most $2^{\mathbf{e}(H)/2}$ different expanded edge-quotients Q of S.

All pendant edges of Q correspond to non-star block atoms, and all loops and half-edges correspond to loop- and half-quotients of proper atoms. Therefore, every edge, loop and half-edge attached in Q has to be matched to one pendant element of A. Similarly as above, for each expanded edge-quotient Q, we test whether there exists a perfect matching between edges, loops and half-edges of Q and the lists of pendant elements of A. We add the edge representing the edge-quotient of the star block atom S to $\mathfrak{L}(x)$, if and only if there exists a perfect matching for some expanded edge-quotient Q of S.

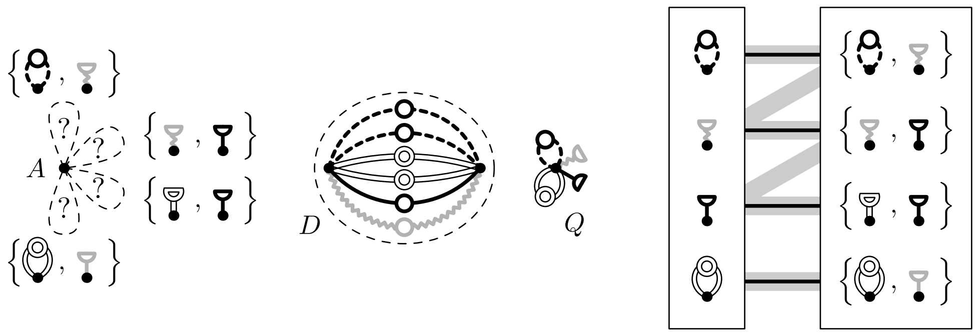


Figure 11.11: The half-edge corresponding to a half-quotient of the dipole D belongs to the list $\mathfrak{L}(x)$ of a pendant element x replacing A because there exists a perfect matching between the loops and half-edges of the half-quotient Q of D and the lists of pendant elements of A.

The procedure computes the list $\mathfrak{L}(x)$ correctly since we test all possible quotients from the catalog, and for each quotient we test all possibilities how it could be matched to A. For each quotient Q, the running time is clearly polynomial, and we have $\mathcal{O}^*(2^{\mathbf{e}(H)/2})$ quotients. □

Algorithm 5 gives the pseudocode for computation of the list $\mathfrak{L}(x)$ of a pendant element x replacing an atom A. If the returned list is empty, we halt the reduction; either H_s is not expandable to H, or we have chosen a wrong core in H.

Testing Expandibility. The reduction with lists ends with a primitive graph $\mathcal{R}_t$ with lists. For one particular choice of a core, $\mathcal{R}_t$ represents the set of graphs $\mathcal{H}_t$ to which H can be reduced.

In the following, we denote by $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$ existence of a *list-compatible isomorphism* which preserves colors and orientations of edges and maps pendant elements x of $\mathcal{R}_t$ into pendant edges, loops and half-edges of H_s such that $\pi(x) \in \mathfrak{L}(x)$. (It corresponds to a list-compatible isomorphism defined in Section **??** when pendant elements/edges/loops/half-edges are removed, as described in the proof of Lemma 11.3.6.)

Lemma 11.3.8. *The graph H_s is expandable to H_0 which is isomorphic to H if and only if $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$ for some choice of the core in H.*

Proof. Suppose that $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$ for some choice of the core. By the definition, every pendant element x of $\mathcal{R}_t$ can be replaced by any member of $\mathfrak{L}(x)$ which can be fully expanded to the block part in H corresponding to x. The list-compatible isomorphism chooses for pendant elements of $\mathcal{R}_t$ realization by edges, loops and half-edges which is compatible with the computed quotient of H_s.

In more detail, we first expand edges in $H_s, \dots, H_{r+1}$ by the unique edge-quotients to reach H_r, this has to be compatible with the sequence of replacements defined by $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$. Then we do replacements in the manner of Theorem 6.6.1, and construct the expansions $H_{r-1}, \dots, H_0$. Since we expand according to the list-compatible isomorphism $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$, the constructed graph H_0 is isomorphic to H.

On the other hand, suppose that H_s is expandable to H_0 which is isomorphic to H. Then according to Lemma 10.3.8, the core of H_s is preserved in H, so it has to correspond to some block or to some articulation of H, which we choose as the core of H. Since there exists a sequence of replacements from H_s which constructs H_0, this sequence of replacements is possible in $\mathcal{R}_t$. Thus $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$. □

Lemma 11.3.9. *Assuming (P3), we can test whether H_s is expandable to H_0 which is isomorphic to H in time $\mathcal{O}^*(2^{\mathbf{e}(H)/2})$.*

Proof. We iterate over all choices of the core in H. For each, we compute the reduction series with lists $H = \mathcal{R}_0, \dots, \mathcal{R}_t$, using Algorithm 5 and Lemmas 11.3.6 and 11.3.7. We modify both graphs $\mathcal{R}_t$ and H_s similarly as in the proof of Lemma 11.3.6. For each pendant element x of $\mathcal{R}_t$ attached at u, we put $\mathfrak{L}(u)$ be the set of all vertices of $\boldsymbol{V}(H_s)$ having an attached pendant edge/loop/half-edge which belongs to $\mathfrak{L}(x)$, and we remove x. We remove all pendant edges/loops/half-edges of H_s.

Algorithm 5: The subroutine for computing lists of pendant elements

```
Require: A block atom $A$ of $\mathcal{R}_i$.
Ensure: The list $\mathfrak{L}(x)$ of the pendant element $x$ replacing $A$ in $\mathcal{R}_{i+1}$.
 1: Initiate the empty list $\mathfrak{L}(x)$.
 2: if $A$ is a non-star block atom then
 3:    Iterate over all quotients from the catalog.
 4:    for each quotient $Q$ from the catalog do
 5:       For each pendant element $x$ of $A$ attached at $u$, set $\mathfrak{L}(u)$ to all vertices $\boldsymbol{V}(Q)$
          having an attached pendant edge/loop/half-edges belonging to $\mathfrak{L}(x)$, and re-
          move $x$.
 6:       Remove all pendant edges/loops/half-edges in $Q$, and test $A \xrightarrow{\mathfrak{L}} Q$ (trivially
          or using (P3)).
 7:       If some list-compatible isomorphism exists, we add the edge/loop/half-edge
          of $Q$ to $\mathfrak{L}(x)$ together with this list-compatible isomorphism.
 8: if $A$ is a star block atom then
 9:    Iterate over all partially expanded dipoles $D$ and star block atoms $S$ in the
       catalog.
10:    for each partially expanded dipole $D$ do
11:       Test whether the loop-quotient of $D$ matches the lists; if yes, then add the
          loop representing $D$ to $\mathfrak{L}(x)$.
12:       Iterate over all half-quotients $Q$ of $D$.
13:       for each half-quotient $Q$ do
14:          Test existence of a perfect matching between the loops and half-edges of $Q$
             and the lists of the pendant elements of $A$.
15:          If a perfect matching exists, add the half-edge of $D$ to $\mathfrak{L}(x)$ together with
             this half-quotient $Q$ and this matching, and proceed with the next dipole.
16:    for each partially expanded star block atom $S$ do
17:       Compute all expanded edge-quotients $Q$ of $S$ by replacing the half-edges
          $h_1, \dots, h_d$ corresponding to the dipoles by all possible combinations of their
          half-quotients $Q_1, \dots, Q_d$.
18:       for each expanded edge-quotient $Q$ do
19:          Test existence of a perfect matching between the edges, loops and half-edges
             of $Q$ and the lists of the pendant elements of $A$.
20:          If a perfect matching exists, then add the edge of $S$ to $\mathfrak{L}(x)$ with this
             expanded edge-quotient $Q$ and this matching, and proceed with the next
             star block atom.
21: return The constructed list $\mathfrak{L}(x)$.
```

Then we test whether $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$. It is trivial to deal with the cases when H_s or $\mathcal{R}_t$ are cycles, K_2 or K_1. Otherwise by Lemma 7.4.2, both H_s and $\mathcal{R}_t$ are 3-connected graphs, and we test $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$ using (P3). By Lemma 11.3.8, this subroutine is correct and runs in time $\mathcal{O}^*(2^{e(H)/2})$. □

11.3.4 Proof of The Main Theorem

Now, we are ready to establish the main algorithmic result of the paper; see Algorithm 6 for the pseudocode. Assuming that a class $\mathcal{C}$ satisfies (P1) to (P3), we show that RegularCover can be solved for $\mathcal{C}$-inputs G in time $\mathcal{O}^*(2^{e(H)/2})$:

Proof of Theorem 6.6.3. We recall the main steps of the algorithm and discuss their time complexity. The reduction series $G_0, \dots, G_r$ can be computed in polynomial time, by Lemmas 7.7.6 and 11.3.4. We reach in G_r one of primitive graphs characterized in Lemma 7.4.2. If G_r is essentially 3-connected, the property (P2) ensures that there are polynomially many semiregular subgroups Γ_r of $\mathrm{Aut}(G_r)$ which can be computed in polynomial time. If G_r is K_2 with attached single pendant edges or essentially a cycle, it is true as well.

For each of these subgroups Γ_r, we compute the quotient $H_r = G_r/\Gamma_r$. Then we compute the reduction series $H_r, \dots, H_s$, again in polynomial time using Lemma 11.3.4. Using Lemma 11.3.9, we test in time $\mathcal{O}^*(2^{e(H)/2})$ whether H_s is expandable to H_0 which is isomorphic to H. We output "yes" if and only if $H_r = G_r/\Gamma_r$ is expandable to H_0 isomorphic to H for at least one the subgroups Γ_r.

To certify the "yes" outputs, we construct the semiregular subgroup $\Gamma \le \mathrm{Aut}(G)$ such that $G/\Gamma \cong H$ as follows. If $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$ for some choice of the core in H, this list-compatible isomorphism describes how to expand H_s to H_0 which is isomorphic to H using Theorem 6.6.1. This expansion replaces edges, loops and half-edges with edge-quotients, loop-quotients and some choices of half-quotients. The expansion towards H_r is deterministic since no half-edges are replaced.

We expand $H_r, \dots, H_0$, together with constructing group extensions $\Gamma_{r-1}, \dots, \Gamma_0$ of Γ_r, where Γ_i is a semiregular subgroup of $\mathrm{Aut}(G_i)$. When G_{i+1} is expanded to G_i, we replace some edges with interiors of some atoms. In the common parts, we define the actions of Γ_i and Γ_{i+1} the same. It remains to define the action of Γ_i on the interiors of these atoms in such a way that $G_i/\Gamma_i = H_i$. When some orbit of atoms is projected to the edge- or loop-quotients, then we isomorphically swap their interiors in Γ_i the same as the corresponding edges are swapped in Γ_{i+1}. For an orbit which is projected to half-quotients, we further compose some automorphisms of Γ_i with the half-quotient defining semiregular involution τ on their interior (when the corresponding edges are flipped in Γ_i). For more details, see [119], proofs of Lemma 4.7 and Theorem 1.3 therein.

It remains to argue correctness of the algorithm. First suppose that the algorithm succeeds. We construct a semiregular subgroup Γ of $\mathrm{Aut}(G)$. By Lemma 11.3.8, some H_s is expandible to H_0 which is isomorphic to H. By Theorem 6.6.1, we get that $G/\Gamma \cong H$ which proves that G regularly covers H. On the other hand, suppose that there exists a semiregular Γ such that $H \cong G/\Gamma$. Then Γ corresponds to the unique semiregular subgroup Γ_r on G_r which is one of the semiregular subgroups tested by the algorithm. Therefore H_r has to be expandable to H_0 isomorphic to H, and we detect this correctly according to Lemma 11.3.8. □

Next, we prove two corollaries. The first corollary states that if G is 3-connected,

Algorithm 6: The meta-algorithm for regular covers – RegularCover

```
Require: A graph $G$ of $\mathcal{C}$ satisfying (P1), (P2) and (P3), and a graph $H$.
Ensure: A semiregular subgroup $\Gamma \le \mathrm{Aut}(G)$ such that $G/\Gamma \cong H$ if it exists.
 1: Compute the reduction series $G_0, \dots, G_r$ ending with the primitive graph $G_r$.
 2: During the reductions, we use Algorithm 4 to add atoms and their quotients into
    the catalog, and to replace them with colored edges.
 3: Using (P2), we compute all semiregular subgroups $\Gamma_r$ of $\mathrm{Aut}(G_r)$.
 4: for each semiregular subgroup $\Gamma_r$ do
 5:   Compute the quotient $H_r = G_r/\Gamma_r$.
 6:   Choose, say, the central block/articulation of $H_r$ as the core.
 7:   Compute the reduction series $H_r, \dots, H_s$ with respect to the core.
 8:   During the reductions, we use Algorithm 4 to add atoms into the catalog, and
      to replace them with colored edges.
 9:   for each guessed position of the core in $H$ do
10:     Compute the reduction series with lists $H = \mathcal{R}_0, \dots, \mathcal{R}_t$.
11:     to compute $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$ do
12:       for each proper atom or dipole $A$ in $\mathcal{R}_i$ do
13:         if $A$ is a proper atom then
14:           Replace its pendant elements with the unique pendant edges from their
              lists. If some list contains no pendant edge, halt and test for other
              choices of the core in $H$.
15:         Replace $A$ with a colored edge using the catalog. Halt and test for other
            choices of the core in $H$ if $A$ is not in the catalog.
16:       for each block atom $A$ in $\mathcal{R}_i$ do
17:         Replace it by a pendant element $x$ whose list $\mathfrak{L}(x)$ is computed using
            Algorithm 5. Halt and test for other choices of the core in $H$ if $\mathfrak{L}(x)$ is
            empty.
18:     Test $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$ using Lemma 11.3.9.
19:     if $\mathcal{R}_t \xrightarrow{\mathfrak{L}} H_s$ then
20:       Using the lists, compute the expansions $H_{s-1}, \dots, H_0$ such that $H_0 \cong H$.
21:       Using Theorem 6.6.1, compute the group extensions $\Gamma_{r-1}, \dots, \Gamma_0 = \Gamma$ to
          the interiors of expanded edges, loops and half-edges. For half-edges, use
          involutions $\tau$ on interiors of replacing half-quotients.
22:       By Lemma 11.3.8, $G/\Gamma \cong H$, so the group $\Gamma$ defines the regular covering
          projection $p : G \to H$.
23:       return The semiregular subgroup $\Gamma \le \mathrm{Aut}(G)$.
24: return The graph $G$ does not regularly cover the graph $H$.
```

H is 2-connected or $k = |G|/|H|$ is odd, the meta-algorithm can avoid the slow subroutine of Lemma 11.3.7 and can be modified to run in polynomial time.

