Karlsruhe Institute of Technology

On Layered Drawings of Planar Graphs

Bachelor Thesis of

Sarah Lutteropp

At the Department of Informatics
Institute of Theoretical Computer Science

Reviewers:	Prof. Dr. Maria Axenovich
	Prof. Dr. Dorothea Wagner
Advisors:	Dipl.-Inform. Thomas Bläsius
	Dr. Tamara Mchedlidze
	Dr. Torsten Ueckerdt

Time Period: 28th January 2014 – 28th May 2014

Acknowledgements

I would like to thank my advisors Thomas Bläsius, Dr. Tamara Mchedlidze and Dr. Torsten Ueckerdt for their great support in form of many hours of discussion, useful ideas and extensive commenting on iterative versions.

It was not trivial to find a topic and place for a combined mathematics and computer science thesis and I am grateful to Prof. Dr. Dorothea Wagner, who allowed me to write my thesis at her institute.

I would also like to thank Prof. Dr. Dorothea Wagner and Prof. Dr. Maria Axenovich for grading my thesis.

Last but not least, I would like to thank all my proofreaders (supervisors included) for giving last-minute comments on my thesis.

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Karlsruhe, 28th May 2014

Abstract

A graph is *k-level planar* if it admits a planar drawing in which each vertex is mapped to one of k horizontal parallel lines and the edges are drawn as non-crossing y-monotone line segments between these lines.

It is not known whether the decision problem of a graph being k-level planar is solvable in polynomial time complexity (in $\mathcal{P}$) or not. However, if the layer assignment, i.e. the mapping of each vertex to a line, is given, we can construct a level planar embedding in linear time if it exists. In case there is no level planar embedding, one can try to minimize the total number of crossings, which is known to be $\mathcal{NP}$-complete.

It is easy to construct an example of a planar graph with a layer assignment that results in a non level planar graph. Motivated by this fact we investigate the situation where it is allowed to perform as few modifications to the layer assignments as possible in order to obtain a level planar drawing. We give an extensive overview of the current state of research related to level planarity and provide a heuristic for changing the layer assignment as little as possible by swapping the layer assignments of pairs of vertices or moving vertices to other layers (with or without introducing new layers) in order to get a level planar result.

Deutsche Zusammenfassung

Ein Graph heißt *k-level planar*, falls man ihn so in der Ebene zeichnen kann, dass die Knoten auf k horizontalen parallelen Geraden angeordnet sind. Die Kanten dürfen sich nicht kreuzen und verlaufen als y-monotone Kurven zwischen diesen Geraden.

Die Menge aller Knoten auf der gleichen Geraden heißt Schicht. Es ist unklar, ob die Frage, ob ein gegebener Graph k-level planar ist in $\mathcal{P}$ liegt oder nicht. Falls die Zuordnungen von Knoten auf Geraden im Voraus bekannt sind, kann mit linearem Zeitaufwand eine solche Einbettung gefunden werden, falls sie existiert. Wenn keine solche Einbettung existiert, kann man versuchen, die Anzahl Kreuzungen zu minimieren. Dies ist bekannterweise $\mathcal{NP}$-vollständig.

In dieser Arbeit versuchen wir, möglichst wenige der vorgegebenen Schichtzuordnungen zu verändern, sodass wir eine level planare Zeichnung des Graphen erhalten. Wir geben einen ausführlichen Überblick zum aktuellen Stand der Forschung zu level planarity und entwickeln eine Heuristik, die versucht durch möglichst wenige Vertauschungen oder Verschiebungen von Knoten zwischen Schichten eine level planare Zeichnung des gegebenen Graphen zu ermöglichen.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Organization of the Thesis . . . 4

2 Preliminaries **5**
2.1 Main Terminology . . . 5
2.2 More Mathematical Background . . . 14
2.2.1 Well-Known Theorems about Planar Graphs . . . 14
2.2.2 Hasse Diagrams . . . 15
2.2.3 Branchwidth . . . 18
2.3 Problem Definitions . . . 20
2.3.1 Known Problems . . . 20
2.3.2 Newly Defined Problems . . . 23
2.3.3 Complexity Overview . . . 24

3 Related Work **27**
3.1 (Extended) Level Planarity Testing . . . 27
3.1.1 Testing Level Planarity . . . 27
3.1.2 Testing Extended Level Planarity . . . 30
3.2 Characterization via Forbidden Patterns . . . 31
3.3 Crossing Minimization . . . 31
3.4 Level Planarity and k-Level Planarity Problems . . . 32
3.5 The Sugiyama Method . . . 34
3.6 Further Related Work . . . 34

4 New Considerations **37**
4.1 Bounds and Complexity . . . 38
4.1.1 Changing and Swapping Layer Assignments . . . 38
4.1.2 Moving Vertices . . . 40
4.2 The Rotation Heuristic . . . 41
4.2.1 Compute the Distance . . . 41
4.2.2 Rotate the Drawing . . . 46
4.2.3 Time Complexity . . . 47
4.3 More Ideas for Heuristics . . . 47

5 Conclusion **49**
5.1 Open Problems and Further Work . . . 49

Bibliography **51**

1. Introduction

1.1 Motivation

In many applications like VLSI-layout[Len90], UML-diagrams and network analysis, human-readable representations of graphs are required. It is important that the main information contained in a graph is clearly visible. Depending on the application, a drawing of a graph has to satisfy certain properties. In order to avoid visual clutter, the drawing is often required to contain as few crossings as possible. For directed graphs it is often required that its drawing emphasizes its underlying hierarchical structure.

The most commonly used technique to draw graphs that represent hierarchical information is given by Sugiyama et al.[STT81]. The method is usually applies to an directed acyclic graph and requires that the vertices are placed in horizontal layers and the edges point downwards. An example of a Sugiyama-style drawing is given in Figure 1.1.

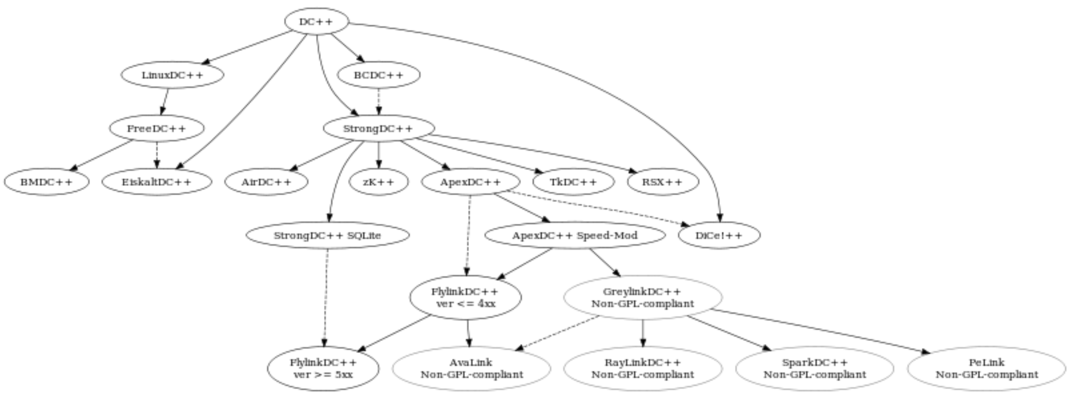


Figure 1.1: A Graph showing the DC++ derivates tree, image from `http://tehnick-8.narod.ru/dc_clients/`.

Ideally, one would like to have an *upward planar* drawing, i.e., a planar drawing of the graph where all edges point in the same direction (without loss of generality upwards). However, this is not always possible, even if the graph itself is *planar* (i.e., it admits a planar drawing), as shown in Figure 1.2.

upward planar

planar

Sugiyama's method works in four steps: First, the input graph is transformed into a directed acyclic graph by reversing edges that cause a directed cycle. In the second step, the vertices

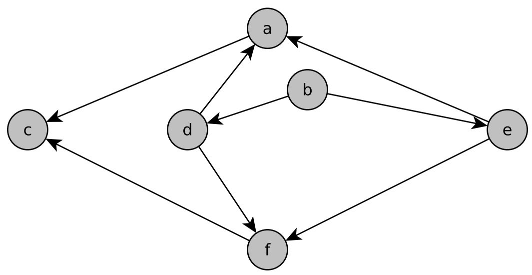


Figure 1.2: A planar directed acyclic graph which is not upward planar since its source b and its sink c cannot be both in the same face.

are placed on horizontal layers, i.e., their vertical position in the resulting embedding is fixed. Next, the number of crossings is minimized by moving the vertices horizontally, thus fixing their horizontal placement. Afterwards, the final drawing is produced and previously reversed edges are put back into their original orientation. Most of these steps contain $\mathcal{NP}$-complete problems, thus they are solved heuristically.

What we do in this Thesis

In this thesis, we concentrate on applications that provide the (initial) layering as part of the input called *leveled graphs*. If the vertices can be permuted within levels to obtain a planar drawing the leveled graph is called *level planar*. While it can happen that the Sugiyama method produces a non-planar drawing of a planar graph, we insist on a drawing without crossings. In order to achieve that, we allow to reassign some of the vertices to other layers than given initially. This is a novel approach to visualize leveled graphs. After explaining the terms needed and giving an extensive overview of related research, we develop heuristics for changing the initial layering as little as possible in order to obtain a level planar drawing of the input graph.

An Example for Layered Graph Drawing

Imagine the following collaboration network in a company (Figure 1.3): People working in the same project are assigned to the same layer. Edges between people on different layers represent further communication between them. Our target is to find a nice visualization of this network.

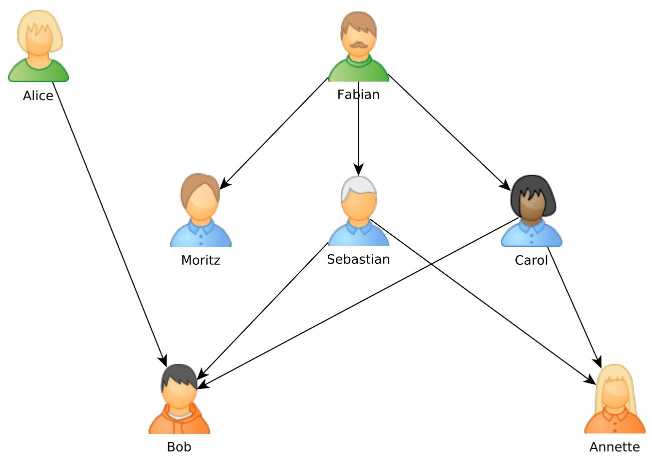


Figure 1.3: Company hierarchy

The company's hierarchy can be modeled as a leveled graph $(G = (V, E), \phi)$ where the vertices represent the people and the edges represent neccessary connections between them. The function $\phi : V \to \{1, \ldots, k\}$ maps each person to a layer. The resulting leveled graph can be seen in Figure 1.4.

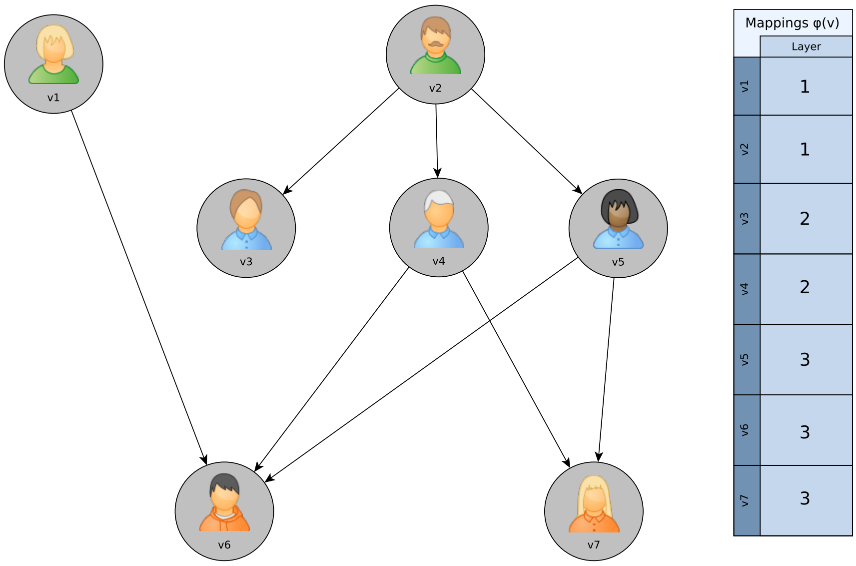


Figure 1.4: Modeling of the company's situation

We now want to find a left-to-right order of vertices which are on the same layer. This left-to-right order for all layers is called an *embedding* of (G, ϕ). When we draw G with lines monotonic in y-direction such that the y-coordinates of the vertices are given by ϕ and their sorting by x-coordinates on the same layer is given by the embedding, we want that no lines cross. If there exists an embedding of (G, ϕ) such that the edges are drawn as curves, monotonic in y-direction, and no edges cross, then the graph (G, ϕ) is called *level planar*. This is a formal phrasing of the definition used earlier.

It is known that it is not always possible to obtain a planar drawing of a planar graph with regard to a given layering of its vertices [DL11], even if the graph itself is upward planar. An easy example is shown in Figure 1.5.

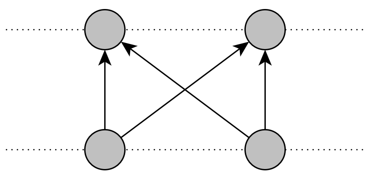


Figure 1.5: An upward planar graph G with a layering ϕ of G such that (G, ϕ) is not level planar.

In current research one tries to keep the layering as it is and minimize the number of crossings. However, in our situation we insist on obtaining a drawing without crossings. Thus, we need to modify the given layering, which means that some of the people working in the same project will not be drawn on the same horizontal line.

Since we colored the peoples' suits by their projects, it is still visible who works in which project after changing the layering. A possible planar drawing of the company's situation is shown in Figure 1.6.

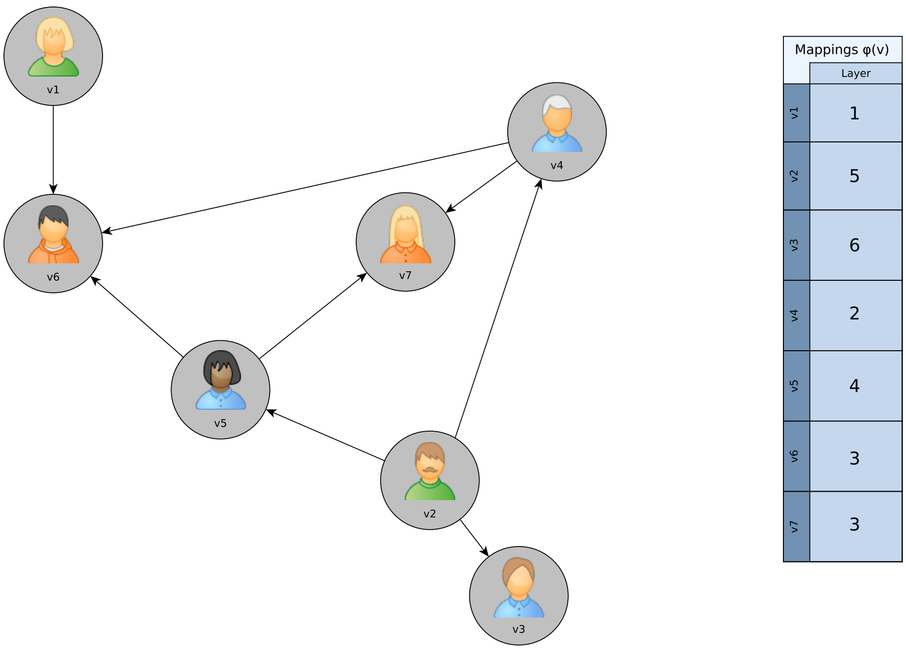


Figure 1.6: A planar drawing of the company's situation

1.2 Organization of the Thesis

In Chapter 2, we give basic terms and notions and define new and related problems. In these new problems, we try to minimize the total number of layer reassignments, swaps between layer assignments of pairs of vertices or moves of vertices to arbitrary layers or newly introduced layers. The current state of related research is summed up in Chapter 3. In Chapter 4, we try to analyze the complexity of the newly defined problems and discuss ideas for heuristics for them. We also introduce a quartic-time heuristic that takes a planar straight-line drawing of the graph and rotates it until it induces a layer assignment which is similar to the one given as part of the input. We conclude the thesis in Chapter 5, where we give a view on further work and open problems.

2. Preliminaries

2.1 Main Terminology

Graphs

A *graph* is an ordered pair $G = (V_G, E_G)$ containing a set V_G of *vertices* or *nodes* and a set E_G of *edges*. An edge is a pair of two vertices. If the edges are represented as ordered pairs (u, v), the graph is *directed*, else the graph is *undirected* and edges are represented as unordered pairs $\{u, v\}$. A directed edge (u, v) from vertex u to vertex v is an *outgoing edge* of vertex u and an *incoming edge* of vertex v. A *source* is a vertex in a directed graph which has no incoming edges. A *sink* is a vertex in a directed graph which has no outgoing edges.

graph
outgoing edge
incoming edge
source
sink

If not explicitly mentioned otherwise, the following definitions will be given for undirected graphs. The definitions can be translated to directed graphs easily.

Two vertices are *adjacent* if there exists an edge between them. An edge is *incident* to a vertex if the edge connects the vertex with another vertex in the graph. The *degree* of a vertex v is the number of incident edges to v (i.e., the number of edges which have v as one of their endpoints).

adjacent
incident
degree

An undirected graph is *complete* if every pair of distinct vertices is connected by exactly one edge. The complete graph on n vertices is denoted by K_n. An undirected graph $G = (V_G, E_G)$ is *bipartite* if V_G can be partitioned into two sets V_1 and V_2 every edge in E_G connects a vertex from V_1 with a vertex from V_2. The *complete bipartite* graph $K_{n,m}$ on $n + m$ vertices is a bipartite graph $G = (V_1 \dot\cup V_2, E_G)$ where $|V_1| = n, |V_2| = m$ and every vertex of V_1 is adjacent to every vertex of V_2.

complete
K_n
bipartite
complete bipartite
$K_{n,m}$

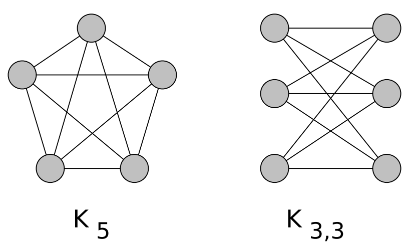


Figure 2.1: K_5 and $K_{3,3}$

subgraph *induced* A graph $H = (V_H, E_H)$ is called a *subgraph* of a graph $G = (V_G, E_G)$ if $V_H \subseteq V_G$ and $E_H \subseteq E_G$. A subgraph H of G is *induced* if every edge in E_G connecting two vertices in V_H is also contained in E_H.

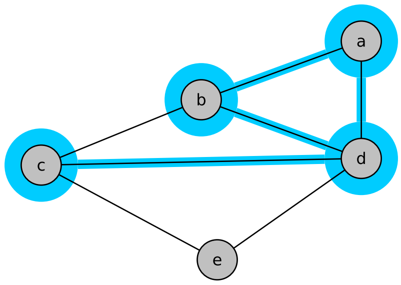


Figure 2.2: The blue marked subgraph is not an induced subgraph since the edge $\{b, c\}$ is missing.

isomorphic Two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are *isomorphic* if there exists a bijection $\sigma : V_G \to V_H$ such that $\{v_1, v_2\} \in E_G \Leftrightarrow \{\sigma(v_1), \sigma(v_2)\} \in E_H$.

minor *contraction* An undirected graph H is called a *minor* of a graph G if H can be constructed out of G by deleting vertices, deleting edges and contracting edges. A *contraction* of an edge is done by deleting the edge and merging the two vertices connected by this edge into a new vertex. The edges incident to one of these two vertices in the old graph will then be incident to the newly introduced vertex in the new graph. Note that vertices incident to both contracted vertices give rise to only one edge in the minor. An example for a minor of a graph is given in Figure 2.3.

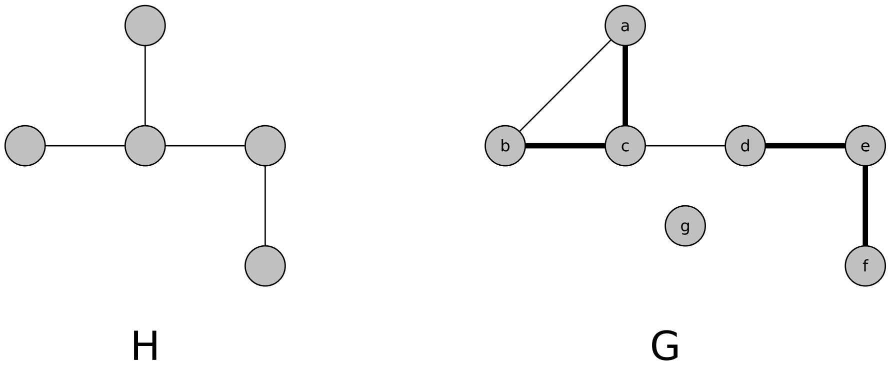


Figure 2.3: The graph H can be formed out of G by deleting the edge $\{a, b\}$, contracting the edge $\{c, d\}$ and deleting the vertex g. Thus, H is a minor of G.

subdivision *homeomorphic* A *subdivision* of an edge $\{u, v\}$ is performed by introducing a new vertex w and replacing the edge by two new edges connecting w to u and v. A subdivision of a graph results from performing zero or more subdivisions of its edges. Two graphs G and H are *homeomorphic* if a subdivision of G is isomorphic to a subdivision of H.

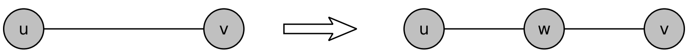


Figure 2.4: Subdividing the edge $\{u, v\}$ results in a new vertex w and two new edges $\{u, w\}$ and $\{w, v\}$.

A *path* $P_k = (V_k, E_k)$ of size or length k is defined as a set V_k of $k+1$ distinct vertices $\{v_1, \ldots, v_{k+1}\}$ and a set E_k of k edges $\{\{v_i, v_{i+1}\} \mid i = 1, \ldots, k\}$. The vertices v_1 and v_{k+1} are said to be *connected* by P_k. A path P_4 is shown in Figure 2.5.

path
connected

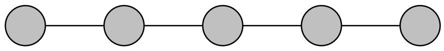


Figure 2.5: A path of size 4.

A *cycle* $C_k = (V_k, E_k)$ of size or length k is defined as a set V_k of k distinct vertices $\{v_1, \ldots, v_k\}$ and a set E_k of k edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_k, v_1\}$. A cycle C_6 is shown in Figure 2.6.

cycle

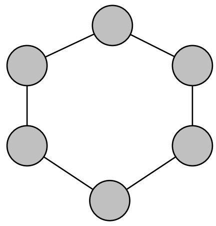


Figure 2.6: A cycle of size 6.

A graph G *contains* a graph H if H is a subgraph of G. Especially, a graph G contains a path P_k (a cycle C_k) if P_k (or C_k) is a subgraph of G.

contains

A graph is *acyclic* if it does not contain a cycle.

acyclic

Two vertices u and v in a graph G are *connected* if G contains a path that connects u and v. A graph is *connected* if all of its vertices are pairwise connected, otherwise it is *disconnected*. A graph is k-connected if the removal of any $k-1$ vertices results in a connected graph. A *connected component* of a graph G is a maximal connected subgraph of G.

connected
connected component

A *forest* is an acyclic graph. A connected forest is called a *tree*. A connected subgraph of a tree is called a *subtree*.

forest
tree
subtree

A *rooted tree* T is a directed tree where one vertex is explicitly chosen to be the *root* of T. The edges are oriented "away from the root". In a rooted tree, the *children* of a vertex u are all vertices v that are connected by an edge (u, v) in T. A *leaf node* is a vertex which has no children. In an unrooted tree, a *leaf node* is a vertex with degree one. An *inner node* is a vertex which is not a leaf.

rooted tree
children
leaf node
leaf node
inner node

Drawings

A *drawing* δ of a graph is obtained by mapping its vertices to distinct points in $\mathbb{R}^2$ and representing its edges by simple curves connecting the respective points. If all edges are drawn as straight-line segments, the drawing itself is called *straight-line*. A *planar* drawing is a drawing where no edges intersect except for at common endpoints. A graph is called *planar* if it admits a planar drawing. A *triangulated* graph is an edge-maximal planar graph, i.e., adding any edge to the graph would destroy its planarity.

drawing
planar
triangulated

A $n \times m$ *grid* Γ is defined by a set of points $p = (i, j)$ in the plane where $i = 1, \ldots, n$ and $j = 1, \ldots, m$ are the x-, respective y-coordinates, of p. The parameter n is known as the *width* of Γ and the parameter m is called the *height* of Γ. A $k \times l$ *grid-drawing* of a graph G is a drawing of G where its vertices are places on points of the $k \times l$ grid. The *height* of

$n \times m$ grid
width
height
$k \times l$ grid-drawing
height

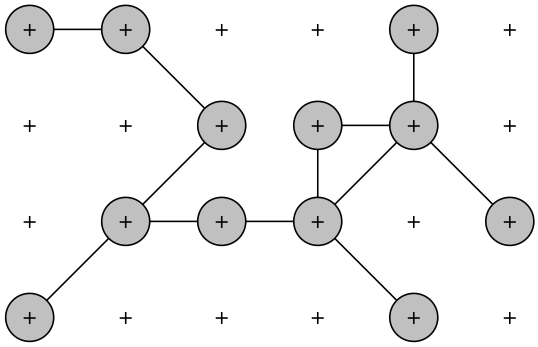


Figure 2.7: A graph drawn on the 6×4 grid.

a $k \times l$ grid-drawing is the minimum of k and l. An example of a grid-drawing is given in Figure 2.7.

faces *outer face* *inner faces*

A planar drawing of a graph splits the plane into *faces* which are connected regions of the plane after the removal of all vertices and edges. One of these faces is unbounded and is called the *outer face*, the other faces are called *inner faces*. An example is given in Figure 2.8.

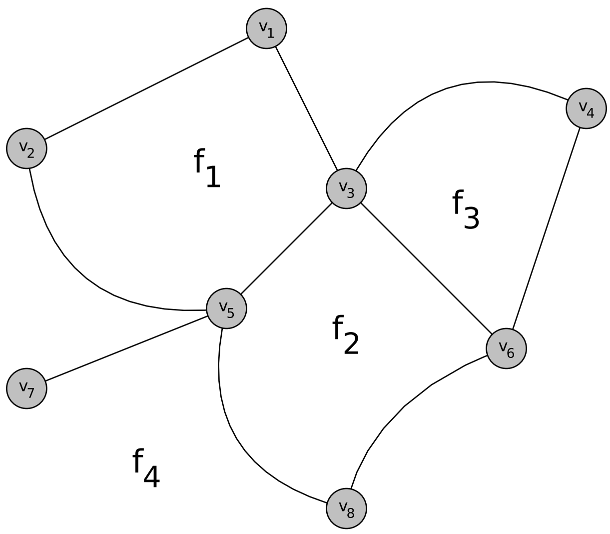


Figure 2.8: The faces of a planar drawing of a planar graph G. The face f_4 is the outer face, the faces f_1, f_2, f_3 are the inner faces.

outerplanar *y-monotone* *weakly y-monotone*

A graph is *outerplanar* if it admits a planar drawing where all vertices lie on the outer face of the drawing. For example, the graph given in Figure 2.8 is outerplanar. A drawing is *y-monotone* if each edge crosses every horizontal line at most once. A drawing is *weakly y-monotone* if each edge either crosses every horizontal line at most once or lies on a horizontal line. See Figure 2.9a-d for examples of different drawings of the same graph.

equivalent *embedding*

Two drawings are *equivalent* if their topological properties coincide. An *embedding* of a graph is an equivalence class of drawings of the graph.

rotation system

A *rotation system*, sometimes also called *combinatorial embedding*, of a graph is given by the clockwise order of incident edges at each vertex.

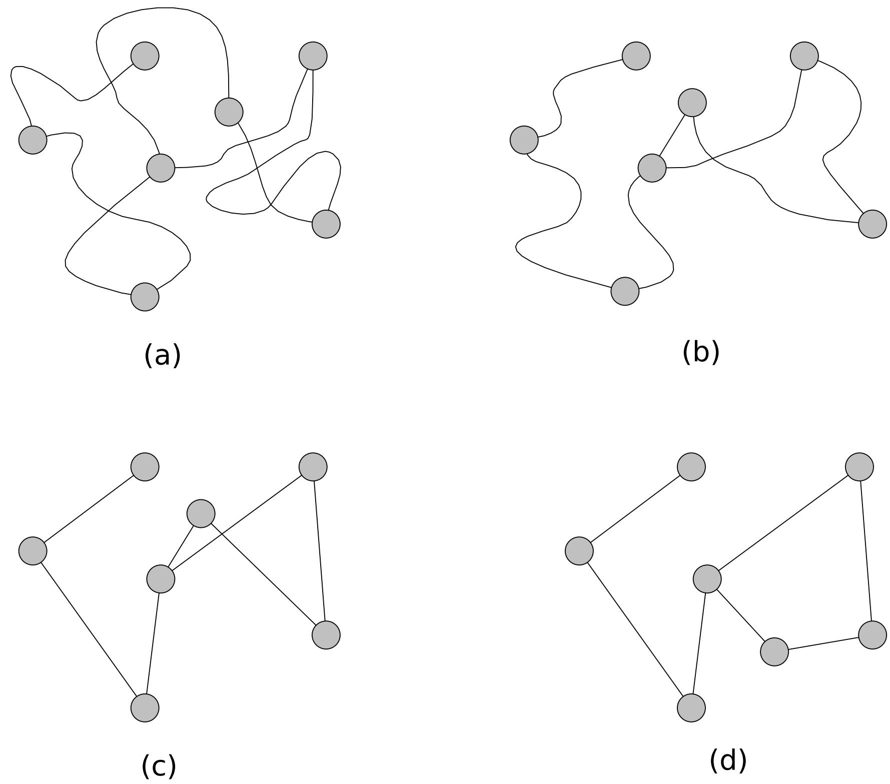


Figure 2.9: Several drawings of the same planar graph G. (a) A drawing of G. (b) A y-monotone drawing of G. (c) A straight-line drawing of G. (d) A planar straight-line drawing of G.