Proof of Corollary 6.6.4. If G is 3-connected, then it is primitive, so $G = G_r$. Therefore, we compute all quotients $H_r = G_r/\Gamma_r$, and test using Lemma 7.7.7 whether

$H_r \cong H$. No reduction with lists needs to be applied. If H is 2-connected, no pendant elements are created the reduction of H with lists, so the slow subroutine can be avoided and even the assumptions (P1), (P2) and (P3*) are sufficient.

If $|\Gamma| = |\Gamma_r|$ is odd, then no half-edges occur in H_r, and so according to Corollary 10.3.6, the expansion gives the unique graph H_0. We just test whether $H_0 \cong H$. □

Next, we prove that we can modify the meta-algorithm to output all regular quotients of G, with a polynomial-time delay.

Proof of Corollary 6.6.5. We compute the reduction series $G = G_0, \dots, G_r$ and all semiregular subgroups Γ_r of $\mathrm{Aut}(G_r)$. Next, we run all possible expansions of $H_r = G_r/\Gamma_r$ to H_0 using Theorem 6.6.1, by all possible choices of half-quotients. All half-quotients of proper atoms can be computed in polynomial time. For dipoles, we can easily generate them with polynomial-time delays. We output all constructed graphs H_0. □

11.4 Star Blocks Atoms with Lists

The bottleneck in the running time of the meta-algorithm of Theorem 6.6.3 is the single slow subroutine in Lemma 11.3.7, computing lists of pendant elements replacing star block atoms in $\mathcal{R}_i$. In this section, we give insights into this problem, which might lead to a faster algorithm for RegularCover of planar graphs.

We show a combinatorial reformulation to finding a certain generalization of a perfect matching which we call IV-Matching. Here we describe a complete derivation of this problem, and in Conclusions we just give its combinatorial statement.

Instance. Figure 11.12 shows an example. Suppose that $\mathcal{R}_i$ contains a star block atom A with attached pendant elements, each with a previously computed list and corresponding to a non-star block atom. We want to determine the list $\mathfrak{L}(x)$ of the pendant element x replacing A in $\mathcal{R}_{i+1}$. The following are the candidates for members of $\mathfrak{L}(x)$:

- Each loop corresponding to the loop-quotient of a dipole D.
- Each half-edge corresponding to a half-quotient of a dipole D.
- Each edge corresponding to the edge-quotient of a star block atom S.

Since loop-quotients of dipoles are uniquely determined, we can easily test them and we can ignore them. The case of a half-quotient of a dipole D can be reduced to a star block atom S with a single half-edge attached corresponding to a half-quotient of D. If S can be matched to A, we instead add the half-edge corresponding to a half-quotient of D to $\mathfrak{L}(x)$. Therefore, in the remainder of this section, we only deal with the case of a star block atoms S. We want to decide whether the edge corresponding to the edge-quotient of S belongs to $\mathfrak{L}(x)$. We shall assume that at least one half-edge attached in S corresponds to a dipole, otherwise the problem is trivial.

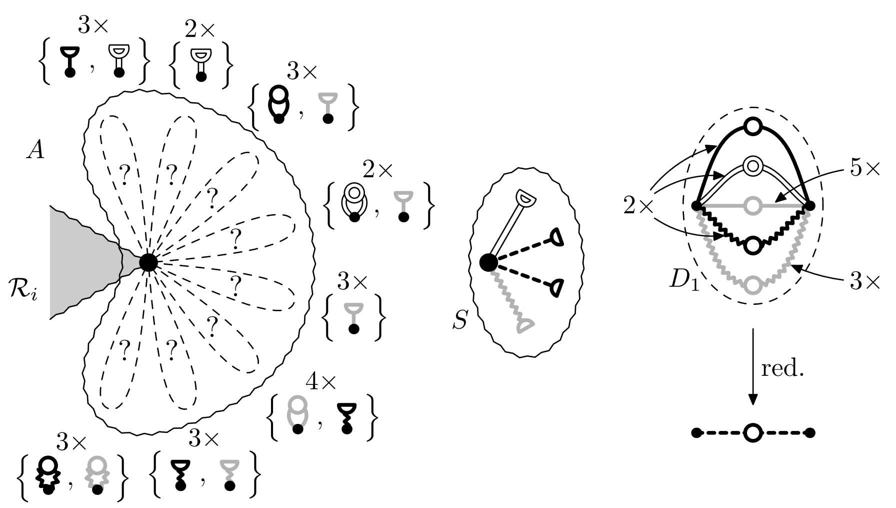


Figure 11.12: On the left, a star block atom A in $\mathcal{R}_i$ with 23 attached pendant elements, together with their lists and multiplicities. On the right, a star block atom S from the catalog which belongs to $\mathfrak{L}(x)$. The bold dashed edges correspond to the partially expanded dipole D_1 whose edges are depicted with multiplicities, the remaining colored edges correspond to proper atoms.

Outline. In Section 11.4.1, we simplify both A and S. In Section 11.4.2, we further apply size constraints to simplify them. In Section 11.4.3, we derive the IV-Matching problem.

11.4.1 Preprocessing Star Block Atoms

The star block atom S has several pendant edges, loops and half-edges attached. Further, we may assume that S is partially expanded (see Section 11.3.2), so its pendant edges correspond to non-star block atoms and its loops correspond to proper atoms. On the other hand, a half-edge can be of two types: either it corresponds to a half-quotient of a proper atom, or of a dipole. For example, in Fig. 11.12 we have two half-edges corresponding to proper atoms, and two half-edges corresponding to dipoles. Further, we may assume that all these dipoles are partially expanded (see Section 11.3.2), so all their edges correspond to proper atoms.

Unifying Dipoles. Recall that every half-quotient of a dipole consists of half-edges

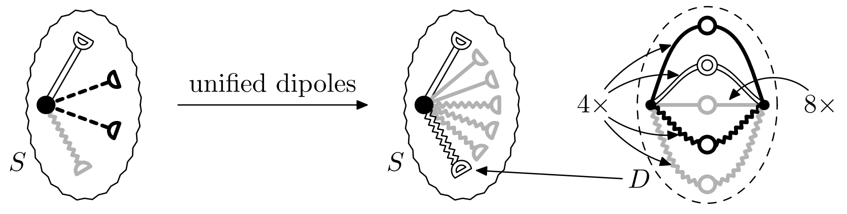


Figure 11.13: We apply the unification on the example in Fig. 11.12 as follows. We unify both occurances of the dipole D_1 into the dipole D. Since the colored classes of gray edges have odd sizes, we attach one half-edge per dipole from each class directly to S.

and loops attached to one vertex. Since S may contain multiple half-edges corresponding to half-quotients of dipoles, we want to unify them into one dipole D containing all their edges. (This may happen in the quotients; see Fig. 11.14.)

The issue is that this unification might introduce additional quotients of D as in Fig. 11.14. If two dipoles both contain an odd number of edges of one color, the unified dipole D has a half-quotient consisting of only loops of this color which is not possible in the case of half-quotients of two separated dipoles. There is an easy fix: we check each dipole and we remove one edge from each color class of odd size (of necessarily halvable edges) and attach the half-edge of this color directly to S. At least one half-edge of this color appears in every half-quotient of this dipole, so the possible half-quotients are not changed. In Fig. 11.13, we illustrate this preprocessing for the example in Fig. 11.12.

Non-halvable Edges of The Dipole. If the dipole D contains some non-halvable edges, then they are paired in every half-quotient of D and form loops. We remove them from D and attach the corresponding number of loops directly in S. After this step, the dipole D contains only even number of halvable edges in each color class.

Attached Pendant Edges and Loops. The star block atom S may have some pendant edges (corresponding to non-star block atoms) and loops (corresponding to proper atoms) attached. Therefore, each attached pendant edge/loop corresponds to exactly one pendant element of A. By Lemma 11.3.5, each is contained in list of only one type of pendant elements, all corresponding to isomorphic block parts in H. Therefore, we can arbitrarily assign pendant elements, remove them from A and remove these pendant edges and loops from S.

Summary. By the preprocessing of S and A described above, we may assume the following. The star block atom S has only half-edges attached, all but one corresponding to proper atoms. The remaining half-edge corresponds to the unified dipole D having only color classes of even sizes of halvable edges corresponding to proper atoms.

For each pendant element x of A, the list $\mathfrak{L}(x)$ contains only half-edges (attached in S or in a half-quotient of D) and loops (corresponding to halvable edges of D).

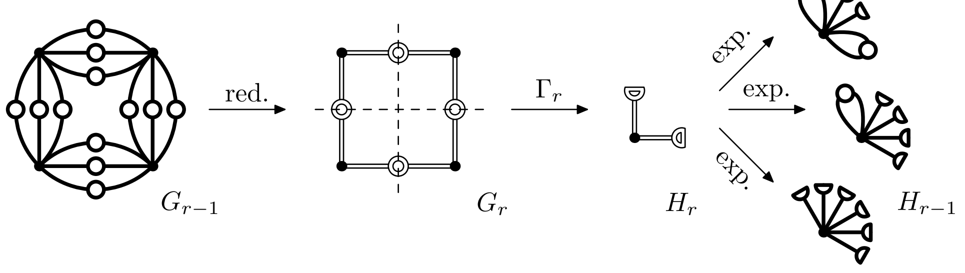


Figure 11.14: For Γ_r generated by two reflections, the quotient H_r consists of a star block atom with two half-edges corresponding to dipoles. All three expansions H_{r-1} up to isomorphism are depicted. But it is not possible to expand H_r to the quotient with three attached loops.

11.4.2 Sizes and Chains

To simplify the problem further, we study sizes of atoms and their quotients. Let A be an atom and let Q be a quotient of this atom. Depending on the type of Q, we get:

- *Q is the edge-quotient:* Then $\boldsymbol{v}(Q) = \boldsymbol{v}(A)$ and $\boldsymbol{e}(Q) = \boldsymbol{e}(A)$.
- *Q is the loop-quotient:* Then $\boldsymbol{v}(Q) = \boldsymbol{v}(A) - 1$ and $\boldsymbol{e}(Q) = \boldsymbol{e}(A)$.
- *Q is a half-quotient:* Then $\boldsymbol{v}(Q) = \boldsymbol{v}(A)/2$ and $\boldsymbol{e}(Q) = \boldsymbol{e}(A)/2$.

Sizes of Expanded Subgraphs and Quotients. Throughout each reduction, we calculate how many vertices and edges are in all the atoms replaced by colored edges which we denote by $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{e}}$. Initially, we put $\hat{\boldsymbol{v}}(e) = 0$ and $\hat{\boldsymbol{e}}(e) = 1$ for every edge $e \in \boldsymbol{E}(G_0)$. For a subgraph X, we define

$$\hat{\boldsymbol{v}}(X) := \boldsymbol{v}(X) + \sum_{e \in \boldsymbol{E}(X)} \hat{\boldsymbol{v}}(e), \qquad \text{and} \qquad \hat{\boldsymbol{e}}(X) := \sum_{e \in \boldsymbol{E}(X)} \hat{\boldsymbol{e}}(e).$$

When an atom A is replaced by an edge e in the reduction, we put $\hat{\boldsymbol{v}}(e) = \hat{\boldsymbol{v}}(\mathring{A})$ and $\hat{\boldsymbol{e}}(e) = \hat{\boldsymbol{e}}(\mathring{A})$. For a subgraph X of G_i, the numbers $\hat{\boldsymbol{v}}(X)$ and $\hat{\boldsymbol{e}}(X)$ are the numbers of vertices and edges when X is fully expanded.