Level Planarity

A *k-leveled graph* is an ordered pair (G, ϕ) consisting of a graph $G = (V_G, E_G)$ and a surjective function $\phi : V_G \to \{1, \dots, k\}$. The function ϕ maps each vertex of G to one of k horizontal parallel lines such that adjacent vertices are mapped to different lines (i.e., for each edge (u, v) it holds that $\phi(u) \neq \phi(v)$). The set of vertices mapped to the same line is then called a *layer* and ϕ is called a *layering* or *layer assignment* of G. A layering is *proper* if the vertices of each edge are mapped to adjacent layers, i.e., for an edge connecting vertex u with vertex v it holds that $|\phi(u) - \phi(v)| = 1$. The *height* of a layering $\phi : V_G \to \{1, \dots, k\}$ of a graph $G = (V_G, E_G)$ is k, i.e., the total number of layers. Its *width* is defined as the maximum number of vertices which are mapped to the same layer, i.e. $\max_{i=1,\dots,k} |\phi^{-1}(i)|$. An example of a leveled drawing of height 5 is given in Figure 2.10.

k-leveled graph

layer

layering

proper

height

width

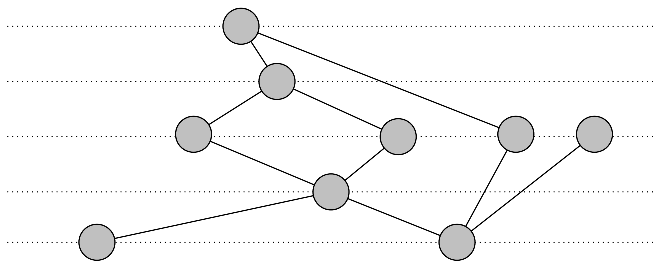


Figure 2.10: A 5-leveled drawing of a 5-leveled graph (G, ϕ). The layering ϕ has height 5 and width 4. The leveled graph (G, ϕ) is level planar. This especially means that G is 5-level planar.

k-leveled drawing
k-level planar drawing

A *k-leveled drawing* of a k-leveled graph (G, ϕ) is an y-monotone drawing of G which respects ϕ, i.e. vertices on the same layer are mapped to points with the same y-coordinate and layer i is above layer $i+1$ for all i from 1 to $k-1$. When k is not essential, we talk about leveled drawing. A *k-level planar drawing* of a k-leveled graph (G, ϕ) is a k-leveled drawing of (G, ϕ) which is planar.

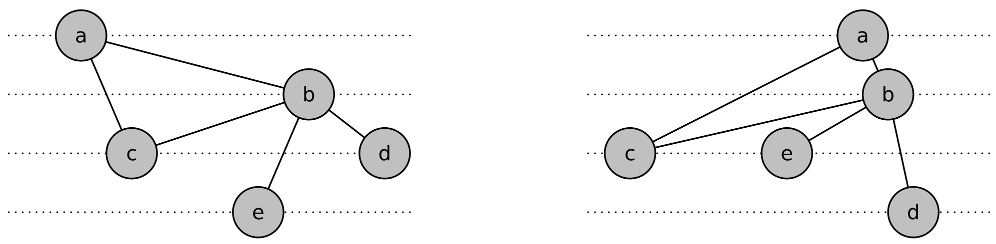
k-level planar
k-level planar
planar

A k-leveled graph (G, ϕ) is *k-level planar* if there exists a k-leveled drawing of it which is planar. A graph G is *k-level planar* if there exists a layering ϕ of height k of G such that (G, ϕ) is k-level planar. A k-leveled graph (G, ϕ) is *planar* if G is planar.

leveled embedding

For leveled graphs, a *leveled embedding* is defined by the order of vertices on the same layer. A leveled embedding represents an equivalence class of leveled drawings. A leveled embedding of a k-leveled graph is called *k-level planar* if there exists a k-leveled planar drawing of G such that the order of vertices on the same layer is the same as given by the leveled embedding.

A rotation system of a graph G does not induce an unique layering of G. An example is shown in Figure 2.11.



(a) A leveled drawing of (G, ϕ). (b) A leveled drawing of (G, ψ).

Figure 2.11: Both leveled drawings have the same rotation system. Thus, a rotation system does not fix the layering.

It is crucial to note that a leveled embedding of a graph G does not uniquely determine a rotation system of G, as shown in Figure 2.12.

(a) A leveled drawing of (G, ϕ). (b) Another leveled drawing of (G, ϕ).

Figure 2.12: Two leveled drawings of the same leveled embedding. The clockwise order of the edges at vertex a differs. Thus, a leveled embedding does not uniquely determine a rotation system of G.

Bachmaier et al.[BBFH10] introduce the term *extended level drawing* which stands for a leveled drawing where edges on the same layer are allowed.

extended layering
short

An *extended layering* is a layering without the restriction that adjacent vertices are not mapped to the same layer, i.e., edges whose endpoints are on the same layer are allowed. An extended layering ϕ is *short* if all adjacent vertices u and v are mapped to the same or *extended k-leveled graph* *extended k-leveled drawing* *extended k-level planar drawing* *extended k-level planar* *extended k-level planar* *planar*

to adjacent layers, i.e., $|\phi(u) - \phi(v)| \leq 1$. An *extended k-leveled graph* is a pair (G, ϕ) of a graph $G = (V_G, E_G)$ and an extended layering $\phi : V_G \to \{1, \ldots, k\}$ of G. An *extended k-leveled drawing* of an extended k-leveled graph (G, ϕ) is a weakly y-monotone drawing of G which respects ϕ, i.e., the vertices of G are mapped to layers according to ϕ. An *extended k-level planar drawing* of an extended k-leveled graph (G, ϕ) is an extended k-leveled drawing of (G, ϕ) which is planar. An extended k-leveled graph (G, ϕ) is *extended k-level planar* if there exists an extended k-leveled drawing of it which is planar. A graph G is *extended k-level planar* if there exists an extended layering ϕ of height k of G such that (G, ϕ) is extended k-level planar. An extended k-leveled graph (G, ϕ) is *planar* if G is planar. The concepts of extended level planarity and minimum height of a planar grid-drawing of a planar graph are the same if the extended layering is not part of the input.

y-ordering *unlabeled level planar*

A *y-ordering* of an n-vertex graph $G = (V_G, E_G)$ is a bijective layering $\gamma : V_G \to \{1, \ldots, n\}$ of G, i.e. every layer contains exactly one vertex. As introduced by Estrella-Balderama et al.[EBFK09a], a graph G is *unlabeled level planar* if (G, γ) is n-level planar for all $n!$ possible y-orderings γ of G.

Treewidth, Pathwidth and Branchwidth

For a graph G, we denote by $V(G)$ the set of vertices of G and by $E(G)$ the set of edges of G.

tree decomposition

As introduced by Rudolf Halin[Hal76] and rediscovered by Robertson and Seymour[RS84], a *tree decomposition* of an undirected graph $G = (V_G, E_G)$ is a pair (T, τ) where $T = (V_T, E_T)$ is a tree and τ is a function that maps each vertex of T to a subgraph of G such that the following conditions hold:

1. $\bigcup_{t \in V_T} \tau(t) = G$.
2. For distinct $t_1, t_2 \in V_T$, let (V_1, E_1) be the subgraph of T given by $\tau(t_1)$ and let (V_2, E_2) be the subgraph of T given by $\tau(t_2)$. It holds that the graph $(V_1 \cap V_2, E_1 \cap E_2)$ is empty.
3. For all $t_1, t_2, t_3 \in V_T$, if t_2 is on the path between t_1 and t_3 then $V(\tau(t_1)) \cap V(\tau(t_3)) \subseteq V(\tau(t_2))$ and $E(\tau(t_1)) \cap E(\tau(t_3)) \subseteq E(\tau(t_2))$.

width *treewidth* *bags*

The *width* of a tree decomposition (T, τ) is the maximum of $|V(\tau(t))| - 1$, taken over all $t \in V_T$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. The sets $\tau(t)_{t \in V_T}$ are called *bags*. In some definitions, $\tau(t)$ is defined to be a subset of vertices of G for $t \in V_T$. In that case, we refer to $\tau(t)$ as an induced subgraph of G.

An example of a tree decomposition of a graph is given in Figure 2.13.

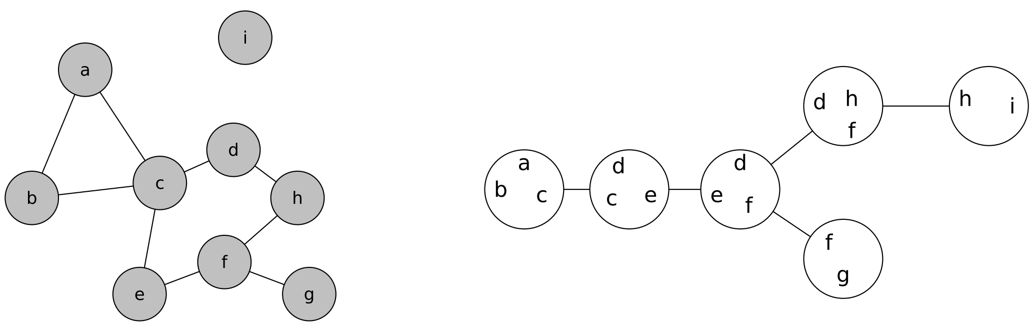


Figure 2.13: A graph G with a tree decomposition of width 2.

path decomposition

As introduced by Robertson and Seymour[RS83], a *path decomposition* of an undirected graph G is a tree decomposition of G where the underlying tree is a path. The *pathwidth* of G is the minimum width over all path decompositions of G.

pathwidth

normalized

Let $(P = (p_1, \ldots, p_k), \rho)$ be a path decomposition of an undirected graph G with pathwidth h. The path decomposition is *normalized* if the following two conditions hold:

1. $|V(\rho(p_i))| = \begin{cases} h+1 & \text{, if } i \text{ is odd,} \\ h & \text{, if } i \text{ is even.} \end{cases}$
2. $V(\rho(p_{i-1})) \cap V(\rho(p_{i+1})) = V(\rho(p_i))$ and $E(\rho(p_{i-1})) \cap E(\rho(p_{i+1})) = E(\rho(p_i))$ for even i.

Given a path decomposition of an undirected graph, a normalized path decomposition of the graph with the same width can be computed in linear time as shown by Gupta et al [GNPR05].

branch decomposition

As also introduced by Robertson and Seymour [RS91], a *branch decomposition* of an undirected graph $G = (V, E)$ is a pair (T, τ) where $T = (V_T, E_T)$ is a tree with vertices of degree 1 or 3 and τ is a bijective function which maps every edge of G to a leaf of T. Removing an edge e of T partitions T into two subtrees T_1 and T_2. This partition induces a partition of the edges in E into two subsets E_1 and E_2. This operation is called an *e-separation* of G. The *width* of an e-separation is the number of vertices of G which are incident both to an edge in E_1 and to an edge in E_2. The *order* of an edge $e \in E_T$ is the width of its corresponding e-separation.

e-separation

width

order

width

branchwidth

The *width* of a branch-decomposition is the maximum width over all of its e-separations. The *branchwidth* of an undirected graph G is the minimum width over all branch decompositions of G. If G has less than two edges, G has no branch decomposition and G is said to have branchwidth zero. An example of a branch decomposition of a graph is given in Figure 2.14.

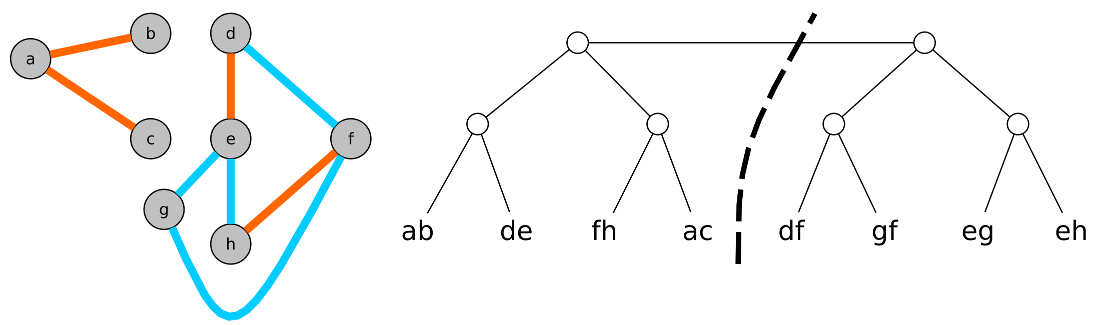


Figure 2.14: A graph with a branch decomposition, showing an e-separation of width 4.

Courcelle [Cou90] and Arnborg et al. [AL91] show that for graphs with bounded branch-width, some dynamic programming algorithms require only polynomial time, even though the problems solved by them are $\mathcal{NP}$-complete for general graphs. These results were originally shown for graphs with bounded treewidth, but Robertson and Seymour [RS91] show that branchwidth and treewidth are closely related to each other as they differ only by a constant factor (see Theorem 2.9). While computing the branchwidth for general graphs is $\mathcal{NP}$-complete, there exist polynomial-time algorithms if the input graph is planar as shown by Seymour and Thomas [ST94].

PQ-Tree

A *PQ-Tree* is a data structure introduced by Booth and Luecker [BL76] that represents a family of permutations over a set S of n elements. PQ-Trees are used in applications that require to find an ordering with respect to a set of constraints. A PQ-Tree T over a set S is a rooted tree with three types of nodes: P-Nodes, Q-Nodes and leaf nodes. The leaf nodes of T are in bijection with the elements of S whereas the P-Nodes and Q-Nodes are inner nodes of T and have a set of possible linear orderings of their children. An example of a PQ-Tree is given in Figure 2.15. *PQ-Tree*

The children of inner nodes of T can be reordered according to the following rules.

A *P-Node* has at least three children. Its children can be arbitrarily reordered. *P-Node*

A *Q-Node* has at least two children. Its children can only be ordered in two ways, which are the reversals of one another. *Q-Node*

Each left-to-right order of the leaves of the PQ-Tree that can be achieved by performing these reorderings represents a permutation of the elements of S. Not every set of permutations can be represented by a PQ-Tree since the reverse of an ordering is always represented by the same PQ-Tree. For example, there is no PQ-Tree which represents the permutation *abc* but not the permutation *cba*.

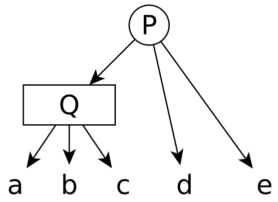


Figure 2.15: A PQ-Tree T over the set $S = \{a, b, c, d, e\}$ that represents the permutations *abcde*, *cbade*, *dabce*, *dcbae*, *eabcd*, *ecbad*, *edabc*, *edcba*, *deabc*, *decba*, *abced* and *cbaed*.

There is one operation on a PQ-Tree T, namely $REDUCE(T, X)$, which uses pattern matching to create a new PQ-Tree that restricts the permutations represented by T such that the elements of $X \subseteq S$ appear in consecutive order in any permutation represented by T. An example of a result of a *REDUCE*-operation is given in Figure 2.16.

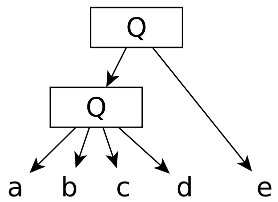


Figure 2.16: This PQ-Tree results from $REDUCE(T, \{c, d\})$ where T is the PQ-Tree from Figure 2.15. It represents the permutations *abcde*, *dcbae*, *eabcd* and *edcba*.

Since the pattern matching contains a lot of cases, working with PQ-Trees can be difficult. Shih and Hsu [kSH99] introduce an alternative data structure, called the *PC-Tree*. The PC-Tree is mainly the result of making the PQ-Tree unrooted which introduces symmetry such that the aforementioned cases reduce to one case. *PC-Tree*

2.2 More Mathematical Background

2.2.1 Well-Known Theorems about Planar Graphs

The following four theorems represent characterizations of planar graphs.

Theorem 2.1 (Kuratowski's [Kur30] theorem)**.** *A graph G is planar if and only if it does not contain a subgraph which is a subdivision of K_5 or $K_{3,3}$.*

Theorem 2.2 (Wagner's [Wag37] theorem)**.** *A graph G is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor.*

Well-written proofs of Kuratowski's theorem and Wagner's theorem can be found at [Ote14] and [wag14], respectively.

Theorem 2.3 (Euler's [Eul58] formula)**.** *Let G be a nonempty connected undirected planar graph. Let δ be a planar drawing of G with f faces, n vertices and e edges. Then $n - e + f = 2$.*

Proof. The following proof has been taken literally from [Muh14].

We do an induction on the number of edges e.

$e = 0$ Since G is connected, $e = 0$ means that $n = 1$ and $f = 1$. Thus, the formula holds for $e = 0$.

$e \to e + 1$ If G is a tree, then $n = e + 1$ and $f = 1$ so the formula holds. If G is not a tree, then G must contain a cycle. Let e_c be an edge on this cycle. By induction hypothesis, we know that the formula holds for the graph $G - e_c$ (i.e., G without the edge e_c). Adding the edge e_c to $G - e_c$ adds 1 edge, 0 vertices and 1 face since e_c causes a cycle. Thus, the formula holds for G.

□

Corollary 2.4 (Number of edges in a planar graph)**.** *Let $n \geq 3$. Every undirected n-vertex planar graph $G = (V, E)$ has at most $3n - 6$ edges.*

Proof. The following proof has been taken mostly literally from [Rud14].

We do an induction on the number of vertices n.

$n = 3$ For $n = 3$, the statement is clearly true since there can be at most 3 edges.

$n \to n + 1$ Consider a planar graph G with a maximal number of edges. If G is not connected, then we can add an edge which contradicts the maximality of G. Thus, G is connected and it holds that $n - e + f = 2$. Every face is bounded by exactly 3 edges due to maximality and every edge lies on the boundary of exactly 2 faces, thus $2e = 3f$. This leads to the following equations:

$$n - e + f = 2$$

$$3n - 3e + 3f = 6$$

$$3n - 3e + 2e = 6$$

$$e = 3n - 6$$

Since we constructed G with maximal number of edges, in general it holds that $e \leq 3n - 6$.

□

While by definition a planar graph can be represented on the plane with curvy edges and no crossing, it is not immediately clear whether the same is possible with straight-line edges. The following theorem answers the question in affirmative.

Theorem 2.5 (Fáry's [Far48] theorem)**.** *Every planar graph has a straight-line planar drawing.*

Although Fáry's theorem is named after István Fáry, it was proven independently by Wagner [Wag36] in 1936, Fáry [Far48] in 1948 and Stein [Ste51] in 1951. A well-written proof of Fáry's theorem can be found in [CLZ10].

2.2.2 Hasse Diagrams

weak partial order

A *weak partial order* over a set S is a binary relation "$\leq$" that is reflexive, antisymmetric and transitive, i.e. the following conditions hold:

Reflexivity $x \leq x$ for all $x \in S$

Antisymmetry $x \leq y$ and $y \leq x \Rightarrow x = y$ for all $x, y \in S$

Transitivity $x \leq y$ and $y \leq z \Rightarrow x \leq z$ for all $x, y, z \in S$

weak partially ordered set

The pair $(S, \leq)$ is called a *weak partially ordered set*. For example, $(\mathbb{R}, \leq)$ (with the standard less-than-or-equal relation) is a weak partially ordered set and for a set S, $(\mathcal{P}(S), \subseteq)$ is a weak partially ordered set.

strict partial order

A *strict partial order* over a set S is a binary relation "$<$" that is irreflexive, antisymmetric and transitive, i.e. the following conditions hold:

Irreflexivity not $x < x$ for all $x \in S$

Asymmetry If $x < y$ then not $y < x$ for all $x, y \in S$

Transitivity $x < y$ and $y < z \Rightarrow x < z$ for all $x, y, z \in S$

strict partially ordered set

The pair $(S, \leq)$ is called a *strict partially ordered set*.

Every weak partial order corresponds to a strict partial order and vice versa. If "$\leq$" is a weak partial order, then the corresponding strict partial order "$<$" is given by $a < b$ if $a \leq b$ and $a \neq b$. Similarly, if "$<$" is a strict partial order, then the corresponding weak partial order "$\leq$" is given by $a \leq b$ if $a < b$ or $a = b$. Thus, while we consider weak partial orders in this subsection, our definitions and considerations can easily be translated to strong partial orders.

We notice that a weak partially ordered set $(S, \leq)$ induces a directed acyclic graph (DAG) G where the vertices of G represent elements of S and two vertices u and v are connected by an edge (u, v) if $u \leq v$.

transitive closure

The *transitive closure* $C(G)$ of a directed graph G is a graph which contains an edge (u, v) whenever there is a directed path from u to v in G (definition taken literally from [Ski90]). An example for the transitive closure of a directed graph is given in Figure 2.17.

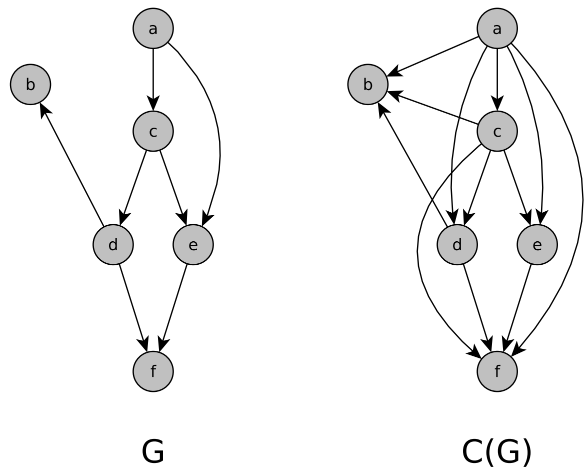


Figure 2.17: A finite directed acyclic graph G and its transitive closure $C(G)$.

transitive reduction

The *transitive reduction* $T(G)$ of a finite directed acyclic graph (DAG) G is an edge-minimal subgraph of G such that the reachability stays the same. This means, if two vertices u and v are connected by a path in G, then they are also connected by a path in $T(G)$. So, $C(T(G)) = C(G)$. An example of a transitive reduction is shown in Figure 2.18.

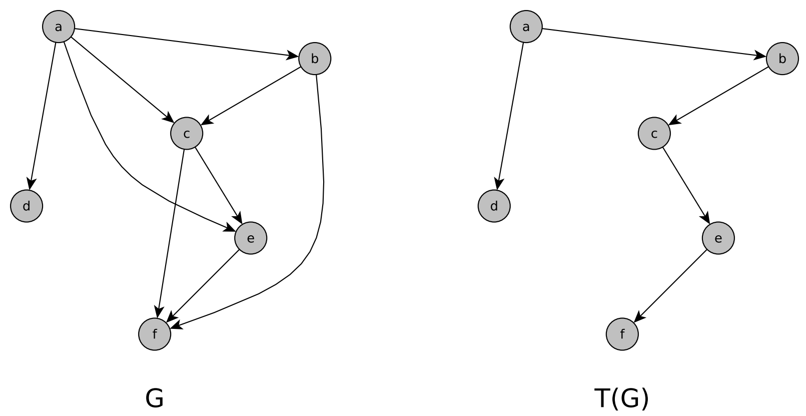


Figure 2.18: A finite directed acyclic graph G and its transitive reduction $T(G)$.

Theorem 2.6 (Moyles and Thompson [MT69])**.** *The transitive reduction of a finite directed acyclic graph is unique.*

superfluous

Proof. An edge $e = (i, j)$ is called *superfluous* if there is a directed i-to-j path not using e.

Observation 2.7. *For every G its subgraph H consisting of all non-superfluous edges has the same transitive closure as G.*

Observation 2.8. *For every G every transitive reduction contains all non-superfluous edges.*

Every non-superfluous edge $e = (u, v)$ has to be contained in a transitive reduction $T(G)$ since the removal of e destroys the property that there exists a path that connects u to v. Hence, the unique transitive reduction of G is the graph consisting of all non-superfluous edges. □

Hasse diagram

The *Hasse diagram* H of a weak partially ordered set $(S, \leq)$ is the directed acyclic graph obtained by performing a transitive reduction on the DAG induced by $(S, \leq)$. Thus, the vertices of H are the elements of S and we have an edge from vertex u to vertex v if $u \leq v$ and there is no other vertex w such that $u \leq w \leq v$. An example of a Hasse Diagram is given in Figure 2.19.

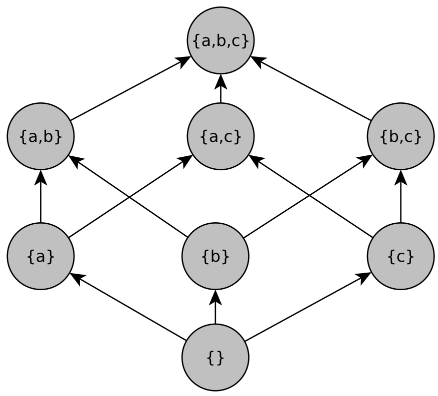


Figure 2.19: A Hasse diagram for $(\mathcal{P}(\{a, b, c\}), \subseteq)$.

When it comes to visualizing a Hasse diagram, we prefer a drawing without crossings where the elements of S appear in a hierarchical order. Sometimes, the elements of S naturally induce a layering. For example, in Figure 2.19, we want sets of the same cardinality to be drawn on the same layer. Thus, Hasse diagrams can become an application for level planar drawings.

comparability graph

For any weak partially ordered set $(S, \leq)$, the *comparability graph* $C(S, \leq) = (V_C, E_C)$ is defined as $V_C = S$ and for $s_1, s_2 \in S$, $\{s_1, s_2\} \in E_C$ if $s_1 \leq s_2$. An example of a comparability graph is given in Figure 2.20.

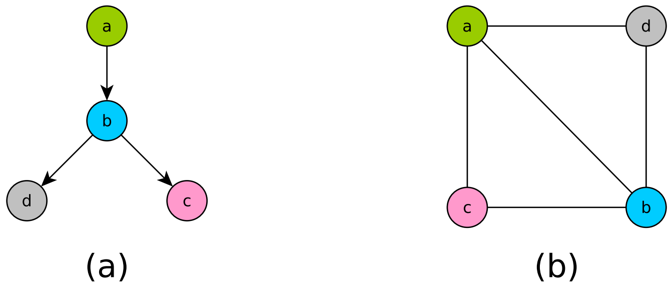


Figure 2.20: (a) A Hasse diagram of a partially ordered set. (b) The corresponding comparability graph.

2.2.3 Branchwidth

Recall the definitions of treewidth and branchwidth in 2.1.

Theorem 2.9 (Robertson, Seymour [RS91]). *For an undirected graph with treewidth t and branchwidth b we have $b - 1 \leq h \leq \lfloor \frac{3}{2} b \rfloor$ - 1.*

Proof. The following proof is taken mostly literally from [RS91], page 168f.

Let $G = (V_G, E_G)$ be an undirected graph with treewidth h and branchwidth b. If $V = \emptyset$, then both h and b are zero and thus the statement holds. If $V \neq \emptyset$ and $|E| \leq 1$, then $b = 0$ and $h = 1$ and thus the statement also holds. Assume that $V \neq \emptyset$ and $|E| \geq 2$. Since the removal of isolated vertices (i.e., vertices with degree 0) does not change h or b, we assume that there are no isolated vertices in G.

We show the second inequality first. Let $(T = (V_T, E_T), \tau)$ be a branch decomposition of G of width b. We construct a tree decomposition (T, σ) of G of width at most $\lfloor \frac{3}{2} b \rfloor - 1$.

For each $t \in V_T$ we define a subgraph $\sigma(t) = (V_t, E_t)$ of V_G as follows:

- If t is a leaf of T, let $\sigma(t)$ consist of the edge $\tau(t)$ and its ends.
- If t is not a leaf of T, let V_t consist of the vertices v of G for which there are edges f and g both incident to v such that t lies on the path of T between $\tau^{-1}(f)$ and $\tau^{-1}(g)$. Let E_t be the empty set. An example of this step is shown in Figure 2.21.

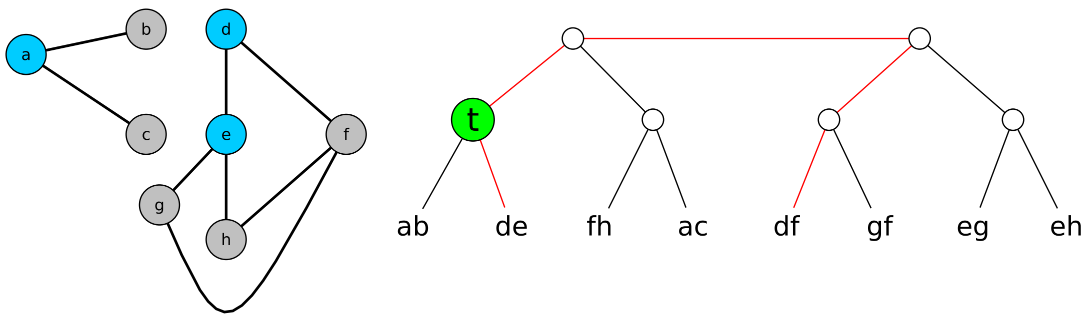


Figure 2.21: On the left: A graph G. The set V_t consists of the vertices a, d and e. On the right: A branch decomposition of G. The path of T between $\tau^{-1}(\{d, e\})$ and $\tau^{-1}(\{d, f\})$ is marked in red.

It is easy to verify that (T, σ) is a tree decomposition of G. If t is a leaf of T, $|\sigma(t)| = 2$. If t is not a leaf of T, let e_1, e_2, e_3 be the edges of T incident to t. For any $v \in \sigma(t)$, v contributes to the order of at least two of e_1, e_2, e_3 and so $2 \cdot |\sigma(t)| \leq 3 \cdot b$. Thus, this tree decomposition has width $\leq \lfloor \frac{3}{2} b \rfloor - 1$ and the second inequality holds.

Now we show the first inequality. Let (T, τ) be a tree decomposition of G of width h. We first show that we can transform (T, τ) into a new tree decomposition of G of width h such that the following assumptions (1)–(3) hold.

(1) We can assume that for each edge $e \in E_G$, there is a leaf t of T where $\tau(t)$ consists of edge e only and hence that $\tau(t)$ has no edges for each $t \in V_T$ with degree ≥ 2.

If for some $e \in E_G$ there is no such t, we choose $t' \in V_T$ such that $G(\tau(t'))$ contains e as an edge. We add a new vertex t to T adjacent only to t' and define $\tau(t')$ to consist of the ends of e. We remove e from $\tau(t')$. This results in a new tree decomposition of G of width h. By repeating this step, we can construct a tree decomposition of G such that (1) holds.

(2) We can assume that $|E(\tau(t))| = 1$ for each leaf t of T.

By (1), $|E(\tau(t))| \leq 1$. If $\tau(t)$ contains no edges, we construct a tree decomposition of G width h as follows. Let $T' = (V_{T'}, E_{T'})$ be obtained by deleting t from T, and let τ' be the restriction of τ to $V_{T'}$. Since G has no isolated vertices, it follows that (T', τ') is a tree decomposition of G of width h that satisfies (1). By repeating this process, we can ensure that (2) holds.

(3) We can assume that every vertex of T has degree ≤ 3.

If t has degree greater than 3, we choose a tree $T' = (V_{T'}, E_{T'})$ and an edge $f \in E_{T'}$ such that T is obtained from T' by contracting f, and the two ends t_1, t_2 both have a smaller degree than t. We define $\tau(t_1) = \tau(t) = \tau(t_2)$. The new tree decompositiuon still has width h and satisfies (1) and (2). By repeating this process, we can ensure that (3) holds.