We similarly define $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{e}}$ for quotients and their subgraphs; the difference is that the quotients may contain half-edges. For a half-edge $h \in \boldsymbol{H}(X)$, created by halving an edge e, we put $\hat{\boldsymbol{v}}(h) = \hat{\boldsymbol{v}}(e)/2$ and $\hat{\boldsymbol{e}}(h) = \hat{\boldsymbol{e}}(e)/2$. For a subgraph X, we define

$$\hat{\boldsymbol{v}}(X) := \boldsymbol{v}(X) + \sum_{e \in \boldsymbol{E}(X)} \hat{\boldsymbol{v}}(e) + \sum_{h \in \boldsymbol{H}(X)} \hat{\boldsymbol{v}}(h), \qquad \text{and} \qquad \hat{\boldsymbol{e}}(X) := \sum_{e \in \boldsymbol{E}(X)} \hat{\boldsymbol{e}}(e) + \sum_{h \in \boldsymbol{H}(X)} \hat{\boldsymbol{e}}(h).$$

Sizes of Pendant Elements. We also inductively define $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{e}}$ for pendant elements x and subgraphs X of $\mathcal{R}_0, \dots, \mathcal{R}_t$. Initially, we put $\hat{\boldsymbol{v}}(e) = 0$ and $\hat{\boldsymbol{e}}(e) = 1$ for every edge $e \in \boldsymbol{E}(\mathcal{R}_0)$. For a subgraph X of $\mathcal{R}_i$, let $\boldsymbol{P}(X)$ be the set of all pendant elements in X. We define

$$\hat{\boldsymbol{v}}(X) := \boldsymbol{v}(X) + \sum_{e \in \boldsymbol{E}(X)} \hat{\boldsymbol{v}}(e) + \sum_{y \in \boldsymbol{P}(X)} \hat{\boldsymbol{v}}(\mathring{y}), \qquad \text{and} \qquad \hat{\boldsymbol{e}}(X) := \sum_{e \in \boldsymbol{E}(X)} \hat{\boldsymbol{e}}(e) + \sum_{y \in \boldsymbol{P}(X)} \hat{\boldsymbol{e}}(y),$$

while for a pendant element x corresponding to an atom A in $\mathcal{R}_i$, we define $\hat{\boldsymbol{v}}(x) = \hat{\boldsymbol{v}}(A)$, $\hat{\boldsymbol{v}}(\mathring{x}) = \hat{\boldsymbol{v}}(\mathring{A}) = \hat{\boldsymbol{v}}(x) - 1$, and $\hat{\boldsymbol{e}}(x) = \hat{\boldsymbol{e}}(A)$.

Restricting Lists by Sizes. Next, we show that these sizes can restrict possible members of lists of pendant elements:

Lemma 11.4.1. *For a pendant element x, the possible pendant edge and all loops and half-edges of the list $\mathfrak{L}(x)$ have the same $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{e}}$ as $\hat{\boldsymbol{v}}(x)$ and $\hat{\boldsymbol{e}}(x)$, respectively.*

Proof. The pendant element x corresponds to a block part of H. All members of $\mathfrak{L}(x)$ can be fully expanded to graphs isomorphic to this block part. Necessarily, these graphs contain the same number of vertices and edges as $\hat{\boldsymbol{v}}(x)$ and $\hat{\boldsymbol{e}}(x)$. □

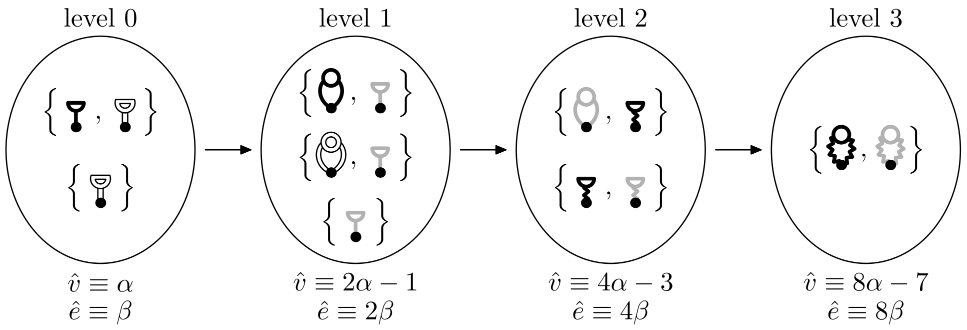


Figure 11.15: A chain of pendant elements with four levels, which is the only chain in Fig. 11.12, for some α and β (we ignore multiplicities of pendant elements). Notice that quotients corresponding to one atom are placed in neighboring levels.

When $\mathfrak{L}(x)$ is computed, we can only consider quotients of the correct sizes, speeding up Algorithm 5. For pendant edges and loops, each belongs to lists of only one type of pendant elements by Lemma 11.3.5. This is not true for half-edges and for purpose of this section, the following is important:

Corollary 11.4.2. *Let x and y be two pendant elements.*

(i) If $\mathfrak{L}(x)$ and $\mathfrak{L}(y)$ share a half-edge, then $\hat{\boldsymbol{v}}(x) = \hat{\boldsymbol{v}}(y)$ and $\hat{\boldsymbol{e}}(x) = \hat{\boldsymbol{e}}(y)$.
(ii) Let $\mathfrak{L}(x)$ contain a loop of a color c and a half-edge of a color c'. Then $\mathfrak{L}(y)$ cannot contain both the loop of the color c' and the half-edge of the color c.

Proof. (i) Implied by Lemma 11.4.1.

(ii) Let $\mathfrak{L}(x)$ contain a loop e and $\mathfrak{L}(y)$ contain a half-edge h of the same color. Then $\hat{\boldsymbol{v}}(x) = \hat{\boldsymbol{v}}(e) + 1$ and $\hat{\boldsymbol{v}}(y) = \hat{\boldsymbol{v}}(e)/2 + 1$ for the vertices, and $\hat{\boldsymbol{e}}(x) = \hat{\boldsymbol{e}}(e)$ and $\hat{\boldsymbol{e}}(y) = \hat{\boldsymbol{e}}(e)/2$ for the edges. Therefore x corresponds to a larger block part in H than y. By the same argument, we deduce that y corresponds to a larger block part in H than x, which gives a contradiction. □

The property (i) relates half-edges together. The property (ii) states that there is a certain size hierarchy on the pendant elements discussed below.

Chains of Pendant Elements. Pendant elements can be partitioned into independent chains, each further partitioned into several levels. The level of size (α, β) consists of all pendant elements x having $\hat{\boldsymbol{v}}(x) = \alpha$ and $\hat{\boldsymbol{e}}(x) = \beta$. Each chain starts with the level 0 of some size (α, β). Further, it contains the levels $m > 0$ of sizes $(2^m\alpha - (2^m - 1), 2^m\beta)$. See Fig. 11.15 for an example.

The key property is the following: if $\mathfrak{L}(x)$ contains a half-edge of a color c and $\mathfrak{L}(y)$ contains the loop of the same color c, then x belongs to a level m and y belongs to the level $m + 1$ of the same chain. A star block atom A can contain multiple chains, but different chains contain completely different colors in their lists, so they are completely independent.

Summary. We may partition S, D, and A according chains of pendant elements of A and test them separately. Therefore, we may assume that there is exactly one chain of pendant elements in A, and only the corresponding edges in D and half-edges in S.

11.4.3 Reduction to the IV-Matching Problem

The star block atom S contains a half-edge corresponding to the dipole D and let $\boldsymbol{H}'(S)$ be the set of the remaining half-edges corresponding to proper atoms. The dipole D has the following half-quotients Q. For each color class of an even size s, we choose an arbitrary integer ℓ such that $0 \le \ell \le \frac{s}{2}$, and attach ℓ loops and $s - 2\ell$ half-edges of this color in Q.

If these values s and ℓ are known for each color class, we can test existence of a perfect matching as described in the proof of Lemma 11.3.7. Since they are not known, we need to solve a generalization of perfect matching called IV-Matching which includes choosing of these values as a part of the problem.

Definition of the Problem. The input of IV-Matching gives the following bipartite graph B. We have $\boldsymbol{V}(B) = \boldsymbol{P}(A) \cup \boldsymbol{E}(D) \cup \boldsymbol{H}'(S)$. For $e \in \boldsymbol{E}(D)$ of a color c and $x \in \boldsymbol{P}(A)$, we have $ex \in \boldsymbol{E}(B)$ if and only if the half-edge or the loop of the color c belongs to the list $\mathfrak{L}(x)$. We call the former case a *half-incidence* and the latter case a *loop-incidence*. Further, we have a *half-incidence* $hx \in \boldsymbol{E}(B)$ between $h \in \boldsymbol{H}'(S)$ of a color c and $x \in \boldsymbol{P}(A)$ if and only if the half-edge of the color c belongs to $\mathfrak{L}(x)$.

We ask whether there exists a *spanning subgraph* B' of B, called an IV-subgraph of B, satisfying the following properties. Each component of connectivity of B' is a path of length one or two (corresponding to I and V in the name). Each $x \in \boldsymbol{P}(A)$ is in B' either half-incident to exactly one vertex in $\boldsymbol{E}(D) \cup \boldsymbol{H}'(S)$, or it is loop-incident to exactly two edges $e, e' \in \boldsymbol{E}(D)$ of the same color class. Further, each vertex of $\boldsymbol{E}(D) \cup \boldsymbol{H}'(S)$ is incident in B' to exactly one $x \in \boldsymbol{P}(A)$. See Fig. 11.16 for an example, with several additional properties which we discuss below.

Level Structure. The structure of levels of the chain of pendant elements transfers into the level structure of B. The part $\boldsymbol{P}(A)$ is partioned into levels called A *levels*. Every half-edge $h \in \boldsymbol{H}'(S)$ is half-incident only to vertices of the A level m of the size $(\hat{\boldsymbol{v}}(h), \hat{\boldsymbol{e}}(h))$. Every edge $e \in \boldsymbol{E}(D)$ is half-incident only to vertices of the A level m of size $(\hat{\boldsymbol{v}}(e)/2, \hat{\boldsymbol{e}}(e)/2)$ and loop-incident only to vertices of the A level $m+1$ of the size $(\hat{\boldsymbol{v}}(e) - 1, \hat{\boldsymbol{e}}(e))$. Therefore, we can define S *levels* for the part $\boldsymbol{E}(D) \cup \boldsymbol{H}'(S)$ such that an S level m contains all vertices half-incident to the A level m or loop-incident to the A level $m+1$. If we depict all levels from left to right according to their order, alternating A levels and S levels, all edges of B go between consecutive levels as depicted in Fig. 11.16.

Clusters. We can view B as a cluster graph. In each S level, $\boldsymbol{E}(D)$ and $\boldsymbol{H}'(S)$ form clusters according to their color classes, called *edge clusters* and *half-edge clusters* respectively. In each A level, the pendant elements form *pendant element clusters* according to equivalence classes of their lists. (We note that two pendant elements with equal lists can correspond to non-isomorphic subgraphs in H. Then their lists contain only half-edges.) Two clusters are either completely adjacent, or not adjacent

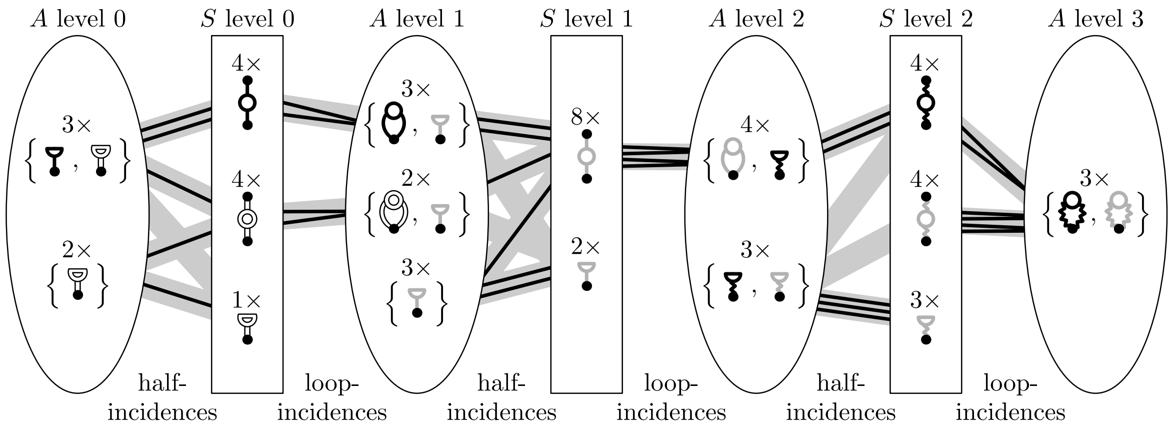


Figure 11.16: The instance of the IV-Matching problem corresponding to the input A in Fig. 11.12 and the preprocessed star block atom S in Fig. 11.13. The edges $\boldsymbol{E}(B)$ are depicted in gray and an IV-subgraph B' is highlighted in bold. We have I's between half-incidences and V's between loop-incidences. The part $\boldsymbol{P}(A)$ is in elipses, the other part $\boldsymbol{E}(D) \cup \boldsymbol{H}'(S)$ is in boxes. The lists $\boldsymbol{P}(A)$, the edges $\boldsymbol{E}(D)$ and the half-edges $\boldsymbol{H}'(S)$ are depicted together with multiplicities.

at all: the subgraph induced by the union of two clusters is either a complete bipartite graph, or contains no edges.