Now let $\sigma(t)$ be E_t for each leaf t of T. Let $S = (V_S, E_S)$ be the tree obtained from T by suppressing each vertex of degree 2, i.e. contracting one of its incident edges. Then, (S, σ) is a branch decomposition of G. For $f \in E_S$, the order of f is at most the number of vertices in $\tau(t)$, where t is an end of f, and hence the first inequality holds. □

2.3 Problem Definitions

In this section we introduce the problem of transforming a planar leveled graph into a level planar graph by performing changes on the layer assignment. Before that we will give a precise formulation of the problems known in literature which are either closely related to the problem of level assignment or were proven to be useful when dealing with those related problems. All of these problems will be mentioned again in Section 3 or used in the proofs.

2.3.1 Known Problems

Planar 3-SAT

satisfiable
Boolean satisfiability problem

A boolean formula $\mathcal{F}$ is *satisfiable* if there exists an assignment of its variables to boolean values (*TRUE* or *FALSE*) such that $\mathcal{F}$ is *TRUE*. The *Boolean satisfiability problem* (SAT) is to check whether a given boolean formula is satisfiable or not. It was shown to be the first $\mathcal{NP}$-complete problem by Cook [Coo71] in 1971.

boolean literal
clause
conjunctive normal form

A *boolean literal* is either a variable or a negated variable. A *clause* is a disjunction of boolean literals. A boolean formula is said to be in *conjunctive normal form* (CNF) if it is a conjunction of clauses.

3-SAT formula

A 3-*SAT formula* is a boolean formula in conjunctive normal form such that each clause consists of at most three literals.

variable-clause-graph

The *variable-clause-graph* of a 3-SAT formula consists of:

- A vertex for each variable.
- A vertex for every clause.
- Each variable x is connected to the clauses containing x or $\neg x$ by an edge.

An example of a variable-clause-graph is given in Figure 2.22.

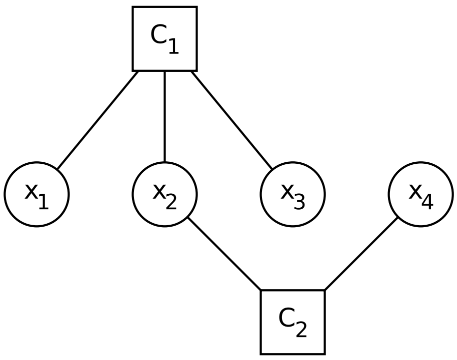


Figure 2.22: The variable-clause-graph of $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_4)$. Notice that the formula $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4)$ has the same variable-clause-graph.

planar 3-SAT formula

A *planar* 3-*SAT formula* is a 3-SAT formula whose variable-clause graph is planar.

> PLANAR 3-SAT PROBLEM
> INPUT: A planar 3-SAT formula F
> QUESTION: Is F satisfiable?

Lichtenstein [Lic82] shows that the PLANAR 3-SAT PROBLEM is $\mathcal{NP}$-complete.

Maximum Independent Set

Given an undirected graph $G = (V_G, E_G)$, an *independent set* of G is a set $D \subseteq V_G$ if there is no edge in E_G between any two vertices in D. A *maximum independent set* $D_{\max}$ of G is an independent set of G with maximum size, i.e. there is no other independent set D' of G with $|D'| > |D_{\max}|$. An example of a maximum independent set is given in Figure 2.23.

independent set

maximum independent set

> MAXIMUM INDEPENDENT SET PROBLEM
> INPUT: An undirected graph G, an integer i
> QUESTION: Does G has a maximum independent set of size $\geq i$?

As shown by Cook [Coo71], the MAXIMUM INDEPENDENT SET PROBLEM is $\mathcal{NP}$-complete.

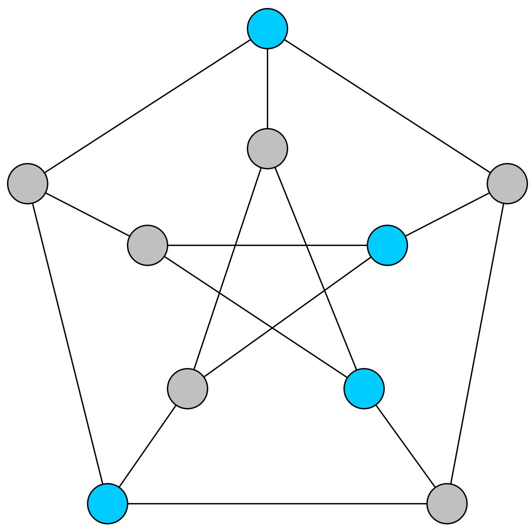


Figure 2.23: A maximum independent set of size 4 in the Petersen [Pet98] Graph.

Level Planarity

> PROPER LEVEL PLANARITY PROBLEM
> INPUT: An undirected graph G
> QUESTION: Does a proper layering ϕ of G exist such that (G, ϕ) is level planar?

Heath and Rosenberg [HR89] show via a reduction from the PLANAR 3-SAT PROBLEM that the PROPER LEVEL PLANARITY PROBLEM is $\mathcal{NP}$-complete.

> (EXTENDED) K-LEVEL PLANARITY PROBLEM
> INPUT: A graph G, an integer k
> QUESTION: Does a layering ϕ of height k of G exist such that (G, ϕ) is (extended) k-level planar?

To our knowledge, the complexity of the (EXTENDED) K-LEVEL PLANARITY PROBLEM is still unknown.

The EXTENDED K-LEVEL PLANARITY PROBLEM is the same as asking whether a graph G has a planar weakly y-monotone grid drawing of height k.

(EXTENDED) LEVEL PLANARITY TESTING PROBLEM
INPUT: An (extended) leveled graph (G, ϕ)
QUESTION: Is (G, ϕ) (extended) level planar?

Both the LEVEL PLANARITY TESTING PROBLEM and the EXTENDED LEVEL PLANARITY TESTING PROBLEM for the class of extended leveled graphs with a bounded number of isolated components are in $\mathcal{P}$ as explained below.

Jünger, Leipert and Mutzel [JLM98] provide a linear algorithm for testing whether a given leveled graph is level planar. Hong and Nagamochi [HN09] provide an $\mathcal{O}(|E|+|V|^{p+1} \cdot (2/p)^p)$ algorithm for testing extended level planarity of an extended leveled graph (G, ϕ) with p isolated components. If (G, ϕ) has no isolated components, their algorithm takes $\mathcal{O}(|E|+|V|)$ time. More information about the testing algorithms will be given in Section 3.1.

Crossing Minimization

(EXTENDED) CROSSING MINIMIZATION PROBLEM
INPUT: An (extended) leveled graph (G, ϕ), an integer c
QUESTION: Does an (extended) level drawing of (G, ϕ) with at most c crossings exist?

Garey and Johnson [GJ83] show that the CROSSING MINIMIZATION PROBLEM is $\mathcal{NP}$-complete. Bachmeier et al. [BBFH10] show the $\mathcal{NP}$-completeness of the EXTENDED CROSSING MINIMIZATION PROBLEM.

Maximum Level Planar Subgraph

edge-maximal planar subgraph
edge-maximum planar subgraph

While an *edge-maximal planar subgraph* of a graph $G = (V_G, E_G)$ is a graph $H = (V_H, E_H)$ such that adding an edge $e \in E_G \backslash E_H$ to E_H would destroy planarity, an *edge-maximum planar subgraph* is an edge-maximal subgraph J with the maximum number of edges, i.e. there is no other edge-maximal planar subgraph which contains more edges than J.

MAXIMUM (EXTENDED) LEVEL PLANAR SUBGRAPH PROBLEM
INPUT: An (extended) leveled graph (G, ϕ), an integer j
QUESTION: Does an edge-maximum (extended) level planar subgraph of G contain at least j edges?

MAXIMUM INDUCED (EXTENDED) LEVEL PLANAR SUBGRAPH PROBLEM
INPUT: An (extended) leveled graph (G, ϕ), an integer h
QUESTION: Does an maximum induced (extended) level planar subgraph of G contain at least h vertices?

Eades and Whitesides [EW94] show that the MAXIMUM LEVEL PLANAR SUBGRAPH PROBLEM is $\mathcal{NP}$-complete. Since every k-level planar graph is also extended k-level planar, it is easy to see that the EXTENDED MAXIMUM LEVEL PLANAR SUBGRAPH PROBLEM is also $\mathcal{NP}$-complete.

To our knowledge, the complexity of the MAXIMUM INDUCED (EXTENDED) LEVEL PLANAR SUBGRAPH PROBLEM has not been analyzed so far.

Straight-line Drawings of Planar Graphs

By *untangling* a non-planar straight-line drawing of a planar graph we understand mapping some of the vertices to other positions in the plane such that the resulting new straight-line drawing is planar. *untangling*

> UNTANGLING PLANAR GRAPH PROBLEM
> INPUT: An undirected planar graph G, a straight-line drawing δ of G, an integer i
> QUESTION: Can we untangle δ by moving at most i vertices in the plane?

Goaoc et al. [GKO$^+$09] show that the UNTANGLING A PLANAR GRAPH PROBLEM is $\mathcal{NP}$-complete via a reduction from the PLANAR 3-SAT PROBLEM.

> PARTIAL DRAWING EXTENSIBILITY PROBLEM
> INPUT: A planar undirected graph $G = (V_G, E_G)$ and a mapping between a subset V' of its vertices and a set of distinct points on the plane.
> QUESTION: Can coordinates be assigned to the vertices in $V \backslash V'$ such that the resulting straight-line drawing of G is planar?

Patrignani [Pat06] shows that the PARTIAL DRAWING EXTENSIBILITY PROBLEM is $\mathcal{NP}$-complete. He shows the $\mathcal{NP}$-hardness via a reduction from the PLANAR 3-SAT PROBLEM.

2.3.2 Newly Defined Problems

In this thesis, we are interested in the following problems.

Reassigning Vertices

By *reassigning* the layer assignment of a vertex v, we mean the change of the value $\phi(v)$. As a result, v is then mapped to another already existing layer. *reassigning*

> (EXTENDED) LAYER REASSIGN PROBLEM
> INPUT: An (extended) planar leveled graph (G, ϕ), an integer r
> QUESTION: Can we get an (extended) level planar graph from (G, ϕ) by changing the layer assignments of at most r vertices in ϕ?

Swapping Layer Assignments

By *swapping* the layer assignments of two vertices u and w, we understand that the vertex u gets mapped to $\phi(v)$ and the vertex v gets mapped to the former $\phi(u)$. *swapping*

> (EXTENDED) LAYER SWAP PROBLEM
> GIVEN: An (extended) planar leveled graph (G, ϕ), an integer s
> QUESTION: Can we obtain an (extended) level planar graph from (G, ϕ) by performing at most s swaps in ϕ?

Moving Vertices

By *moving* a vertex v, we understand that either the layer assignment of v is changed or a new layer L is introduced and v is then assigned to L. The newly introduced layer is then inserted above, between or below the already existing layers. *moving*

(EXTENDED) VERTEX MOVE PROBLEM
INPUT: An (extended) planar leveled graph (G, ϕ), an integer m
QUESTION: Can we get an (extended) level planar graph from (G, ϕ) by moving at most m vertices in ϕ?

2.3.3 Complexity Overview

Table 2.1 sums up the complexity results of the problems mentioned in this thesis.

Problem	Time complexity	Shown by
Planar 3-SAT Problem	A	Lichtenstein [Lic82]
Maximum Independent Set Problem	A	Cook [Coo71]
Proper Level Planarity Problem	A	Heath and Rosenberg [HR89]
k-Level Planarity Problem	C	
Extended k-Level Planarity Problem	C	
Level Planarity Testing Problem	B	Juenger, Leipert and Mutzel [JLM98]
Extended Level Planarity Testing Problem	E	Hong and Nagamochi [HN09]
Crossing Minimization Problem	A	Garey and Johnson [GJ83]
Extended Crossing Minimization Problem	A	Bachmeier et al. [BBFH10]
Maximum Level Planar Subgraph Problem	A	Eades and Whitesides [EW94]
Maximum Extended Level Planar Subgraph Problem	A	Trivial result
Maximum Induced Level Planar Subgraph Problem	C	
Maximum Induced Extended Level Planar Subgraph Problem	C	
Untangling Planar Graph Problem	A	Goaoc et al. [GKO+09]
Partial Drawing Extensibility Problem	A	Patrignani [Pat06]
Layer Reassign Problem	C	
Extended Layer Reassign Problem	D	This thesis
Layer Swap Problem	C	
Extended Layer Swap Problem	D	This thesis
Vertex Move Problem	C	
Extended Vertex Move Problem	C	

Table 2.1: Complexity results. A : $\mathcal{NP}$-complete, B : $\mathcal{P}$, C : unknown, D : unknown but $\mathcal{NP}$-complete if the Extended k-Level Planarity Problem is $\mathcal{NP}$-complete, E : unknown but in $\mathcal{P}$ if the number of isolated components is bounded by a constant.

3. Related Work

Much research has been done regarding level planarity. We divide this chapter into several main aspects. In Section 3.1, we discuss known algorithms for testing (extended) level planarity of a given (extended) leveled graph. Motivated by K_5 and $K_{3,3}$ as forbidden patterns for planarity, there have been attempts to define forbidden patterns for level planarity, too. The results of these attempts are summarized in Section 3.2. If an (extended) leveled graph is not (extended) level planar, one can try to minimize crossings between the layers. Section 3.3 is devoted to known results about (extended) crossing minimization. Section 3.4 deals with the case when the (extended) layering is not part of the input, i.e. it has to be computed first. We give an overview on results known about minimizing the height of an (extended) level planar drawing.

A common framework for creating layered drawings of directed graphs is the Sugiyama framework. In Section 3.5, we describe the idea of that framework. Section 3.6 contains further related work which is needed in Chapter 4.

3.1 (Extended) Level Planarity Testing

3.1.1 Testing Level Planarity

The input of known level planarity testing algorithms consists of a directed graph $G = (V(G), E(G))$ and a layering φ with the property that the source of an edge is mapped to a lower layer than the sink of an edge, i.e. for each edge $(u, v) \in E$ it holds that $\varphi(u) < \varphi(v)$. If we want to test an undirected graph with a given layering for level planarity, we can direct the edges of it first and then use one of the following algorithms.

3.1.1.1 Testing with PQ-Tree

Jünger, Leipert and Mutzel [JLM98] provide a linear algorithm for testing whether a given leveled graph for is level planar using PQ-Trees. Jünger and Leipert [JL99] extend this algorithm to also compute a leveled embedding of a level planar graph in linear time.

The main idea of their algorithm is to perform a layer-by-layer sweep and store the set of possible permutations of vertices on the same layer for the current layer in a PQ-Tree.

Despite its best known time complexity, there exist other approaches for testing level planarity which are more intuitive and easier to implement.

3.1.1.2 Testing with Labeled Vertex Exchange Graph

A quadratic time testing and embedding algorithm which is easier to implement and uses a datastructure called the *labeled vertex-exchange graph* was presented by Healy and Kuusik [HK99b] [HK99a]. Healy and Kuusik require the layering ϕ to be *proper*, which means that for each edge $(u, v) \in E(G)$ it holds that $|\phi(u) - \phi(v)| = 1$. Like in their paper, we will denote an ordered pair of vertices as $[u, v]$.

The labeled vertex-exchange graph $\mathcal{VE}(\pi) = (\mathcal{V}, \mathcal{E}, \mathcal{L})$ of a leveled embedding π of a directed graph G is defined as follows. The set vertices $\mathcal{V}$ of the labeled vertex-exchange graph consists of all ordered pairs of distinct vertices of G which are on the same layer. Their ordering is induced by the given leveled embedding, i.e. the pair $[u, v]$ is contained in $\mathcal{V}$ if u and v are on the same layer and $\pi(u) < \pi(v)$, i.e., u comes before v in the given leveled embedding.