Each pendant element cluster may be loop-adjacent to several edge clusters. On the other hand, each edge cluster is loop-adjacent to at most one pendant element cluster: the loop of the corresponding color belongs to at most one isomorphism class of pendant elements by Lemma 11.3.5. There are no constraints for half-incidences between clusters.

Only Logarithmically Many Levels. The sizes of graphs are growing exponentially with the level number. Therefore, we have at most logarithmically many levels with respect to the size of the input graph H.

Complexity. Since IV-Matching can be used to solve the slow subroutine of Lemma 11.3.7, we get the following in relation to the meta-algorithm of Theorem 6.6.3:

Proposition 11.4.3. *If the IV-Matching problem can be solved for logarithmically many levels in polynomial time, then we can modify the meta-algorithm of Theorem 6.6.3 to run in polynomial time as well.*

Unfortunately, Folwarczný and Knop [128] recently proved that this problem is NP-complete, even for two A levels. Nevertheless, we believe that the particular instances arizing from the RegularCover problem might be solvable in polynomial time and the properties described in this section might be useful for that.

Numbers of Edges Between Levels. We do not know how many half- and loop-incidences are in B' at each cluster, otherwise we could solve the problem directly by finding a perfect matching in a modified graph. On the other hand, these numbers are determined between consecutive A levels and S levels as follows. Let a_m be the number of pendant elements in the A level m and let s_m be the number of edges and half-edges in the S level m. Let B' be an IV-subgraph and let b_i and b'_i be the numbers of edges in B' between the A level i and the S level i, and between the S level i and the A level $i+1$, respectively.

Every pendant element in the A level 0 is half-incident in B' to the S level 0, so $b_0 = a_0$. Therefore, the number of remaining vertices in the S level 0 is $s_0 - b_0$. These vertices are loop-incident in B' to the A level 1, so $b'_0 = s_0 - b_0$ and $a_1 - \frac{b'_0}{2}$ pendant elements remain in the A level 1. We can proceed in this way further, and we get the following inductive formulas:

$$b_i = a_i - \frac{b'_{i-1}}{2}, \qquad \text{and} \qquad b'_i = s_i - b_i.$$

Clearly, each number b'_i has to be even, otherwise no IV-subgraph exists.

11.5 Applying the Meta-algorithm to Planar Graphs

In this section, we discuss automorphism groups of 3-connected planar graphs and we show that the meta-algorithm of Theorem 6.6.3 applies to the class of planar graphs.

Properties (P1) to (P3). We are ready to establish the following:

Lemma 11.5.1. *The class of planar graphs satisfies (P1) to (P3).*

Proof. The class of planar graphs clearly satisfies (P1). For (P2), Lemma 8.5.1 allows to compute $\mathrm{Aut}(G)$ in time $\mathcal{O}(\boldsymbol{v}^2(G))$. Since it is a spherical group, we can generate all linearly many subgroups and check which ones act semiregularly. The property (P3) holds for projectively planar graphs since ListIso can be solved in time $\mathcal{O}(\boldsymbol{v}^{5/2}(G))$ for graphs of bounded genus [209] (even for lists on both vertices and half-edges). □

Theorem 6.6.2. By Lemma 11.5.1, we can apply Theorem 6.6.3. □

11.6 Concluding Remarks

This paper is based on the structural results of [119], describing behaviour of regular graph covering with respect to 1-cuts and 2-cuts in G. In Theorem 6.6.3, we derive an FPT meta-algorithm for testing regular graph covers for $\mathcal{C}$-inputs G where $\mathcal{C}$ is a class of graphs satisfying (P1) to (P3). In particular, this meta-algorithm is tailored for the class of planar graphs (Theorem 6.6.2).

When working with 3-connected decomposition, we described two subroutines we need to solve. First, we have rediscovered graph isomorphism restricted by lists, introduced by Lubiw [260], which lead to several fruitful results in [209]. Second, we introduce a generalization of bipartite matching called the IV-Matching problem, proved to be NP-complete by Folwarcný and Knop [128].

We conclude by several remarks and open problems.

Running Time of The Meta-algorithm. We have omitted polynomial factors in the complexity since the main goal was to establish that RegularCover can be solved in FTP time for $\mathcal{C}$-inputs G for $\mathcal{C}$ satisfying (P1) to (P3). The degree of the polynomial depends on the complexity of polynomial time algorithms in (P2) and (P3).

We roughly estimate the degree of the polynomial for running time of the algorithm of Theorem 6.6.2 for planar graphs. Let $n = \boldsymbol{v}(G) \ge \boldsymbol{v}(H)$. For planar graphs, each primitive graph has $\mathcal{O}(n)$ quotients and each proper atom has at most two half-quotients. Therefore, the catalog contains $\mathcal{O}(n^2)$ atoms and quotients. Each catalog query can be answered in time $\mathcal{O}(n^4)$, and with a suitable canonization even in time $\mathcal{O}(n^3)$.

The simplified reduction series can be computed in time $\mathcal{O}(n)$ by Lemma 7.7.1, the symmetry type of each atom can be determined in time $\mathcal{O}(n^2)$ by Lemma 8.5.1. When adding a proper atom to the catalog, we compute its at most two half-quotients in time $\mathcal{O}(n^2)$ and compute their reduction series in time $\mathcal{O}(n^4)$. Together with catalog queries, we can compute the reduction series and G_r in time $\mathcal{O}(n^5)$.

Next, we iterate over $\mathcal{O}(n)$ quotients H_r of G_r. For each, we compute the reduction series in time $\mathcal{O}(n^4)$. Next, we iterate over $\mathcal{O}(n)$ choices of the core in H. We compute the reduction series with lists where $\mathcal{O}(n)$ subroutines are called. The subroutine of Lemma 11.3.6 runs in time $\mathcal{O}(n^{9/2})$ since ListIso takes time $\mathcal{O}(n^{5/2})$ [209] and we compare each non-star block atom A with $\mathcal{O}(n^2)$ candidates from the catalog. The subroutine of Lemma 11.3.7 runs in time $\mathcal{O}(n^2 2^{\boldsymbol{e}(H)/2})$. The final test in Lemma 11.3.9 runs in time $\mathcal{O}(n^{5/2})$.

In total, the running time of the algorithm in Theorem 6.6.2 is $\mathcal{O}(n^5 \cdot (n^{5/2} + 2^{\boldsymbol{e}(H)/2}))$. By a more careful analysis of the subroutines and possibly reordering them, it should be possible to decrease the degree little bit.

We did not try to optimize the factor $2^{\boldsymbol{e}(H)/2}$. This estimate is certainly very rough and maybe some further techniques from parameterized complexity can be applied to solve IV-Matching faster. We believe that this approach should be followed only when we can prove that RegularCover is NP-complete for planar inputs G. Also, to prove Theorem 6.6.3, it might be possible to design a simpler FPT algorithm running in time $\mathcal{O}^*(2^{\boldsymbol{e}(H)/2})$. Our goal was to obtain as much understanding of the problem as possible, in order to construct a polynomial time algorithm. We failed with the subroutine of Lemma 11.3.7, but nevertheless further structural results may be established and the problem may be solvable in polynomial time.

Possible Extensions of The Meta-algorithm. There are several possible natural extensions of the meta-algorithm. First, we can easily generalize it for input graph G and H with half-edges, directed edges and halvable edges, and also for colored graphs. Further, for a regular covering testing, one can prescribe a list $\mathfrak{L}(u) \subseteq \boldsymbol{V}(H)$ of allowed images of a regular covering projection p for each vertex $u \in \boldsymbol{V}(G)$ such that $p(u) \in \mathfrak{L}(u)$: the expandability testing subroutine can compute with these lists as well.

Complexity of Regular Graph Covering. By Lemmas 11.1.1 and 11.1.3, it follows that RegularCover is GI-hard and belongs to NP. Its complexity remains an open problem:

Problem 11.6.1. *What is the complexity of the* RegularCover *problem?*

One possibility to attack this problem would be to prove that it is NP-hard, or to construct an efficient algorithm using an oracle for GraphIso. If RegularCover is not NP-hard, another possibility is to prove that RegularCover satisfies some properties which unlikely hold for any NP-hard problem. See Section 6.4 for more information about GraphIso.

As a possible next direction of research, we suggest to attack classes of graphs close to planar graphs, for instance projective planar graphs or toroidal graphs. To do so, it seems that new techniques need to be built. Even the automorphism groups of projective planar graphs and toroidal graphs are not yet well understood.

It is natural to ask whether FPT running time of the meta-algorithm of Theorem 6.6.3 is needed:

Problem 11.6.2. *Can the* RegularCover *problem be solved in polynomial time for $\mathcal{C}$-inputs G where $\mathcal{C}$ satisfies (P1) to (P3)? Can it be solved in polynomial time for planar inputs G?*

The IV-Matching Problem. In Section 11.4, we have shown that the bottleneck of the algorithm of Theorem 6.6.3 reduces to a generalized matching problem called IV-Matching. Here, we describe a purely combinatorial formulation of the IV-Matching problem. This reformulation can be useful to understand the problem without regular covering and the structural results described in Chapters 10 and 11.

The input of IV-Matching consists of a bipartite graph B with a partitioning $V_1, \dots, V_\ell$ of its vertices $\boldsymbol{V}(B)$ which we call *levels*, with all edges between of consecutive levels V_i and V_{i+1}, for $i = 1, \dots, \ell - 1$. The levels $V_1, V_3, \dots$ are called *odd* and the levels $V_2, V_4, \dots$ *even*. Further each level V_i is partitioned into several *clusters*, each consisting of a few vertices with identical neighborhoods. There are three key properties:

- The incidences in B respect the clusters; between any two clusters the graph B induces either a complete bipartite graph, or an edge-less graph.
- Each cluster of an even level V_{2t} is incident with at most one cluster at V_{2t+1}.
- The incidences between the clusters of V_{2t-1} and V_{2t} can be arbitrary.

The problem IV-Matching asks whether there is a *spanning subgraph B'* called an IV-subgraph of B. Each component of connectivity of B' equals to a path of length one or two. Each vertex of an odd level V_{2t+1} is in B' adjacent either to exactly one vertex of V_{2t+2}, or to exactly two vertices of V_{2t}. Each vertex of an even level V_{2t} is adjacent to exactly one vertex of the levels $V_{2t-1} \cup V_{2t+1}$. In other words, from V_{2t-1} to V_{2t} the edges of B' form a matching, not necessarily perfect. From V_{2t} to V_{2t+1}, the edges of B' form independent V-shapes, with their centers in the level V_{2t+1}. Figure 11.17 shows an example.

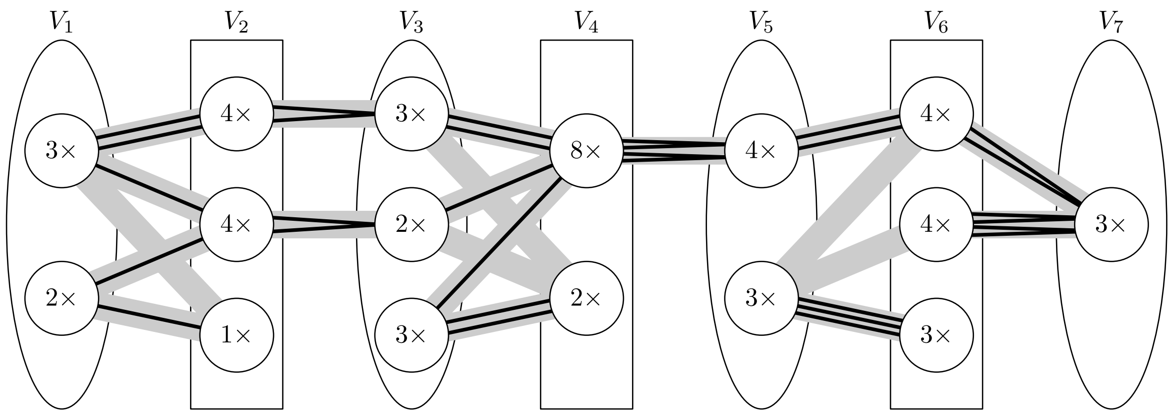


Figure 11.17: An example input B, the clusters are depicted by circles together with their sizes. The odd levels are drawn in circles and the even ones in rectangles. The edges of B are depicted by gray lines between clusters representing complete bipartite graphs. One spanning subgraph B' solving the IV-Matching problem is depicted in bold.

In the conference version of the paper [118], we asked as an open problem what is the complexity of the IV-Matching problem. Recently, Folwarcný and Knop [128] answered this by proving that IV-Matching is strongly NP-hard even for $\ell = 3$. We note that this does not imply NP-hardness of RegularCover, and it is possible that specific instances of IV-Matching arising from RegularCover can be solved in polynomial time.

A Weaker Assumption (P2'). To make Theorem 6.6.3 a more natural generalization of Babai's algorithm [11] for graph isomorphism, it would be nice to replace (P2) with a weaker assumption:

(P2') For a 3-connected graph $G \in \mathcal{C}$ with colored vertices and colored possibly directed edges and a graph H with colored vertices and colored possibly directed edges, half-edges and loops, we can test RegularCover(G, H) in polynomial time.

It is an open problem whether our meta-algorithm can be modified for $\mathcal{C}$ satisfying (P1), (P2') and (P3):[1]

Problem 11.6.3. *Can the* RegularCover *problem be solved in FPT time (with respect to the parameter $\boldsymbol{e}(H)$) for $\mathcal{C}$-inputs G where $\mathcal{C}$ satisfies (P1), (P2'), and (P3)?*

The modified algorithm would have to process both G and H simultaneously. For instance, it is not needed to decide whether a proper atom is halvable or symmetric. Also, we do not need to compute all half-quotients of a proper atom, we only need to test whether some subgraphs located in H are one of them. There are several issues with this approach which make this generalization a very tricky problem.

[1]Even more natural would be to replace (P3) with (P3*), but this problem seems even more tricky.

As illustrated in Fig. 11.6, a half-quotient of a proper atom might not be essentially 3-connected (but it is always essentially 2-connected if the proper atom is not a path). Therefore, we would locate Q' in H which is not a half-quotient of A. To test it using (P2), we would need to expand some proper atoms and dipoles in Q' to reach Q, but it is not clear which ones.

Even more involved is to avoid finding of all quotients $H_r = G_r/\Gamma_r$. Since H_r might consist of many blocks which might contain many proper atoms and dipoles, it is not clear how to locate it in H to test existence of a regular covering using (P2).

Bibliography

[1] J. Abello, M. R. Fellows, and J. C. Stillweil. On the complexity and combinatorics of covering finite complexes. *Australian Journal of Combinatorics*, 4:103–112, 1991.

[2] S. B. Akers and B. Krishnamurthy. On group graphs and their fault tolerance. *IEEE Trans. Comput.*, 36(7):885–888, 1987.

[3] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[4] P. Angelini, G. Di Battista, F. Frati, V. Jelínek, J. Kratochvíl, M. Patrignani, and I. Rutter. Testing planarity of partially embedded graphs. *ACM Transactions on Algorithms*, 11(4):32:1–32:42, 2015.

[5] D. Angluin. Local and global properties in networks of processors. In *ACM Symposium on Theory of Computing*, pages 82–93. ACM, 1980.

[6] D. Angluin and A. Gardiner. Finite common coverings of pairs of regular graphs. *J. Combin. Theory Ser. B*, 30(2):184–187, 1981.

[7] K. Appel and W. Haken. Every planar map is four colorable. *Mathematical Solitaires & Games*, 145, 1980.

[8] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, April 1987.

[9] B. Aspvall, M.F. Plass, and R.E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[10] L. Babai. Automorphism groups of planar graphs i. *Discrete Mathematics*, 2(4):295–307, 1972.

[11] L. Babai. Automorphism groups of planar graphs II. In *Infinite and finite sets (Proc. Conf. Keszthely, Hungary, 1973) Bolyai-North-Holland*, pages 29–84, 1975.

[12] L. Babai. On the isomorphism problem. 1977.

[13] L. Babai. Automorphism groups, isomorphism, reconstruction. In *Handbook of combinatorics (vol. 2)*, pages 1447–1540. MIT Press, 1996.

[14] L. Babai. Graph isomorphism in quasipolynomial time. In *STOC*, 2016.

[15] L. Babai, D.Y. Grigoryev, and D.M. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 310–324. ACM, 1982.

[16] K. A. Baker, P. C. Fishburn, and F. S. Roberts. Partial orders of dimension 2. *Networks*, 2:11–28, 1972.

[17] M. Baker and S. Norine. Harmonic morphisms and hyperelliptic graphs. *Int. Math. Res. Notes*, 15:2914–2955, 2009.

[18] M. Balko, P. Klavík, and Y. Otachi. Bounded representations of interval and proper interval graphs. In *Algorithms and Computation, ISAAC 2013*, volume 8283 of *Lecture Notes in Computer Science*, pages 535–546, 2013.

[19] M. Balko, P. Klavík, and Y. Otachi. Bounded representations of interval and proper interval graphs. *In preparation*, 2017.

[20] J. Bang-Jensen, J. Huang, and X. Zhu. Completing orientations of partially oriented graphs. *CoRR*, abs/1509.01301, 2015.

[21] R. Bar-Yehuda and T. Etzion. Connections between two cycles – a new design of dense processor interconnection networks. *Discrete Applied Mathematics*, 37–38:29–43, 1992.

[22] R. Belmonte, Y. Otachi, and P. Schweitzer. Induced minor free graphs: Isomorphism and clique-width. *CoRR*, abs:1605.08540, 2016.

[23] S. Benzer. Fine structure of a genetic region in bacteriophage. *Proceedings of the National Academy of Sciences*, 41(6):344–354, 1955.

[24] S. Benzer. On the topology of the genetic fine structure. *Proc. Nat. Acad. Sci. U.S.A.*, 45:1607–1620, 1959.

[25] C. Berge. Färbung von graphen, deren sämtliche bzw. deren ungerade kreise starr sind. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 10(114):88, 1961.

[26] C. Berge. *Some classes of perfect graphs.* Internat. Computation Centre, 1966.

[27] D. Bienstock and C. L. Monma. On the complexity of covering vertices by faces in a planar graph. *SIAM Journal on Computing*, 17(1):53–76, 1988.

[28] N. Biggs. *Algebraic graph theory.* Cambridge University Press, 1993.

[29] O. Bílka, J. Jirásek, P. Klavík, M. Tancer, and J. Volec. On the complexity of planar covering of small graphs. In *LNCS, WG*, volume 6986, pages 83–94, 2011.

[30] G. Birkhoff. On groups of automorphisms. *Rev. Un. Mat. Argentina*, 11:155–157, 1946.

[31] M. Biró, M. Hujter, and Zs. Tuza. Precoloring extension. I. Interval graphs. *Discrete Mathematics*, 100(1–3):267–279, 1992.

[32] B. Blank. An imaginary tale book review. *Notices of the AMS*, 46(10):1233–1236, 1999.

[33] T. Bläsius, S. G. Kobourov, and I. Rutter. Simultaneous embedding of planar graphs. In *Handbook of Graph Drawing and Visualization*, pages 349–383. CRC Press, 2013.

[34] T. Bläsius and I. Rutter. Simultaneous PQ-ordering with applications to constrained embedding problems. *ACM Trans. Algorithms*, 12(2):16:1–16:46, 2015.

[35] A. Bobenko, C. Mercat, and M. Schmies. Discrete conformal maps. *Symmetries and integrability of difference equations*, 255:97, 1999.

[36] H. L. Bodlaender. The classification of coverings of processor networks. *Journal of Parallel and Distributed Computing*, 6(1):166–182, 1989.

[37] H. L Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11(4):631–643, 1990.

[38] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209(1):1–45, 1998.

[39] K. S. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms. *J. Comput. System Sci.*, 13:335–379, 1976.

[40] K.S. Booth and C.J. Colbourn. Problems polynomially equivalent to graph isomorphism. *Technical Report CS-77-04, Computer Science Department, University of Waterloo*, 1979.

[41] A. Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, 7(3):243–254, 1987.

[42] A. Bouchet. Unimodularity and circle graphs. *Discrete Mathematics*, 66(1-2):203–208, 1987.

[43] J. M. Boyer and W. J. Myrvold. On the cutting edge: Simplified $o(n)$ planarity by edge addition. *J. Graph Algorithms Appl.*, 8(2):241–273, 2004.

[44] G. R. Brightwell and E. R. Scheinerman. Representations of planar graphs. *SIAM Journal on Discrete Mathematics*, 6(2):214–229, 1993.

[45] P. Buneman. A characterisation of rigid circuit graphs. *Discrete mathematics*, 9(3):205–212, 1974.

[46] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[47] L. Campbell. Dense group networks. *Discrete Applied Mathematics*, 37–38:65–71, 1992.

[48] L. Campbell, G. E. Carlsson, M. J. Dinneen, V. Faber, M. R. Fellows, M. A. Langston, J. W. Moore, A. P. Mullhaupt, and H. B. Sexton. Small diameter symmetric networks from linear groups. *IEEE Trans. Comput.*, 41(2):218–220, 1992.

[49] C. Carathéodory. Untersuchungen über die konformen Abbildungen von festen und veränderlichen Gebieten. *Mathematische Annalen*, 72(1):107–144, 1912.

[50] G. E. Carlsson, J. E. Cruthirds, H. B. Sexton, and C. G. Wright. Interconnection networks based on a generalization of cube-connected cycles. *IEEE Trans. Comput.*, 100(8):769–772, 1985.

[51] N. Carter. *Visual group theory.* MAA, 2009.

[52] A. Cayley. The theory of groups: Graphical representation. *Amer. J. Math.*, 1:174–176, 1878.

[53] M. R. Cerioli, F. de S. Oliveira, and J. L. Szwarcfiter. Linear-interval dimension and pi orders. *Electronic Notes in Discrete Mathematics*, 30:111–116, 2008.

[54] M. R. Cerioli, F. de S. Oliveira, and J. L. Szwarcfiter. On counting interval lengths of interval graphs. *Discrete Applied Mathematics*, 159(7):532 – 543, 2011.

[55] S. Chaplick, P. Dorbec, J. Kratochvíl, M. Montassier, and J. Stacho. Contact representations of planar graphs: Extending a partial representation is hard. In *WG'14*, volume 8747 of *Lecture Notes in Computer Science*, pages 139–151. 2014.

[56] S. Chaplick, S. Felsner, U. Hoffmann, and V. Wiechert. Grid intersection graphs and order dimension. *In preparation.*

[57] S. Chaplick, J. Fiala, P. van't Hof, D. Paulusma, and M. Tesař. Locally constrained homomorphisms on graphs of bounded treewidth and bounded degree. *Theoretical Computer Science*, 590:86–95, 2015.

[58] S. Chaplick, R. Fulek, and P. Klavík. Extending partial representations of circle graphs. In *Graph Drawing, GD 2013*, volume 8242 of *Lecture Notes in Computer Science*, pages 131–142, 2013.

[59] S. Chaplick, R. Fulek, and P. Klavík. Extending partial representations of circle graphs. *CoRR*, abs/1309.2399, 2015.

[60] S. Chaplick, G. Guśpiel, G. Gutowski, T. Krawczyk, and G. Liotta. The partial visibility representation extension problem. In *Graph Drawing and Network Visualization: 24th International Symposium, GD 2016*, volume 9801 of *Lecture Notes in Computer Science*, pages 266–279, 2016.

[61] S. Chaplick, M. Toepfer, J. Voborník, and P. Zeman. On h-topological intersection graphs. In *WG 2017*, Lecture Notes in Computer Science, 2017.

[62] E. Chargaff. Chemical specificity of nucleic acids and mechanism of their enzymatic degradation. *Cellular and Molecular Life Sciences*, 6(6):201–209, 1950.

[63] F. Cheah and D. G. Corneil. On the structure of trapezoid graphs. *Discrete Applied Mathematics*, 66(2):109–133, 1996.

[64] R. Chitnis, L. Egri, and D. Marx. List h-coloring a graph by removing few vertices. In *Algorithms – ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, LNCS, pages 313–324, 2013.

[65] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164:51–229, 2006.

[66] F. R. K. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[67] C. J. Colbourn. On testing isomorphism of permutation graphs. *Networks*, 11(1):13–21, 1981.

[68] C. J. Colbourn and K. S. Booth. Linear times automorphism algorithms for trees, interval graphs, and planar graphs. *SIAM J. Comput.*, 10(1):203–225, 1981.

[69] C. J. Colbourn and M. J. Colbourn. Isomorphism problems involving self-complementary graphs and tournaments. In *Proceedings of the Eighth Manitoba Conference on Numerical Mathematics and Computing (Univ. Manitoba, Winnipeg, Man., 1978), Congr. Numer*, volume 22, pages 153–164, 1978.

[70] J. H. Conway, H. Burgiel, and C. Goodman-Strauss. *The symmetries of things.* CRC Press, 2016.

[71] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 2009.

[72] D. G. Corneil and P. A. Kamula. Extensions of permutation and interval graphs. *Congressus Numerantium*, 58:267–275, 1987.

[73] D. G. Corneil, H. Kim, S. Natarajan, S. Olariu, and A. P. Sprague. Simple linear time recognition of unit interval graphs. *Inf.Process.Lett*, 55(2):99–104, 1995.

[74] D. G. Corneil and D. G. Kirkpatrick. A theoretical analysis of various heuristics for the graph isomorphism problem. *SIAM Journal on Computing*, 9(2):281–297, 1980.

[75] D. G. Corneil, S. Olariu, and L. Stewart. The LBFS structure and recognition of interval graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1905–1953, 2009.

[76] D.G. Corneil, H. Lerchs, and L.S. Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981.

[77] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. *Trees in Algebra and Programming—CAAP'94*, pages 68–84, 1994.

[78] W.H. Cuningham and J. Edmonds. A combinatorial decomposition theory. *Canad. J. Math.*, 32:734–765, 1980.