Two pairs $[u, v]$ and $[x, y]$ in $\mathcal{V}$ are connected by an edge in $\mathcal{E}$ if their vertices are not on the same layer and either $(u, x) \in E(G)$ and $(v, y) \in E(G)$ or $(u, y) \in E(G)$ and $(v, x) \in E(G)$.

corresponding edges

We understand by the *corresponding edges* of an edge $\{[u, v], [x, y]\} \in \mathcal{E}$ either (u, x) and (v, y) or (u, y) and (v, x). In case that both $(u, x), (v, y) \in E(G)$ and $(u, y), (v, x) \in E(G)$, there are two edges $\{[u, v], [x, y]\}$ in $\mathcal{E}$, one that corresponds to (u, x) and (v, y) and one that corresponds to (u, y) and (v, x).

The function $\mathcal{L}$ labels each edge $\{[a, b], [c, d]\} \in \mathcal{E}$ with '–' or '+', depending on whether its two corresponding edges in the given leveled embedding of G cross or not. In Figure 3.1, an example of a graph with corresponding labeled vertex-exchange graph is given.

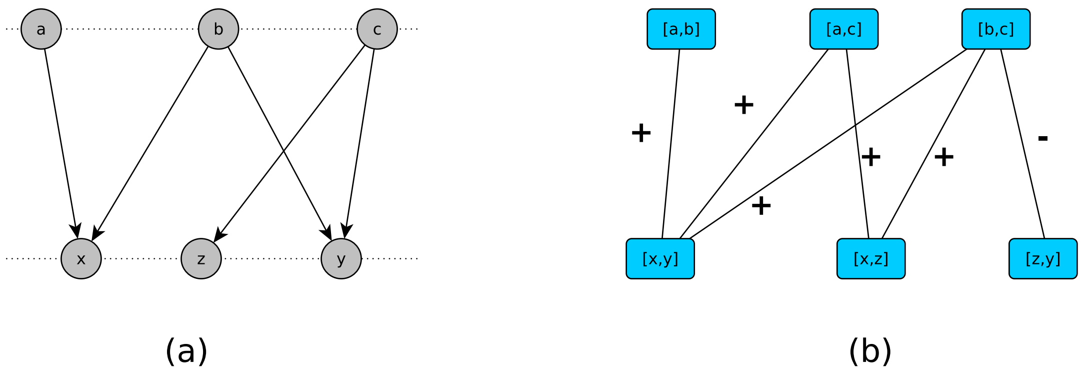


Figure 3.1: (a) A leveled drawing of a leveled embedding ϵ of a directed graph G, (b) The labeled vertex-exchange graph $\mathcal{VE}(\epsilon)$

No-3-cycles property

Healy and Kuusik observe that a labeled vertex exchange graph has the *No-3-cycles property*. The No-3-cycles property states that vertex orderings correspond to linear orderings, i.e. for vertices x, y and z of G, the orderings $[x, y]$, $[y, z]$, $[z, x]$ are invalid since this implies $\pi(x) < \pi(y) < \pi(z) < \pi(x)$ for a leveled embedding π.

In order to use the labeled vertex-exchange graph to test level planarity, Healy and Kuusik give the following lemma.

Lemma 3.1 (Healy, Kuusik [HK99b],[HK99a])**.** *If a leveled graph with a proper layering is not level planar, then for each leveled embedding π of it it holds that $\mathcal{VE}(\pi)$ contains '–'-labeled edges.*

Healy and Kuusik start with an arbitrary leveled embedding of the leveled graph provided by the input. In order to deal with other leveled embeddings, they modify the labeled vertex-exchange graph via a so-called *ve-operation*. A ve-operation $ve([u, v])$ switches the labels of all edges incident to $[u, v]$ in the labeled vertex-exchange graph, i.e. '+' becomes '–' and '–' becomes '+'. This corresponds to reversing the order of the two vertices in the leveled embedding, thus $[u, v]$ becomes $[v, u]$. *ve-operation*

A sequence S of ve-operations on a labeled vertex-exchange graph is *valid* if the labeled vertex-exchange graph constructed by S still has the No-3-cycles property. I.e., a sequence of ve-operations on a labeled vertex-exchange graph $\mathcal{VE}(\pi_1)$ is valid if it transforms π_1 into a leveled embedding π_2. *valid*

Theorem 3.2 (Healy, Kuusik [HK99b],[HK99a])**.** *A leveled graph (G, ϕ) with a proper layering ϕ is level planar if and only if for any leveled embedding π of it, there exists some valid sequence of ve-operations that removes all '–'-labeled edges from $\mathcal{LVE}(\pi)$ or equivalently, $\mathcal{LVE}(\pi)$ does not contain a cycle with an odd number of '–'-labeled edges.*

Idea of the Proof.

"⇒:" A level planar graph has a level planar embedding, i.e. no edges are crossing there. Thus, the labeled vertex-exchange graph of this embedding does not contain any '–'-labeled edges.

"⇐:" If there is a cycle with an odd number of '–'-labeled edges, we cannot perform a sequence of ve-operations such that there are no '–'-labeled edges afterwards. Else, there is always such a sequence. We need to show that this sequence is also valid, at least in case that the leveled graph is level planar.

Assume that we get a 3-cycle $[a, b]$, $[b, c]$ and $[c, a]$ after performing these ve-operations. There are several cases how this can happen, one example is given in Figure 3.2 where $[x_1, x_2]$ and $[y_1, y_2]$ represent the vertex pair on the highest or lowest layer of the paths. However, the graph constructed in this example is not level planar. The authors claim that for each possible configuration that leads to a 3-cycle problem, the proof can be done in a similar way.

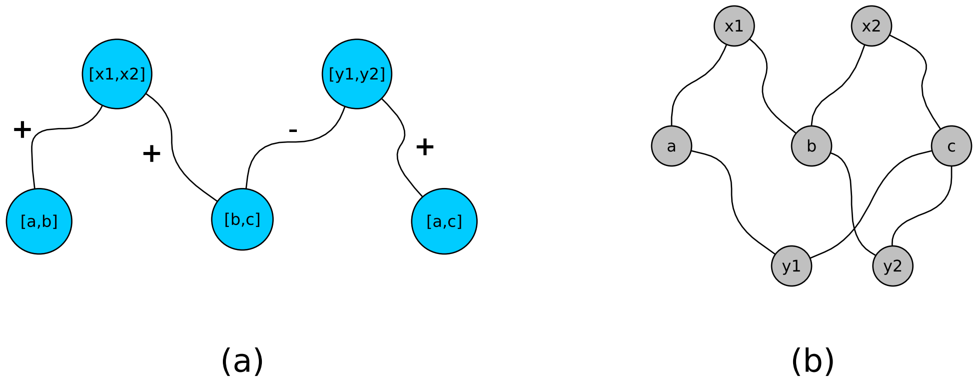


Figure 3.2: In this figure, paths are represented by curved edges. The label '–' on a path means that there is an odd number of edges labeled with '–'. (a) A labeled vertex-exchange graph that leads to the 3-cycle problem. (b) An embedding of a graph corresponding to this labeled vertex-exchange graph.

□

Based on this theorem, the authors give a Depth-First-Search algorithm for testing a given directed graph with proper layering for level planarity. In order to compute a level planar embedding, they provide an additional level-by-level traversal that has to be done after level planarity testing since their Depth-First-Search algorithm does not necessarily compute a valid sequence of ve-operations.

Fulek et al. [FPSS13] claim that there is no proof of Theorem 3.2 in the case that the labeled vertex-exchange graph is not connected. Thus, the characterization of level planarity that Healy and Kuusik give is not yet established.

3.1.1.3 Testing with Hanani-Tutte

y-monotone

In the definition by Fulek et al.[FPSS13], a drawing is *y-monotone* if every edge crosses each horizontal line at most once **and every horizontal line contains at most one vertex**.

Fulek et al.[FPSS13] show that testing whether a leveled graph (G, ϕ) is level planar can be reduced to testing y-monotonicity (in their paper they use x-monotonicity since they regard vertical lines instead of horizontal ones), resulting in a quadratic-time algorithm.

Theorem 3.3 (Weak Monotone Hanani-Tutte, [PT02]). *If G has a y-monotone drawing in which every pair of edges crosses evenly, then G has a planar y-monotone drawing with the same vertex locations and rotation system.*

independent

Two edges are called *independent* if they are not incident to the same vertex.

Theorem 3.4 (Monotone Hanani-Tutte, [FPSS13]). *If G has a y-monotone drawing in which every pair of independent edges crosses evenly, then G has a planar y-monotone drawing with the same vertex locations.*

Based on Theorem 3.4, Fulek et al. provide a quadratic-time algorithm for testing whether a graph with a given placement of its vertices to points on the plane has a planar y-monotone drawing that respects this placement.

In order to test whether a leveled graph (G, ϕ) is level-planar, Fulek et al. construct a graph G' and a placement of $V(G')$ to points on the plane such that (G, ϕ) is level planar if and only of G' has a planar y-monotone drawing that respects the given placement. They add at most $|V(G)|$ vertices and edges to G in order to construct G'. Their construction results in the following theorem.

Theorem 3.5 (Testing Level Planarity, [FPSS13]). *Testing level planarity can be reduced to testing y-monotonicity.*

3.1.1.4 Testing with 2-SAT

Randerath et al.[RSB+01] reduce the Level Planarity Testing Problem to the 2-SAT Problem. However, Fulek et al.[FPSS13] state that there seem to be some gaps in the reduction.

3.1.2 Testing Extended Level Planarity

Hong and Nagamochi [HN09] provide an $\mathcal{O}(|E| + |V|^{p+1} \cdot (2/p)^p)$ algorithm for testing extended h-level planarity of an extended leveled graph (G, ϕ) with p isolated components. If (G, ϕ) has no isolated components, their algorithm takes $\mathcal{O}(|E|+|V|)$ time. They provide a construction that converts extended leveled graphs into leveled graphs with more layers and then test level planarity via the linear time algorithm from Mutzel et al. [JLM98].

3.2 Characterization via Forbidden Patterns

Estrella-Balderrama et al. [EBFK09a] characterize all unlabeled level planar trees by providing two forbidden substructures which prevent unlabeled level planarity, i.e. a tree contains a subtree homeomorphic to one of the structures, if and only if the tree is not unlabeled level planar. Fowler et al. [FK07a] extend this characterization to unlabeled level planar graphs by providing five more forbidden subgraphs. This set $\mathcal{F}$ of seven forbidden subgraphs is shown in Figure 3.3. As an example, a y-ordering λ of G_α such that (G_α, λ) is not level planar is given in Figure 3.4. For bad y-orderings of the other graphs of $\mathcal{F}$, see the paper from Fowler et al.

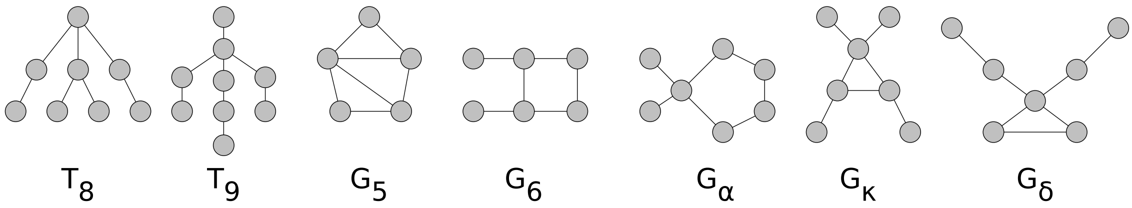


Figure 3.3: The seven forbidden subgraphs $\mathcal{F} = \{T_8, T_9, G_5, G_6, G_\alpha, G_\kappa, G_\delta\}$

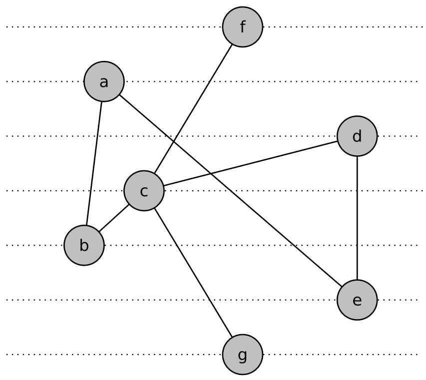


Figure 3.4: A y-ordering that prevents G_α from being unlabeled level planar.

There also have been attempts to characterize level planar graphs by providing minimal level non-planar (MLNP) patterns [HKL04], [FK07b], [EBFK09a]. A leveled graph is *minimal level non-planar* if the removal of any edge makes the resulting leveled graph level planar. However, Estrella-Balderrama et al. [EBFK09b] show that there are infinitely many of these MLNP-patterns for trees.

minimal level non-planar

3.3 Crossing Minimization

If a graph with a given layering is not level planar, one can try to minimize the number of crossings in a straight-line drawing of the graph by reordering the vertices on the same layer. However, crossing-minimization is $\mathcal{NP}$-hard even for 2-layered graphs [GJ83]. If you fix the permutation of vertices on one layer, the problem remains $\mathcal{NP}$-hard [EW94].

Matuszewski et al. [MSM99] provide a heuristic for k-layer crossing minimization that uses a technique called *sifting* to find a good position for a vertex. In the sifting technique, every vertex is visited once. The vertex is moved to a position on its layer such that the current total number of crossings is minimized. Then, its position is fixed and another previously unvisited vertex is visited.

Mutzel [Mut01] suggests an alternative method for crossing minimization, namely k-level planarization. This means that first, a maximal level planar subgraph is drawn and then the previously removed edges are reinserted. A maximal planar subgraph is taken since Eades and Whitesides [EW94] show that finding a maximum level planar subgraph is $\mathcal{NP}$-hard even if there are only two layers and the order of the vertices on one layer is fixed.

The problem of determining whether an undirected graph can be drawn on k layers with at most r crossings is fixed-parameter tractable (but there still remains a huge constant factor), as shown by Dujmović et al [DFK+08]. They provide an $f(k,r) \cdot \mathcal{O}(n)$ algorithm to decide whether a graph can be drawn on k layers such that there are no more than r crossings. Their algorithm is based on computing a path decomposition of the input graph and their observation that a k-level planar graph has pathwidth at most $k-1$.

Bachmaier et al. [BBFH10] show that minimizing the number of crossings in extended level-drawings is $\mathcal{NP}$-hard and provide a heuristic that reduces the number of crossings by up to 30 percent compared to previous crossing minimization heuristics.

3.4 Level Planarity and k-Level Planarity Problems

Heath and Rosenberg [HR89] show that given an undirected n-vertex graph G, it is $\mathcal{NP}$-complete to determine whether there exists a proper layering ϕ of G such that (G, ϕ) is level planar. They show $\mathcal{NP}$-hardness via a reduction from the Planar 3-SAT Problem.

It seems to be still open whether determining the minimum number of layers needed to obtain an (extended) level planar drawing of a planar graph is $\mathcal{NP}$-complete or not. Of course, every undirected n-vertex planar graph G is n-level planar since any planar straight-line drawing of it can be rotated such that the points are mapped to distinct y-coordinates.

Dujmović et al [DFK+08] et al. show that the problem whether an undirected graph is k-level planar is fixed-parameter tractable (but there still remains a huge constant factor). They provide an $f(k) \cdot \mathcal{O}(n)$ algorithm to decide whether a given graph is k-level planar. The algorithm is based on computing a path decomposition of the input graph and their observation that a k-level planar graph has pathwidth at most $k-1$.

Suderman and Matthew [Sud04] give optimal lower and upper bounds for the minimum number of layers needed in an (extended) leveled planar drawing of a tree T with pathwidth h and linear time drawing algorithms for them matching their upper bounds. Their results are listed in Table 3.1.

Type of Layering	Lower Bound	Upper Bound
Layering	h	$3h/2$
Proper Layering	h	$3h-2$
Extended Layering	$h-1$	$3h/2$
Short Extended Layering	h	$2h-1$

Table 3.1: Bounds on the minimum value k for any tree T with pathwidth h such that T is (extended) k-level planar.

Given an (extended) level planar embedding of a graph, there exists a linear-time divide and conquer straight-line drawing algorithm by Eades et al [EFLN06].

However, Lin [Lin92] shows that the area of a grid needed for any planar straight-line drawing of a level planar graph can be exponential. Lin defines a class of level planar graphs H_n of $10n-6$ vertices and $4n-1$ layers such that any straight-line level planar grid-drawing of it needs a grid of at least width $(2n-2)!$. Figure 3.5 shows H_2.

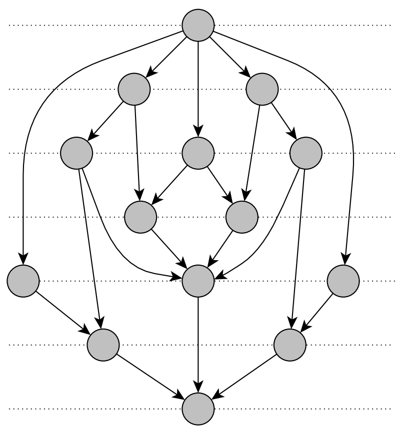


Figure 3.5: H_2

Planar Straight-Line Grid-Drawings of Planar Graphs

Since every extended level planar drawing can be straightened, minimizing k in an extended k-level planar drawing of a graph G is the same as minimizing the height of a planar straight-line grid drawing of G.

Mondal et al. [MA11] provide a linear-time algorithm for computing the minimum height of a straight-line planar grid-drawing of a tree. A linear-time algorithm for computing the minimum height of a planar straight-line grid-drawing of a planar triangulated graph with treewidth 3 is given by Mondal et al. [MNA11]. Biedl [Bie13] gives a linear-time algorithm that computes a straight-line planar drawing of a 2-connected outerplanar graph such that the height of the drawing is a 4-approximation of the minimum height needed.

Schnyder [Sch90] shows that every n-vertex planar graph has a planar straight-line drawing on the $2n-4$ by $n-2$ grid which can be computed in linear time. Chrobak and Nakano [CN95] show that for a triangulated n-vertex planar graph with $n \geq 3$, both dimensions of any planar straight-line grid-drawing of it are at least of size $\lceil 2(n-1)/3 \rceil$. They show that this bound is tight by providing a polynomial-time algorithm that produces a planar straight-line grid-drawing on the $\lceil 2(n-1)/3 \rceil \times 4\lceil 2(n-1)/3 \rceil - 1$ grid.

Minimum-Height Layerings with Additional Restrictions

Some applications define additional restrictions to a layering. Bastert and Matuszewski [BM99] describe a linear-time algorithm that maps the vertices of a directed acyclic graph $G = (V, E)$ to a minimum number of layers via a layering ϕ such that $\phi(u) < \phi(v)$ for all $(u, v) \in E$, resulting in a layering with that property of minimum height.

1. A *sink* is a vertex with no outgoing edges. Each sink $s \in V$ is mapped to $\phi(s) = 1$. *sink*
2. Each vertex v which is not a sink is recursively placed by

$$\phi(v) = \max\{i \mid w \in N^+(v) \text{ and } \phi(w) = i\} + 1$$

where $N^+(v) = \{w \mid (v, w) \in E\}$.

The height of the resulting layering ϕ is equal to the length of the longest path in the input graph. The leveled graph (G, ϕ) is not required to be level planar. Moreover, they mention that it is $\mathcal{NP}$-complete to minimize simultaneously width (maximum number of vertices on a layer) and height (number of layers) of a layering with the property that $\phi(u) < \phi(v)$ for all $(u, v) \in E$.

3.5 The Sugiyama Method

The most commonly used framework to construct a layered drawing of a directed graph is given by Sugiyama et al [STT81]. It mainly consists of four steps:

1. **Cycle Removal** Reverse edges in order to obtain an acyclic graph.
2. **Layer Assignment** Assign the vertices to layers, insert dummy vertices in order to obtain a proper layering.
3. **Crossing Minimization** Minimize crossings between adjacent layers.
4. **Coordinate Assignment and Drawing** Assign coordinates to the vertices and construct a straight-line drawing.

A visualization of the Sugiyama Layout is shown in Figure 3.6.

3.6 Further Related Work

Work related to Moving Vertices

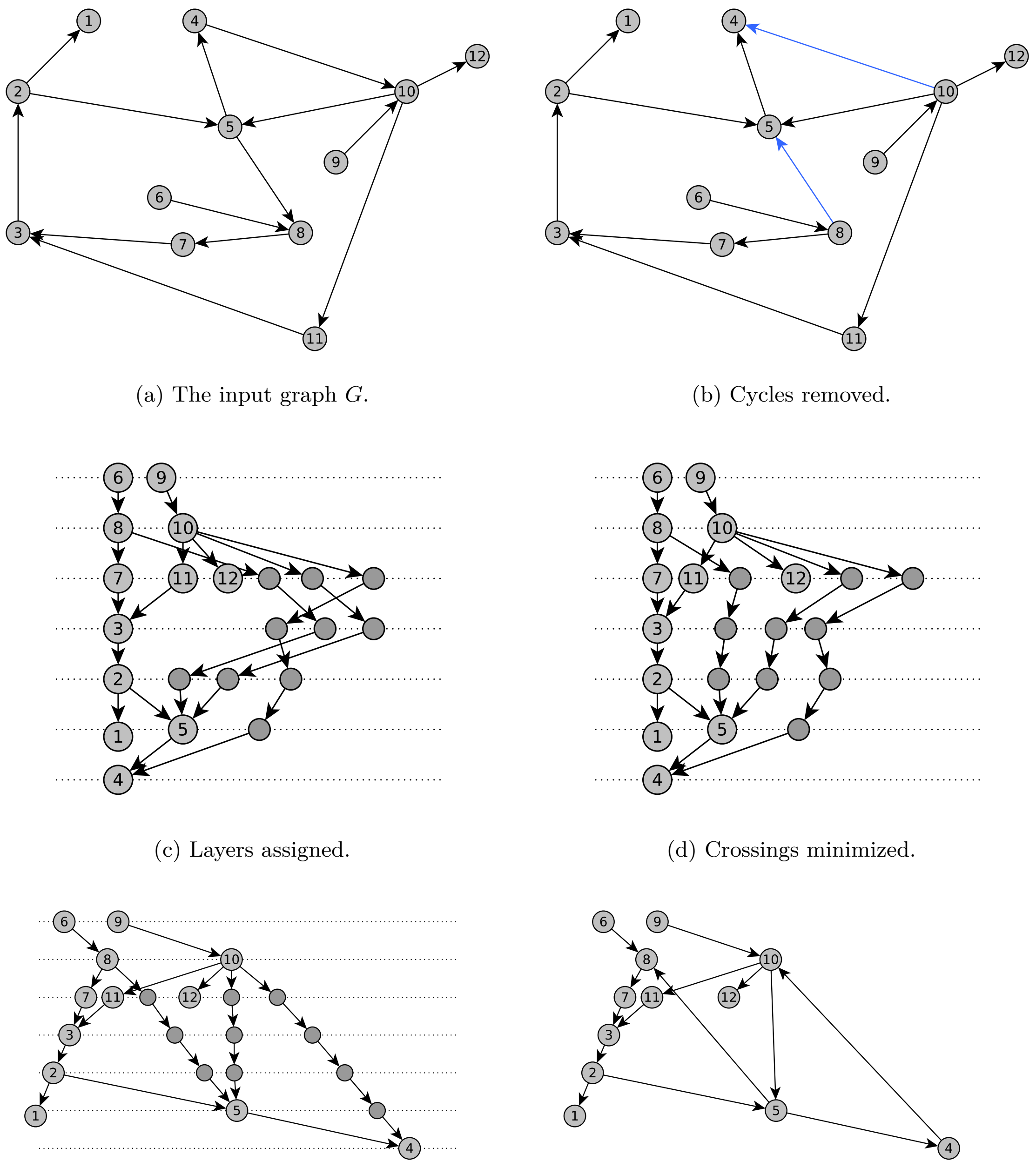
Goaoc et al. [GKO$^+$09] show that untangling a straight-line drawing of a planar graph is $\mathcal{NP}$-hard and the minimum number of vertices needed to be moved is impossible to approximate if $\mathcal{P} \neq \mathcal{NP}$. They also provide a quadratic-time algorithm for untangling a nonplanar straight-line drawing of an n-vertex planar graph that fixes at least $\sqrt{((\log n) - 1)/\log\log n}$ vertices (i.e., these vertices are not moved). If the graph is outerplanar, their algorithm fixes $\sqrt{n/2}$ vertices. They show that for outerplanar graphs, their algorithm is asymptotically worst-case optimal. Bose et al. [BDH$^+$07] show that in any nonplanar straight-line drawing of a planar graph, $\sqrt[4]{(n+1)/2}$ vertices can always be fixed (i.e., not moved).

Patrignani [Pat06] shows that given a planar straight-line drawing of a subgraph of a planar graph G, it is $\mathcal{NP}$-hard to extend it to a planar straight-line drawing of G. However, if the aforementioned subgraph is a cycle in G whose vertices are mapped to a convex polygon, Mchedlidze et al.[MNR13] provide a linear-time testing and drawing algorithm.

Work related to Swapping Vertices

Bulteau et al. [BFR12] show that it is $\mathcal{NP}$-hard to determine the minimum number of transpositions (i.e., swaps of consecutive elements) needed to transform a given permutation into identity.

Cicirello and Cernera [CC13] show that given two permutations, one can determine in polynomial time how many swaps or reinsertions are needed to transform one permutation into the other. The minimum number of moves needed can be solved via computing the edit distance with costs 0.5 for deletion, 0.5 for insertion and ∞ for replacement. They provide a polynomial time algorithm for computing the minimum number of swaps needed, based on the observation from Oliver et al. [OSH87] that each crossover cycle of length k leads to $k - 1$ needed swaps.



(a) The input graph G.

(b) Cycles removed.

(c) Layers assigned.

(d) Crossings minimized.

(e) Edges straightened.

(f) The final drawing.

Figure 3.6: The steps of the Sugiyama layout.

4. New Considerations

In this chapter we analyze the newly defined problems from Section 2.3.2 and provide heuristics for them.

Recall that by *reassigning* the layer assignment of a vertex v, we mean the change of the value $\phi(v)$. As a result, v is then mapped to another already existing layer. *reassigning*

(EXTENDED) LAYER REASSIGN PROBLEM
INPUT: An (extended) planar leveled graph (G, ϕ), an integer r
QUESTION: Can we get an (extended) level planar graph from (G, ϕ) by changing the layer assignments of at most r vertices in ϕ?

Recall that by *swapping* the layer assignments of two vertices u and w, we understand that the vertex u gets mapped to $\phi(v)$ and the vertex v gets mapped to the former $\phi(u)$. *swapping*

(EXTENDED) LAYER SWAP PROBLEM
GIVEN: An (extended) planar leveled graph (G, ϕ), an integer s
QUESTION: Can we obtain an (extended) level planar graph from (G, ϕ) by performing at most s swaps in ϕ?

Recall that by *moving* a vertex v, we understand that either the layer assignment of v is changed or a new layer L is introduced and v is then assigned to L. The newly introduced layer is then inserted above, between or below the already existing layers. *moving*

(EXTENDED) VERTEX MOVE PROBLEM
INPUT: An (extended) planar leveled graph (G, ϕ), an integer m
QUESTION: Can we get an (extended) level planar graph from (G, ϕ) by moving at most m vertices in ϕ?

4.1 Bounds and Complexity

In this section, we show that the Extended k-Level Planarity Problem can be reduced both to the Extended Layer Reassign Problem, and to the Extended Layer Swap Problem. Moreover, we show that in order to solve the (Extended) Vertex Move Problem, one cannot fix the vertices of a maximum induced (extended) level planar subgraph of an (extended) leveled graph.

We deduce bounds for the (Extended) Vertex Move Problem from related problems for planar graphs.

4.1.1 Changing and Swapping Layer Assignments

Lemma 4.1. *If the* Extended k-Level Planarity Problem *is $\mathcal{NP}$-complete, then the* Extended Layer Reassign Problem *is also $\mathcal{NP}$-complete.*

Proof. The Extended Layer Reassign Problem is in $\mathcal{NP}$ since for a given extended k-layered graph (G, ϕ), an integer r and a set R of layer reassignments on ϕ, one can check in polynomial time whether $|R| \leq r$ and whether (G, ϕ') is extended k-level planar. The extended layering ϕ' results from performing the layer reassignments given by R on ϕ.

Extended k-Level Planarity Problem $\propto$ Extended Layer Reassign Problem: Let $\mathcal{I} = (G, k)$ be an instance of the Extended k-Level Planarity Problem. We construct an instance $\mathcal{J} = ((G', \phi), r)$ of the Extended Layer Reassign Problem as follows.

- $G' = (V_{G'}, E_{G'}) := G$
- The integer r is set to n, i.e. the number of vertices in G.
- The extended layering ϕ is constructed as follows. We order the vertices of G' arbitrarily such that $V_{G'} = \{v_1, \ldots, v_n\}$. We define $\phi(v_i) = i$ for $i = 1, \ldots, k$ and $\phi(v_j) = 1$ for $j = k + 1, \ldots, n$.

We show that an extended layering ψ of height k of G exists such that (G, ψ) is extended k-level planar if and only if we can make (G', ϕ) extended k-level planar by reassigning at most n vertices in ϕ.

$\Rightarrow$: Assume that there is an extended layering ψ of height k of G such that (G, ψ) is extended k-level planar. The layering ϕ can trivially be transformed into the layering ψ by reassigning at most n vertices.

$\Leftarrow$: Assume that we can make (G', ϕ) extended k-level planar by reassigning at most n vertices in ϕ. Let ϕ_2 be the extended layering obtained by performing those reassignments. Obviously, (G, ϕ_2) is extended k-level planar.

Thus, if the Extended k-Level Planarity Problem is $\mathcal{NP}$-complete, then the Extended Layer Reassign Problem is also $\mathcal{NP}$-complete. $\square$

Lemma 4.2. *If the* Extended k-Level Planarity Problem *is $\mathcal{NP}$-complete, then the* Extended Layer Swap Problem *is also $\mathcal{NP}$-complete.*

Proof. The Extended Layer Swap Problem is in $\mathcal{NP}$ since for a given extended k-layered graph (G, ϕ) and a sequence S of swaps on ϕ, one can check in polynomial time whether $|S| \leq s$ and whether (G, ϕ') is extended k-level planar. The extended layering ϕ' results from performing the layer swaps given by S on ϕ.

Extended k-Level Planarity Problem $\propto$ Extended Layer Swap Problem: Let $\mathcal{I} = (G, k)$ be an instance of the Extended k-Level Planarity Problem. We construct an instance $\mathcal{J} = ((G', \phi), s)$ of the Extended Layer Swap Problem as follows.

- The integer s is set to n, i.e. the number of vertices in G.
- Let φ be the extended layering obtained by the following construction. We order the vertices of G' arbitrarily such that $V_{G'} = \{v_1, \ldots, v_n\}$. We define $\varphi(v_i) = i$ for $i = 1, \ldots, k$ and $\varphi(v_j) = 1$ for $j = k+1, \ldots, n$.
- Add n new isolated vertices (i.e., vertices with degree zero) to each layer in φ, $k \cdot n$ vertices in total. These extra vertices allow that each vertex can "swap" to any layer. Since $k \leq n$, this adds $\mathcal{O}(n^2)$ new vertices. Let $G' = (V_{G'}, E_{G'})$ be G combined with the newly added vertices of degree zero. Let ϕ be φ with the new vertex mappings added.