[79] W.H. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3:214–228, 1982.

[80] 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. *Discrete Mathematics and Theoretical Computer Science*, 15(1):157–182, 2013.

[81] I. Dagan, M. C. Golumbic, and R. Y. Pinter. Trapezoid graphs and their coloring. *Discrete Applied Mathematics*, 21(1):35–46, 1988.

[82] E. Dahlhaus. Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition. *Journal of Algorithms*, 36(2):205–240, 1998.

[83] V. Dalmau, L. Egri, P. Hell, B. Larose, and A. Rafiey. Descriptive complexity of list h-coloring problems in logspace: A refined dichotomy. In *Logic in Computer Science (LICS), 2015 30th Annual ACM/IEEE Symposium on*, pages 487–498, 2015.

[84] P. Damaschke. The hamiltonian circuit problem for circle graphs is NP-complete. *Information Processing Letters*, 32(1):1–2, 1989.

[85] B. Das, M. K. Enduri, and I. V. Reddy. Polynomial-time algorithm for isomorphism of graphs with clique-width at most three. *CoRR*, 1506.01695, 2015.

[86] S. Datta, N. Limaye, and P. Nimbhorkar. 3-connected planar graph isomorphism is in log-space. In *FSTTCS*, 2008.

[87] S. Datta, N. Limaye, P. Nimbhorkar, T. Thierauf, and F. Wagner. Planar graph isomorphism is in logspace. In *Proc. 24th IEEE CCC*, pages 203–214, 2009.

[88] H. de Fraysseix. Local complementation and interlacement graphs. *Discrete Mathematics*, 33(1):29–35, 1981.

[89] H. de Fraysseix and P. O. de Mendez. On a characterization of gauss codes. *Discrete & Computational Geometry*, 22(2):287–295, 1999.

[90] H. de Fraysseix, P. O. de Mendez, and J. Pach. Representation of planar graphs by segments. In *Intuitive Geometry*, volume 63 of *Coll. Math. Soc. J. Bolyai*, pages 109–117, 1994.

[91] H. De Fraysseix, P. O. de Mendez, and P. Rosenstiehl. On triangle contact graphs. *Combinatorics, Probability and Computing*, 3(02):233–246, 1994.

[92] H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. Trémaux trees and planarity. *International Journal of Foundations of Computer Science*, 17(05):1017–1029, 2006.

[93] J. W. Demmel. *Applied numerical linear algebra.* SIAM, 1997.

[94] 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.

[95] G. Di Battista and R. Tamassia. Incremental planarity testing. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 436–441. IEEE, 1989.

[96] G. Di Battista and R. Tamassia. On-line graph algorithms with SPQR-trees. In *International Colloquium on Automata, Languages, and Programming*, pages 598–611. Springer, 1990.

[97] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM Journal on Computing*, 25(5):956–997, 1996.

[98] R. Diestel. *Graph theory.* Springer, 2000.

[99] G. A. Dirac. On rigid circuit graphs. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 25, pages 71–76. Springer, 1961.

[100] T. Drakengren and P. Jonsson. Eight maximal tractable subclasses of Allen's algebra with metric time. *Journal of Artificial Intelligence Research*, 7:25–45, 1996.

[101] T. A. Driscoll and L. N. Trefethen. *Schwarz-Christoffel mapping*, volume 8. Cambridge University Press, 2002.

[102] B. Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63(3):600–610, 1941.

[103] P. Eades and X. Lin. Spring algorithms and symmetry. *Theor. Comput. Sci.*, 240(2):379–405, 2000.

[104] G. Ehrlich, S. Even, and R. E. Tarjan. Intersection graphs of curves in the plane. *Journal of Combinatorial Theory, Series B*, 21(1):8–20, 1976.

[105] M. Elberfeld and P. Schweitzer. Canonizing Graphs of Bounded Tree Width in Logspace. In *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, volume 47 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:14, 2016.

[106] P. Erdös, S. Fajtlowicz, and A. J. Hoffman. Maximum degree in graphs of diameter 2. *Networks*, 10(1):87–90, 1980.

[107] S. Evdokimov and I. Ponomarenko. Circulant graphs: recognizing and isomorphism testing in polynomial time. *St. Petersburg Mathematical Journal*, 15(6):813–835, 2004.

[108] S. Evdokimov and I.N. Ponomarenko. Isomorphism of coloured graphs with slowly increasing multiplicity of jordan blocks. *Combinatorica*, 19(3):321–333, 1999.

[109] S. Even and A. Itai. Queues, stacks, and graphs. *Theory of Machines and Computation (Z. Kohavi and A. Paz, Eds.)*, pages 71–76, 1971.

[110] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691, 1976.

[111] S. Even, A. Pnueli, and A. Lempel. Permutation graphs and transitive graphs. *Journal of the ACM (JACM)*, 19(3):400–410, 1972.

[112] S. Even and R. E. Tarjan. Computing an st-numbering. *Theoretical Computer Science*, 2(3):339–344, 1976.

[113] H. M. Farkas and I. Kra. Riemann surfaces. In *Riemann Surfaces*, volume 71 of *Graduate Texts in Mathematics*, pages 9–31. 1992.

[114] I. Fáry. On straight-line representation of planar graphs. *Acta Sci. Math. (Szeged)*, 11:229–233, 1948.

[115] T. Feder and P. Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory, Series B*, 72(2):236–250, 1998.

[116] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[117] J. Fiala. Note on the computational complexity of covering regular graphs. In *9th Annual Conference of Doctoral Students, WDS'00*, pages 89–90. Matfyzpress, 2000.

[118] J. Fiala, P. Klavík, J. Kratochvíl, and R. Nedela. Algorithmic aspects of regular graph covers with applications to planar graphs. In *ICALP 2014*, volume 8572 of *LNCS*, pages 489–501, 2014.

[119] J. Fiala, P. Klavík, J. Kratochvíl, and R. Nedela. 3-connected reduction for regular graph covers. *CoRR*, abs/1503.06556, 2017.

[120] J. Fiala, P. Klavík, J. Kratochvíl, and R. Nedela. Algorithmic aspects of regular graph covers. *CoRR*, abs/1609.03013, 2017.

[121] J. Fiala and J. Kratochvíl. Locally constrained graph homomorphisms—structure, complexity, and applications. *Computer Science Review*, 2(2):97–111, 2008.

[122] L.S. Filotti and J.N. Mayer. A polynomial time algorithm for determining the isomorphism of graphs of fixed genus. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, STOC '80, pages 236–243, 1980.

[123] P. C. Fishburn. Intransitive indifference with unequal indifference intervals. *Journal of Mathematical Psychology*, 7(1):144–149, 1970.

[124] P. C. Fishburn. Paradoxes of two-length interval orders. *Discrete Math.*, 52(2):165–175, 1984.

[125] P. C. Fishburn. A characterization of uniquely representable interval graphs. *Discrete Applied Mathematics*, 12:191–194, 1985.

[126] P. C. Fishburn. Interval graphs and interval orders. *Discrete Math.*, 55(2):135–149, 1985.

[127] P. C. Fishburn. *Interval orders and interval graphs: A study of partially ordered sets.* John Wiley & Sons, 1985.

[128] L. Folwarczný and D. Knop. IV-matching is strongly NP-hard. *CoRR*, abs/1506.08388, 2015.

[129] F.V. Fomin, J. Kratochvíl, D. Lokshtanov, F. Mancini, and J.A. Telle. On the complexity of reconstructing h-free graphs from their star systems. *Journal of Graph Theory*, 68(2):113–124, 2011.

[130] M. Fontet. Linear algorithms for testing isomorphism of planar graphs. In *Proceedings Third Colloquium on Automata, Languages, and Programming*, pages 411–423, 1976.

[131] R. Frucht. Herstellung von graphen mit vorgegebener abstrakter gruppe. *Compositio Mathematica*, 6:239–250, 1939.

[132] R. Frucht. Graphs of degree three with a given abstract group. *Canadian J. Math*, 1:365–378, 1949.

[133] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pac. J. Math.*, 15:835–855, 1965.

[134] M. Furst, J. Hopcroft, and E. Luks. Polynomial-time algorithms for permutation groups. In *FOCS*, pages 36–41. IEEE, 1980.

[135] C. P. Gabor, K. J. Supowit, and W. Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36(3):435–473, 1989.

[136] J. Gajarský, D. Lokshtanov, J. Obdržálek, S. Ordyniak, M.S. Ramanujan, and S. Saurabh. FO model checking on posets of bounded width. In *FOCS 2015*, pages 963–974.

[137] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Hungarica*, 18(1):25–66, 1967.

[138] R. Ganian, P. Hlinený, D. Král, J. Obdržálek, J. Schwartz, and J. Teska. FO model checking of interval graphs. *Logical Methods in Computer Science*, 11(4:11):1–20, 2015.

[139] M. Garey, D. Johnson, G. L. Miller, and C. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980.

[140] M. R. Garey and D. S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

[141] M. R. Garey and D. S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. *San Francisco, LA: Freeman*, 58, 1979.

[142] T. Gavenčiak, P. Gordinowicz, V. Jelínek, P. Klavík, and J. Kratochvíl. Cops and robbers on string graphs. In *Algorithms and Computation, ISAAC 2015*, volume 9472 of *Lecture Notes in Computer Science*, pages 355–366, 2015.

[143] T. Gavenčiak, P. Gordinowicz, V. Jelínek, P. Klavík, and J. Kratochvíl. Cops and robbers on intersection graphs. *CoRR*, abs/1607.08058, 2016.

[144] T. Gavenčiak, V. Jelínek, P. Klavík, and J. Kratochvíl. Cops and robbers of intersection graphs. In *Algorithms and Computation, ISAAC 2013*, volume 8283 of *Lecture Notes in Computer Science*, pages 174–184, 2013.

[145] 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.

[146] F. Gavril. A recognition algorithm for the intersection of graphs of paths in trees. *Discrete Mathematics*, 23:211–227, 1978.

[147] F. Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73(5-6):181–188, 2000.

[148] A. Ghouila-Houri. Caractérisation des graphes non orientés dont on peut orienter les arêtes de manière à obtenir le graphe d'une relation d'ordre. *C. R. Acad. Sci. Paris*, 254:1370–1371, 1962.

[149] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Can. J. Math.*, 16:539–548, 1964.

[150] E. Gioan, C. Paul, M. Tedder, and D. Corneil. Practical and efficient circle graph recognition. *Algorithmica*, 69(4):759–788, 2014.

[151] C. Godsil and G. F. Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.

[152] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *FOCS*, volume 86, pages 174–187, 1986.

[153] D. M. Goldschmidt. Automorphisms of trivalent graphs. *Annals of Mathematics*, 111(2):377–406, 1980.

[154] M. C. Golumbic. The complexity of comparability graph recognition and coloring. *J. Combin. Theory, Ser. B*, 22:68–90, 1977.

[155] M. C. Golumbic. Reasoning about time. In *Mathematical Aspects of Artificial Intelligence, F. Hoffman, ed.*, volume 55, pages 19–53, 1998.

[156] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. North-Holland Publishing Co., 2004.

[157] M. C. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. *Journal of Algorithms*, 19(3):449–473, 1995.

[158] M. C. Golumbic, D. Rotem, and J. Urrutia. Comparability graphs and intersection graphs. *Discrete Mathematics*, 43(1):37–46, 1983.

[159] M. C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the ACM (JACM)*, 40(5):1108–1133, 1993.

[160] R. E. Greene and K.-T. Kim. The riemann mapping theorem from riemann's viewpoint. *Complex Analysis and its Synergies*, 3(1):1, 2017.

[161] R. J. Greenspan. Seymour Benzer (1921–2007). *Current Biology*, 18(3):R106–R110, 2008.

[162] M. Grohe. Descriptive complexity, canonisation, and definable graph structure theory. *Manuscript*, 2012.

[163] M. Grohe and D. Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 173–192, 2012.

[164] M. Grohe and P. Schweitzer. Isomorphism testing for graphs of bounded rank width. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1010–1029, 2015.

[165] J. L. Gross and T. W. Tucker. *Topological graph theory*. Courier Dover Publications, 2001.

[166] D. Guo, J. Wu, H. Chen, and X. Luo. Moore: An extendable peer-to-peer network based on incomplete Kautz digraph with constant degree. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 821–829, 2007.

[167] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In *International Symposium on Graph Drawing*, pages 77–90. Springer, 2000.

[168] M. Habib, R. McConnell, C. Paul, and L. Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234:59–84, 2000.

[169] M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.

[170] H. Hadwiger, H. Debrunner, and V. Klee. *Combinatorial geometry in the plane.* Courier Corporation, 2015.

[171] A. Hajnal and J. Surányi. Über die auflösung von graphen in vollständige teilgraphen. *Ann. Univ. Sci. Budapest, Eötvös Sect. Math*, 1:113–121, 1958.

[172] G. Hajós. Über eine Art von Graphen. *Internationale Mathematische Nachrichten*, 11:65, 1957.

[173] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management science*, 17(3):219–229, 1970.

[174] P. Hanlon. Counting interval graphs. *Transactions of the American Mathematical Society*, 272(2):383–426, 1982.

[175] F. Harary. *Graph theory.* Addison-Wesley, Reading, MA, 1969.

[176] W. A. Harris. Seymour Benzer 1921–2007 the man who took us from genes to behaviour. *PLoS Biol*, 6(2):e41, 2008.

[177] I. B. Hartman, I. Newman, and R. Ziv. On grid intersection graphs. *Discrete Mathematics*, 87(1):41–52, 1991.

[178] Z. Hedrlín and A. Pultr. On full embeddings of categories of algebras. *Illinois Journal of Mathematics*, 10(3):392–406, 1966.

[179] P. Hell, D. Kirkpatrick, P. Klavík, and Y. Otachi. Minimal forbidden induced subgraphs for k-nested interval graphs. *In preparation*, 2017.

[180] P. Hell and J. Nešetřil. On the complexity of H-coloring. *J. Combin. Theory Ser. B*, 48(1):92–110, 1990.

[181] P. Hell and J. Nešetril. *Graphs and homomorphisms*, volume 28 of *Oxford Lecture Series in Mathematics and its Applications.* Oxford University Press, 2004.

[182] A. J. Hoffman and R. R. Singleton. On Moore graphs with diameters 2 and 3. *IBM Journal of Research and Development*, 4(5):497–504, 1960.

[183] I. Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981.

[184] S. Hong and P. Eades. Drawing planar graphs symmetrically, II: Biconnected planar graphs. *Algorithmica*, 42(2):159–197, 2005.

[185] S. Hong and P. Eades. Drawing planar graphs symmetrically, III: Oneconnected planar graphs. *Algorithmica*, 44(1):67–100, 2006.

[186] S. Hong, B. McKay, and P. Eades. Symmetric drawings of triconnected planar graphs. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 356–365, 2002.

[187] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[188] J. E. Hopcroft and R. E. Tarjan. A V^2 algorithm for determining isomorphism of planar graphs. *Information Processing Letters*, 1(1):32–34, 1971.

[189] J. E. Hopcroft and R. E. Tarjan. Isomorphism of planar graphs. In *Complexity of computer computations*, pages 131–152. Springer, 1972.

[190] J. E. Hopcroft and R. E. Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

[191] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973.

[192] J. E. Hopcroft and R. E. Tarjan. A $V \log V$ algorithm for isomorphism of triconnected planar graphs. *Journal of Computer and System Sciences*, 7(3):323–331, 1973.

[193] J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[194] J. E. Hopcroft and J. Wong. Linear time algorithm for isomorphism of planar graphs. In *STOC*, pages 172–184. ACM, 1974.

[195] W. L. Hsu. $\mathcal{O}(M \cdot N)$ algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM Journal on Computing*, 24(3):411–439, 1995.

[196] N. Immerman and E. Lander. Describing graphs: A first-order approach to graph canonization. In *Complexity theory retrospective*, pages 59–81. Springer, 1990.

[197] K. Jampani and A. Lubiw. The simultaneous representation problem for chordal, comparability and permutation graphs. In *Algorithms and Data Structures*, volume 5664 of *Lecture Notes in Computer Science*, pages 387–398. 2009.

[198] K. Jampani and A. Lubiw. Simultaneous interval graphs. In *Algorithms and Computation*, volume 6506 of *Lecture Notes in Computer Science*, pages 206–217. 2010.

[199] Y.-N. Jan and L. Jan. Seymour Benzer (1921-2007). *Science*, 319(5859):45–45, 2008.

[200] V. Jelínek, J. Kratochvíl, and I. Rutter. A kuratowski-type theorem for planarity of partially embedded graphs. *Comput. Geom.*, 46(4):466–492, 2013.

[201] F. Joos, C. Löwenstein, F. de S. Oliveira, D. Rautenbach, and J. L. Szwarcfiter. Graphs of interval count two with a given partition. *Inform. Process. Lett.*, 114(10):542–546, 2014.

[202] C. Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, 70:185–190, 1869.

[203] H. Kaplan and Y. Nussbaum. A simpler linear-time recognition of circular-arc graphs. *Algorithmica*, 61(3):694–737, 2011.

[204] R. M. Karp. Mapping the genome: Some combinatorial problems arising in molecular biology. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 278–285, 1993.

[205] M. O. Katanaev. All universal coverings of two-dimensional gravity with torsion. *Journal of mathematical physics*, 34:700–736, 1993.

[206] K. Kawarabayashi. Graph isomorphism for bounded genus graphs in linear time. *CoRR*, abs/1511.02460, 2015.

[207] K. Kawarabayashi and B. Mohar. Graph and map isomorphism and all polyhedral embeddings in linear time. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 471–480, 2008.

[208] D. G. Kendall. Incidence matrices, interval graphs and seriation in archaeology. *Pac. J. Math*, 28(3):565–570, 1969.

[209] P. Klavík, D. Knop, and P. Zeman. Graph isomorphism restricted by lists. *CoRR*, abs/1607.03918, 2016.

[210] P. Klavík, J. Kratochvíl, T. Krawczyk, and B. Walczak. Extending partial representations of function graphs and permutation graphs. In *Algorithms, ESA 2012*, volume 7501 of *Lecture Notes in Computer Science*, pages 671–682, 2012.

[211] P. Klavík, J. Kratochvíl, Y. Otachi, I. Rutter, T. Saitoh, M. Saumell, and T. Vyskočil. Extending partial representations of proper and unit interval graphs. In *Algorithm Theory, SWAT 2014*, volume 8503 of *Lecture Notes in Computer Science*, pages 253–264, 2014.

[212] P. Klavík, J. Kratochvíl, Y. Otachi, I. Rutter, T. Saitoh, M. Saumell, and T. Vyskočil. Extending partial representations of proper and unit interval graphs. *Algorithmica*, 77(4):1071–1104, 2017.

[213] P. Klavík, J. Kratochvíl, Y. Otachi, and T. Saitoh. Extending partial representations of subclasses of chordal graphs. In *Algorithms and Computation, ISAAC 2012*, volume 7676 of *Lecture Notes in Computer Science*, pages 444–454, 2012.

[214] P. Klavík, J. Kratochvíl, Y. Otachi, and T. Saitoh. Extending partial representations of subclasses of chordal graphs. *Theoretical Computer Science*, 576:85–101, 2015.

[215] P. Klavík, J. Kratochvíl, Y. Otachi, T. Saitoh, and T. Vyskočil. Extending partial representations of interval graphs. *Algorithmica*, 2016.

[216] P. Klavík, J. Kratochvíl, and T. Vyskočil. Extending partial representations of interval graphs. In *Theory and Applications of Models of Computation, TAMC 2011*, volume 6648 of *Lecture Notes in Computer Science*, pages 276–285, 2011.

[217] P. Klavík, R. Nedela, and P. Zeman. Jordan-like characterization of automorphism groups of planar graphs. *CoRR*, abs/1506.06488, 2017.

[218] P. Klavík, R. Nedela, and P. Zeman. On orbits of isometries of polyhedra. 2017.

[219] P. Klavík, Y. Otachi, and J. Šejnoha. On the classes of interval graphs of limited nesting and count of lengths. In *27th International Symposium on Algorithms and Computation, ISAAC 2016*, volume 64 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 45:1–45:13, 2016.

[220] P. Klavík, Y. Otachi, and J. Šejnoha. Extending partial representations of interval graphs of limited nesting. *In preparation*, 2017.

[221] P. Klavík, Y. Otachi, and J. Šejnoha. On the classes of interval graphs of limited nesting and count of lengths. *CoRR*, abs/1510.03998, 2017.

[222] P. Klavík and M. Saumell. Minimal obstructions for partial representations of interval graphs. In *Algorithms and Computation, ISAAC 2014*, volume 8889 of *Lecture Notes in Computer Science*, pages 401–413, 2014.

[223] P. Klavík and M. Saumell. Minimal obstructions for partial representations of interval graphs. *CoRR*, 1406.6228, 2015.

[224] P. Klavík and P. Zeman. Automorphism groups of geometrically represented graphs. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 540–553, 2015.

[225] P. Klavík and P. Zeman. Automorphism groups of geometrically represented graphs. *CoRR*, abs/1407.2136, 2015.

[226] V. Klee. What are the intersection graphs of arcs in a circle? *The American Mathematical Monthly*, 76(7):810–813, 1969.

[227] T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[228] J. Köbler and S. Kuhnert. The isomorphism problem of k-trees is complete for logspace. In *Proc. 34th MFCS*, pages 537–448, 2009.

[229] J. Köbler, S. Kuhnert, B. Laubner, and O. Verbitsky. Interval graphs: Canonical representations in logspace. *SIAM Journal on Computing*, 40(5):1292–1315, 2011.

[230] J. Kobler, S. Kuhnert, and O. Watanabe. Interval graph representation with given interval and intersection lengths. In *Algorithms and Computation*, volume 7676 of *Lecture Notes in Computer Science*, pages 517–526. 2012.

[231] S. G. Kobourov. Spring embedders and force directed graph drawing algorithms. *CoRR*, abs/1201.3011, 2012.

[232] P. Koebe. *Kontaktprobleme der konformen Abbildung.* Hirzel, 1936.

[233] Y. Koren. On spectral graph drawing. In *International Computing and Combinatorics Conference*, pages 496–508. Springer, 2003.

[234] Y. Koren. Drawing graphs by eigenvectors: theory and practice. *Computers & Mathematics with Applications*, 49(11):1867–1888, 2005.

[235] N. Korte and R. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18(1):68–81, 1989.

[236] A. Kostochka and J. Kratochvíl. Covering and coloring polygon-circle graphs. *Discrete Mathematics*, 163(1-3):299–305, 1997.

[237] J. Kratochvíl. String graphs. II. recognizing string graphs is NP-hard. *Journal of Combinatorial Theory, Series B*, 52(1):67–78, 1991.

[238] J. Kratochvíl and J. Matoušek. String graphs requiring exponential representations. *J. Comb. Theory, Ser. B*, 53(1):1–4, 1991.

[239] J. Kratochvíl and M. Pergel. Intersection graphs of homothetic polygons. *Electronic Notes in Discrete Mathematics*, 31:277–280, 2008.

[240] J. Kratochvíl, A. Proskurowski, and J. A. Telle. Covering regular graphs. *J. Comb. Theory Ser. B*, 71(1):1–16, 1997.

[241] J. Kratochvíl and Zs. Tuza. Algorithmic complexity of list colorings. *Discrete Applied Mathematics*, 50(3):297–302, 1994.

[242] S. Kratsch and P. Schweitzer. Graph isomorphism for graph classes characterized by two forbidden induced subgraphs. In *WG 2012*, volume 7551 of *Lecture Notes in Computer Science*, pages 34–45, 2012.

[243] T. Krawczyk and B. Walczak. Extending partial representations of trapezoid graphs. In *WG 2017*, Lecture Notes in Computer Science, 2017.

[244] A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *J. ACM*, 50(5):591–640, 2003.

[245] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15:217–283, 1930.

[246] F. Lalonde. Le problème d'étoiles pour graphes est NP-complet. *Discrete Mathematics*, 33(3):271–280, 1981.

[247] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*, pages 296–303. ACM, 2014.

[248] R. Leibowitz, S. F. Assmann, and G. W. Peck. The interval count of a graph. *SIAM J. Algebr. Discrete Methods*, 3:485–494, 1982.

[249] C. Lekkerkerker and D. Boland. Representation of finite graphs by a set of intervals on the real line. *Fund. Math.*, 51:45–64, 1962.

[250] P. A. Levene. The structure of yeast nucleic acid. *Studies from the Rockefeller Institute for Medical Research*, 30:221, 1919.

[251] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[252] D. Lichtenstein. Isomorphism for graphs embeddable on the projective plane. In *ACM Symposium on Theory of Computing*, STOC '80, pages 218–224, 1980.

[253] M. C. Lin and J. L. Szwarcfiter. Characterizations and recognition of circular-arc graphs and subclasses: A survey. *Discrete Mathematics*, 309(18):5618–5635, 2009.

[254] S. Lindell. A logspace algorithm for tree canonization. In *Proc. 24th ACM STOC*, pages 400–404, 1992.

[255] N. Lindzey and R. M. McConnell. Linear-time algorithms for finding tucker submatrices and lekkerkerker–boland subgraphs. *SIAM Journal on Discrete Mathematics*, 30(1):43–69, 2016.

[256] D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. In *FOCS*, pages 186–195, 2014.

[257] P. J. Looges and S. Olariu. Optimal greedy algorithms for indifference graphs. *Comput. Math. Appl.*, 25:15–25, 1993.

[258] L. Lovász. A characterization of perfect graphs. *Journal of Combinatorial Theory, Series B*, 13(2):95–98, 1972.

[259] L. Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 306(10):867–875, 2006.

[260] A. Lubiw. Some NP-complete problems similar to graph isomorphism. *SIAM Journal on Computing*, 10(1):11–21, 1981.

[261] R. D. Luce. Semiorders and a theory of utility discrimination. *Econometrica, Journal of the Econometric Society*, pages 178–191, 1956.

[262] G. S. Lueker and K. S. Booth. A linear time algorithm for deciding interval graph isomorphism. *Journal of the ACM (JACM)*, 26(2):183–195, 1979.

[263] E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.

[264] T.-H. Ma and J. P. Spinrad. On the 2-chain subgraph cover and related problems. *Journal of Algorithms*, 17(2):251–268, 1994.

[265] S. Mac Lane. A structural characterization of planar combinatorial graphs. *Duke Mathematical Journal*, 3(3):460–472, 1937.

[266] B. Maddox. The double helix and the 'wronged heroine'. *Nature*, 421(6921):407–408, 2003.

[267] A. Malnič, R. Nedela, and M. Škoviera. Lifting graph automorphisms by voltage assignments. *European Journal of Combinatorics*, 21(7):927–947, 2000.

[268] P. Mani. Automorphismen von polyedrischen graphen. *Mathematische Annalen*, 192(4):279–303, 1971.

[269] E. S. Marczewski. Sur deux propriétés des classes d'ensembles. *Fund. Math.*, 33:303–307, 1945.

[270] R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8(3):131–136, 1979.

[271] R. M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, 2003.

[272] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Math.*, 201(1-3):189–241, 1999.

[273] T. Mchedlidze, M. Nöllenburg, and I. Rutter. Drawing planar graphs with a prescribed inner face. In *Graph Drawing*, pages 316–327, 2013.

[274] B. D. McKay, M. Miller, and J. Širáň. A note on large graphs of diameter two and given maximum degree. *J. Combin. Theory Ser. B*, 74(1):110–118, 1998.

[275] T. A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory.* SIAM Monographs on Discrete Mathematics and Applications, 1999.

[276] F. R. McMorris, C. Wang, and P. Zhang. On probe interval graphs. *Discrete Applied Mathematics*, 88(1-3):315–324, 1998.

[277] O. Merino. A short history of complex numbers. *University of Rhode Island*, 2006.

[278] G. B. Mertzios. The recognition of simple-triangle graphs and of linear-interval orders is polynomial. *SIAM Journal on Discrete Mathematics*, 29(3):1150–1185, 2015.

[279] G. L. Miller. Isomorphism testing for graphs of bounded genus. In *ACM Symposium on Theory of Computing*, STOC '80, pages 225–235, 1980.

[280] G.L. Miller. Isomorphism testing and canonical forms for k-contractable graphs (a generalization of bounded valence and bounded genus). In *International Conference on Fundamentals of Computation Theory*, pages 310–327. Springer, 1983.

[281] M. Miller and J. Širáň. Mnore graphs and beyond: A survey of the degree/diameter problem. *Electronic Journal of Combinatorics*, 61:1–63, 2005.

[282] B. Mohar and C. Thomassen. *Graphs on Surfaces.* Johns Hopkins University Press, 2001.

[283] T. H. Morgan. *The Theory Of The Gene.* New Haven, Yale University Press, 1928.

[284] J. Mycielski. Sur le coloriage des graphes. In *Colloq. Math*, volume 3, page 9, 1955.

[285] W. Naji. *Graphes de Cordes: Une Caracterisation et ses Applications.* PhD thesis, l'Université Scientifique et Médicale de Grenoble, 1985.

[286] B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: a maximal tractable subclass of allen's interval algebra. *J. ACM*, 42(1):43–66, 1995.

[287] T. Needham. *Visual complex analysis.* Oxford University Press, 1998.

[288] S. Negami. The spherical genus and virtually planar graphs. *Discrete Mathematics*, 70(2):159–168, 1988.

[289] Y. Otachi and P. Schweitzer. Isomorphism on subgraph-closed graph classes: A complexity dichotomy and intermediate graph classes. In *International Symposium on Algorithms and Computation*, pages 111–118. Springer, 2013.

[290] Y. Otachi and P. Schweitzer. Reduction techniques for graph isomorphism in the context of width parameters. In *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, pages 368–379, 2014.

[291] S. Oum. Rank-width and vertex-minors. *J. Comb. Theory, Ser. B*, 95(1):79–100, 2005.

[292] M. Patrignani. On extending a partial straight-line drawing. *Int. J. Found. Comput. Sci.*, 17(5):1061–1070, 2006.

[293] I. Pe'er and R. Shamir. Realizing interval graphs with size and distance constraints. *SIAM Journal on Discrete Mathematics*, 10(4):662–687, 1997.

[294] M. Pergel. Recognition of polygon-circle graphs and graphs of interval filaments is NP-complete. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 238–247. Springer, 2007.

[295] M. Pirlot. Minimal representation of a semiorder. *Theory and Decision*, 28:109–141, 1990.

[296] M. Pirlot. Synthetic description of a semiorder. *Discrete applied mathematics*, 31(3):299–308, 1991.

[297] M. Pirlot and P. Vincke. *Semiorders: Properties, representations, applications*, volume 36. Springer Science & Business Media, 2013.

[298] I.N. Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. *Journal of Soviet Mathematics*, 55(2):1621–1643, 1991.

[299] L. Pray. Discovery of DNA structure and function: Watson and Crick. *Nature Education*, 1(1):100, 2008.

[300] A. Proskurowski and J. A. Telle. Classes of graphs with restricted interval models. *Discrete Mathematics & Theoretical Computer Science*, 3(4):167–176, 1999.

[301] O. Reingold. Undirected connectivity in logspace. *Journ. of ACM*, 55(4), 2008.

[302] B. Riemann. *Grundlagen für eine allgemeine Theorie der Functionen einer veränderlichen complexen Grösse.* EA Huth, 1851.

[303] G. Ringel and J. Youngs. Solution of the Heawood map-coloring problem. *Proceedings of the National Academy of Sciences of the United States of America*, 60(2):438–445, 1968.

[304] F. S. Roberts. Indifference graphs. *Proof techniques in graph theory*, pages 139–146, 1969.

[305] F. S. Roberts. *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems.* Prentice-Hall, Englewood Cliffs, 1976.

[306] N. Robertson, D. P. Sanders, P. Seymour, and R. Thomas. Efficiently four-coloring planar graphs. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 571–575. ACM, 1996.

[307] N. Robertson and P. Seymour. Graph minors xvi. excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 77:1–27, 1999.

[308] R. W. Robinson. Enumeration of non-separable graphs. *Journal of Combinatorial Theory*, 9(4):327–356, 1970.

[309] B. Rodin and D. Sullivan. The convergence of circle packings to the riemann mapping. *J. Differential Geom*, 26(2):349–360, 1987.

[310] D. J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597–609, 1970.

[311] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SICOMP*, 5(2):266–283, 1976.

[312] J.J. Rotman. *An Introduction to the Theory of Groups.* Graduate Texts in Mathematics. Springer, 1994.

[313] G. Sabidussi. Graphs with given group and given graph-theoretical properties. *Canad. J. Math*, 9:515–525, 1957.

[314] M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs in NP. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 1–6, 2002.

[315] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226. ACM, 1978.

[316] A. A. Schäffer. A faster algorithm to recognize undirected path graphs. *Discrete Appl. Math*, 43:261–295, 1993.

[317] W. Schnyder. Planar graphs and poset dimension. *Order*, 5(4):323–343, 1989.

[318] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, 1988.

[319] E. Schrödinger. *What is Life.* Cambridge, 1944.

[320] P. Schweitzer. Towards an isomorphism dichotomy for hereditary graph classes. In *STACS 2016*, volume 30 of *LIPIcs*, 2015.

[321] D. J. Skrien. A relationship between triangulated graphs, comparability graphs, proper interval graphs, proper circular-arc graphs, and nested interval graphs. *Journal of Graph Theory*, 6(3):309–316, 1982.

[322] D. J. Skrien. Chronological orderings of interval graphs. *Discrete Appl. Math.*, 8(1):69–83, 1984.

[323] F. J. Soulignac. Bounded, minimal, and short representations of unit interval and unit circular-arc graphs. Chapter I: theory. *J. Graph Algorithms Appl*, 21(4):455–489, 2017.

[324] F. J. Soulignac. Bounded, minimal, and short representations of unit interval and unit circular-arc graphs. Chapter II: algorithms. *J. Graph Algorithms Appl*, 21(4):491–525, 2017.

[325] D. Spielman. Spectral graph theory. In *Combinatorial Scientific Computing.* Chapman and Hall / CRC Press, 2012.

[326] J. Spinrad. On comparability and permutation graphs. *SIAM J. Comput.*, 14(3):658–670, 1985.

[327] J. P. Spinrad. Recognition of circle graphs. *J. of Algorithms*, 16(2):264–282, 1994.

[328] J. P. Spinrad. *Efficient Graph Representations.* Field Institute Monographs, 2003.

[329] E. Steinitz. *Polyeder und raumeinteilungen.* Teubner, 1916.

[330] K. Stephenson. The approximation of conformal structures via circle packing. *Series in Approximations and Decompositions*, 11:551–582, 1999.

[331] J. Stillwell. *Geometry of surfaces.* Springer, 1992.

[332] K. E. Stoffers. Scheduling of traffic lights–a new approach. *Transportation Research*, 2:199–234, 1968.

[333] G. Strang. *Introduction to applied mathematics*, volume 16. Wellesley-Cambridge Press Wellesley, MA, 1986.

[334] G. Strang. *Linear algebra and its applications, 4th Edition.* Thomson, Brooks/Cole, 2006.

[335] A. Takaoka. Graph isomorphism completeness for trapezoid graphs. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 98(8):1838–1840, 2015.

[336] A. Takaoka. Recognizing simple-triangle graphs by restricted 2-chain subgraph cover. In *International Workshop on Algorithms and Computation*, pages 177–189. Springer, 2017.

[337] M. Tedder, D. Corneil, M. Habib, and C. Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. *Automata, languages and programming*, pages 634–645, 2008.

[338] M.C. Thornton. Spaces with given homeomorphism groups. *Proc. Amer. Math. Soc.*, 33:127–131, 1972.

[339] W. P. Thurston. Geometry and topology of 3-manifolds. In *Princeton lecture notes*, 1980.

[340] B.A. Trakhtenbrot. Towards a theory of non-repeating contact schemes. *Trudi Mat. Inst. Akad. Nauk SSSR*, 51:226–269, 1958.

[341] W. T. Trotter. New perspectives on interval orders and interval graphs. In *in Surveys in Combinatorics*, pages 237–286. Cambridge Univ. Press, 1997.

[342] A. Tucker. An efficient test for circular-arc graphs. *SIAM Journal on Computing*, 9(1):1–24, 1980.

[343] W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767, 1963.

[344] W. T. Tutte. *Connectivity in graphs*, volume 15. University of Toronto Press, 1966.

[345] R. Uehara. Tractabilities and intractabilities on geometric intersection graphs. *Algorithms*, 6(1):60–83, 2013.

[346] R. Uehara, S. Toda, and T. Nagoya. Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. *Discrete Applied Mathematics*, 145(3):479 – 482, 2005.

[347] M. Vilian and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proc. Fifth Nat'l. Conf. on Artificial Intelligence*, pages 337–382, 1986.

[348] V. G. Vizing. Vertex colorings with given colors. *Metody Diskret. Analiz.*, 29:3–10, 1976.

[349] J. Von Neumann and H. H. Goldstine. Numerical inverting of matrices of high order. *Bulletin of the American Mathematical Society*, 53(11):1021–1099, 1947.

[350] J. Šiagiová. A note on the McKay-Miller-Širáň graphs. *J. Combin. Theory Ser. B*, 81(2):205–208, 2001.

[351] K. Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–590, 1937.

[352] T.R.S. Walsh. Counting unlabeled three-connected and homeomorphically irreducible two-connected graphs. *J. Combin. Theory B*, 32:12–32, 1982.

[353] J.D. Watson and F.H.C. Crick. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, (4356), 1953.

[354] J. Weiner. *Time, love, memory: a great biologist and his quest for the origins of behavior.* Vintage, 2014.

[355] B. Weisfeiler and A.A. Leman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 9:12–16, 1968.

[356] B. Weisfeiler (ed.). *On Construction and Identification of Graphs*, volume 558 of *Lect. Notes in Math.* Springer, 1976.

[357] D. B. West. Parameters of partial orders and graphs: Packing, covering, and representation. In *Graphs and order*, pages 267–350. Springer, 1985.

[358] H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932.

[359] H. Whitney. Nonseparable and planar graphs. *Trans. Amer. Math. Soc.*, 34:339–362, 1932.

[360] G. Wolf. Friedrich Miescher: The man who discovered DNA. *Chemical Heritage*, 21(10-11):37–41, 2003.

[361] E. S. Wolk. The comparability graph of a tree. *Proceedings of the American Mathematical Society*, 13(5):789–795, 1962.

[362] M. Yannakakis. The complexity of the partial order dimension problem. *SIAM Journal on Algebraic Discrete Methods*, 3(3):351–358, 1982.

[363] P. Zeman. Extending partial representations of unit circular-arc graphs. *CoRR*, abs/1706.00928, 2017.

[364] P. Zhang, E. A. Schon, S. G. Fischer, E. Cayanis, J. Weiss, S. Kistler, and P. E. Bourne. An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA. *Computer applications in the biosciences: CABIOS*, 10(3):309–317, 1994.

[365] S. Zhou. A class of arc-transitive Cayley graphs as models for interconnection networks. *SIAM Journal on Discrete Mathematics*, 23(2):694–714, 2009.

Index