We show that an extended layering ψ of height k of G exists such that (G, ψ) is extended k-level planar if and only if we can make (G', ϕ) extended k-level planar by performing at most n swaps in ϕ.

$\Rightarrow$: Assume that there is an extended layering ψ of height k of G such that (G, ψ) is extended k-level planar. By swapping at most n vertices in ϕ, we can reach that that $\phi|_{V_G} = \psi$. This can always be done since the newly added vertices on each layer allow each vertex to be "swapped" to any of the k layers. Since all vertices in $V_{G'} \setminus V_G$ have degree zero, it is irrelevant for level planarity on which layer they are placed. Thus, (G', ϕ) can be made level planar by swapping at most n vertices in ϕ.

$\Leftarrow$: Assume that we can make (G', ϕ) extended k-level planar by performing at most n swaps in ϕ. Let ϕ_2 be the extended layering obtained by performing those swaps. Obviously, (G, ϕ_2) is extended k-level planar.

Thus, if the Extended k-Level Planarity Problem is $\mathcal{NP}$-complete, then the Extended Layer Swap Problem is also $\mathcal{NP}$-complete. □

In the aforementioned reductions, we need to find an extended layering of height k in polynomial time. We emphasize that we do not require extended level planarity here. The minimum height of an extended layering is always one since each vertex can be mapped to the same layer.

In order to transfer our reductions from above to the Layer Reassign Problem and the Layer Swap Problem via a reduction from the k-Level Planarity Problem, it is needed to construct a layering of height k of a graph G in polynomial time if it exists. A natural approach is to compute a minimum-height layering of G and check whether its height is less or equal k. If it is greater than k, it is clear that there is no layering ϕ of G such that (G, ϕ) is k-level planar. Otherwise, new layers can be added to a minimum-height layering of G and vertices can be reassigned to these layers until we obtain a layering of height k of G. Then, we can apply the same ideas for the reduction as before.

This approach does not help as we show in the following lemma.

Lemma 4.3. *Let $G = (V_G, E_G)$ be a graph. It is $\mathcal{NP}$-complete to find a minimum-height layering of G.*

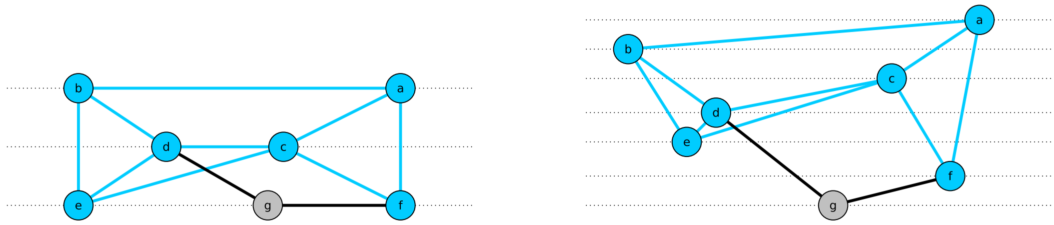
Proof idea. Let ϕ be a minimum-height layering of G. Let h be the height of ϕ. Since ϕ is a layering, it holds that $\phi(u) \neq \phi(v)$ for all adjacent vertices $u, v \in V_G$. This implies that the vertices of G can be colored with h colors such that adjacent vertices have different colors and h is the minimum number of colors needed. However, it is $\mathcal{NP}$-complete to determine this number (known as the *chromatic number* of G) [GJS74]. □

4.1.2 Moving Vertices

An approach that can be tried in order to analyze the complexity of the (Extended) Vertex Move Problem is to reduce of the Maximum Induced (Extended) Level Planar Subgraph Problem to the (Extended) Vertex Move Problem. While it is unclear whether the Maximum Induced (Extended) Level Planar Subgraph Problem is $\mathcal{NP}$-complete or not, a reduction from it would still give some classification.

A natural approach to do this reduction is to state that the vertices of a maximum induced (extended) level planar subgraph of an (extended) leveled graph need not to be moved in order to make the (extended) leveled graph (extended) level planar. This attempt does not work, as we show in the following observation. This provides some useful insight about the (Extended) Vertex Move Problem.

Observation 4.4. *Let (G, ϕ) be a planar (extended) leveled graph which is not (extended) level planar. Let H be a maximum induced (extended) level planar subgraph of (G, ϕ). We cannot always obtain an (extended) k-level planar drawing of G by only moving vertices of G in ϕ which are not contained in H. An example is given in Figure 4.1.*



(a) An extended layering of G.

(b) An y-ordering of G.

Figure 4.1: In both cases, an (extended) level planar drawing of G cannot be obtained by moving only the vertex g. The graph G has been constructed out of one of the forbidden patterns for partially embedded graphs from Jelínek et al[JKR13]. We have used the level planarity testing algorithm from Healy and Kuusik [HK99b] in a brute-force approach in order to test whether the given leveled graph can be made level planar by moving only the vertex g or not. Our implementation did not find a way to move only g in order to make the given leveled graph level planar.

An upper bound for the (Extended) Vertex Move problem

It is known that for untangling a straight-line drawing of an n-vertex planar graph, no more than $\sqrt[4]{(n+1)/2}$ need to be moved [BDH$^+$07]. The following corollary is a direct implication from that result.

Corollary 4.5. *Given an n-vertex planar graph G and an (extended) layering ϕ of G, no more than $\sqrt[4]{(n+1)/2}$ vertices need to be moved in ϕ in order to make (G, ϕ) (extended) level planar.*

4.2 The Rotation Heuristic

In this section, we give a heuristic that can be applied to the (Extended) Layer Change problem, the (Extended) Layer Swap problem and the (Extended) Vertex Move problem.

Our input consists of an (extended) leveled graph (G, ϕ) such that (G, ϕ) is not (extended) level planar but G is planar. We perform the following steps, which are explained in detail in the following.

1. Compute a planar straight-line drawing δ of G, for example with the linear-time algorithm by Schnyder [Sch89].
2. Compute the set of all possible circular orderings of the points given by δ. Each circular ordering induces a y-ordering ψ of the graph. This is explained in Section 4.2.2.
3. For each induced y-ordering ψ, compute the number of operations needed to transform ϕ into ψ, as explained in Section 4.2.1.
4. Return the minimum of the distances computed in steps 2 and 3.

If a solution is found via the Rotation Heuristic, it is guaranteed that the resulting drawing is (extended) level planar.

4.2.1 Compute the Distance

In this subsection, we consider the distance of two extended layerings since this provides a more general result.

The distance between two extended layerings can be computed as follows, depending on the problem which we want to find an heuristical solution for. Let ϕ and ψ be two extended layerings of $G = (V_G, E_G)$ (the extended layering ϕ is part of the input, the extended layering ψ is induced by the current drawing).

Extended Layer Reassignment Problem

In order to transform ϕ into ψ by reassigning vertices to other existing layers, we need to change the values $\phi(v)$ for all $v \in V_G$ where $\phi(v) \neq \psi(v)$. However, since we do not allow inserting new layers, we have to require that the height of ψ is less than or equal to the height of ϕ. We do not have to require that ϕ and ψ have exactly the same height since reassigning vertices to other existing layers is allowed to produce empty layers.

Algorithm 4.1 computes the minimum number of reassignments needed to transform the extended layering ϕ into the extended layering ψ.

Algorithm 4.1: MinReassign finds the minimum number of layer reassignments.

Input: Two extended layerings ϕ and ψ of a graph $G = (V, E)$
Output: The minimum number r of reassignments needed to transform ϕ into ψ

```
r ← 0
if height(ψ) > height(φ) then
    return false
else
    foreach v ∈ V do
        if φ(v) ≠ ψ(v) then
            r ← r + 1
    return r
```

Extended Layer Swap Problem

Swapping the layer assignments of two vertices does not change the total number of vertices on each layer (if we swap two vertices u and v, we swap u “away” from its layer but the gap is then filled with v).

We observe that in order to be able to transform the ϕ into ψ by swapping vertices, the following two conditions are necessary.

1. Since swapping vertices in ϕ can neither reduce the number of layers nor increase the number of layers, the extended layerings ϕ and ψ must have the same height, say h.
2. Since the number of vertices on each layer stays invariant during swap operations, it must hold that $|\phi^{-1}(i)| = |\psi^{-1}(i)|$ for $i = 1, \ldots, h$, i.e., the number of vertices on layer i is the same for both extended layerings.

We make the following observations in order to obtain the minimum number of swaps needed to transform ϕ into ψ.

correct
incorrect

We say that a vertex v is *correct* if $\phi(v) = \psi(v)$, otherwise it is *incorrect*.

Observation 4.6. *We do not need to swap vertices that are on the same layer.*

Observation 4.7. *Each needed swap increases the total number of correct vertices.*

Observation 4.8. *We do not need to swap a correct vertex with another vertex. In other words, each needed swap swaps two incorrect vertices.*

Observation 4.9. *Let u, v and w be three pairwise distinct incorrect vertices. Let $\phi(v) = \psi(u) = \phi(w)$ (i.e., both v and w are on the desired layer of u). Let $\psi(v) = \phi(u)$ and $\psi(w) \neq \phi(u)$. Since swapping the vertices u and v makes both of them correct and swapping the vertices u and w makes only one of them correct, it is better to swap u and v.*

Based on the aforementioned observations, Algorithm 4.2 computes the minimum number of swaps needed to transform ϕ into ψ. If we restrict us to y-orderings, the minimum number of swaps needed to transform ϕ into ψ can also be computed by the algorithm of Cicirello and Cernera [CC13] in polynomial time.

Algorithm 4.2: MinSwaps finds the minimum number of swaps.

Input: Two extended layerings ϕ and ψ of a graph $G = (V, E)$
Output: The minimum number s of swaps needed to transform ϕ into ψ

$s \leftarrow 0$
if *height*(ψ) $\neq$ *height*(ϕ) **then**
    return *false*
else if $\exists i\colon |\psi^{-1}(i)| \neq |\phi^{-1}(i)|$ **then**
    return *false*
else
    $F \leftarrow \{v \in V \mid \phi(v) = \psi(v)\}$ `// vertices which already are on their desired layer`
    foreach $v \in V \setminus F$ **do**
        candidates $\leftarrow \{u \in (V \setminus F) \mid u \in \phi^{-1}(\psi(v))\}$ `// vertices which are on the layer that` v `wants to be swapped to`
        if *candidates* $= \emptyset$ **then**
            return *false*
        else if $\exists u \in$ *candidates* : $\psi(u) = \phi(v)$ **then**
            swap v and u in ϕ
            $F \leftarrow F \cup \{v\} \cup \{u\}$
            $s \leftarrow s + 1$
        else
            swap v and some vertex $u \in$ candidates in ϕ
            $F \leftarrow F \cup \{v\}$
            $s \leftarrow s + 1$
    return s

Extended Vertex Move Problem

We first show that computing the minimum number of moves needed to transform ϕ into ψ can be reduced to computing the maximum common subsequence (which is defined below), then we show that it can be computed in polynomial time.

For two vertices $u, v \in V$ and an extended layering ℓ of $G = (V, E)$, we define

$$\text{relation}(u, v, \ell) = \begin{cases} -1, & \text{if } \ell(u) < \ell(v) \\ 0, & \text{if } \ell(u) = \ell(v) \\ 1, & \text{if } \ell(u) > \ell(v) \end{cases}$$

common subsequence *maximum common subsequence*

A *common subsequence* of the extended layerings ϕ and ψ is a set $S \subseteq V$ such that for all vertices $u, v \in S$ it holds that $\text{relation}(u, v, \phi) = \text{relation}(u, v, \psi)$. A *maximum common subsequence* of the extended layerings ϕ and ψ is a common subsequence $S_{\max}$ of ϕ and ψ with maximum size, i.e. there is no common subsequence S' of ϕ and ψ with $|S'| > |S_{\max}|$.

Lemma 4.10. *Let (G, ϕ) be an extended leveled graph. Let ψ be an extended layering of G. Let S_{max} be a maximum common subsequence of ϕ and ψ. Let m_{min} be the minimum number of moves needed to transform ϕ into ψ. It holds that $m_{min} = |V| - |S_{max}|$.*

Proof idea. A maximum common subsequence $S_{\max}$ of ϕ and ψ contains the maximum number of vertices that can be fixed in ϕ since they appear already in the correct order, i.e. they do not need to be moved. Thus, $|V| - |S_{\max}| = m_{\min}$. □

conflict graph

Given two extended layerings ϕ and ψ of a graph $G = (V, E)$ and a vertex $v \in V$, we define $\text{Forbidden}(v) := \{w \in V \mid \text{relation}(v, w, \phi) \neq \text{relation}(v, w, \psi)\}$. We can then define the *conflict graph* $K = (V_K, E_K)$ of ϕ and ψ as $V_K := V$ and $\{u, v\} \in E_K$ if and only if $v \in \text{Forbidden}(u)$. An example of a conflict graph is given in Figure 4.2.

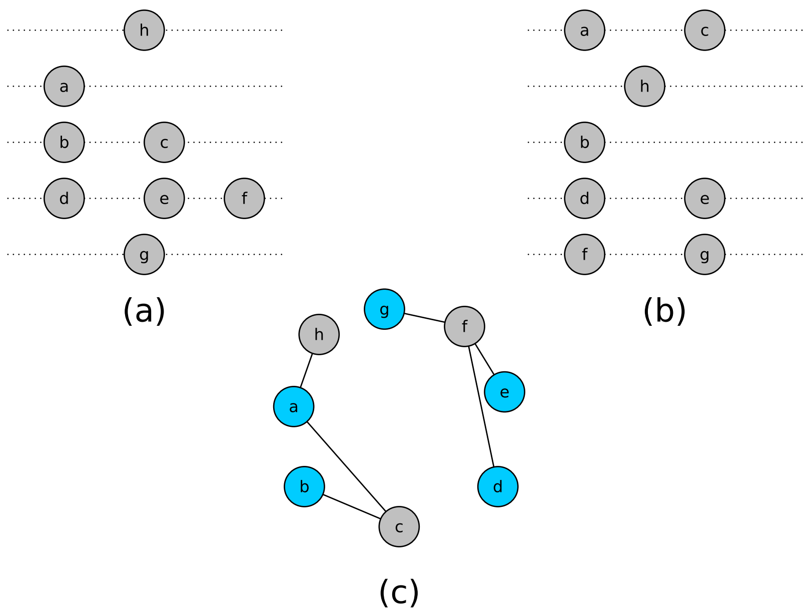


Figure 4.2: (a) A layering ϕ of G. (b) A layering ψ of G. (c) The conflict graph K of ϕ and ψ. A maximum common subsequence of ϕ and ψ is marked in blue. Notice that it is also a maximum independent set of K. In this example, at least 3 vertices need to be moved.

We observe that a maximum common subsequence of ϕ and ψ is a maximum independent set in the conflict graph of ϕ an ψ.

While the Maximum Independent Set Problem is $\mathcal{NP}$-complete for general graphs [Coo71], there are polynomial-time algorithms for it if the graph belongs to a special class of graphs, the so-called *perfect graphs* [GLS88]. We show that the conflict graph K of ϕ and ψ is a comparability graph, which is a member of the class of perfect graphs [Dil90].

Lemma 4.11. *Let ϕ and ψ be two extended layerings of a graph $G = (V_G, E_G)$. The conflict graph $K = (V_K, E_K)$ of ϕ and ψ is a comparability graph.*

Proof. We define the strong partially ordered set $(S, <_S)$ as follows.

- $S := V_G$
- $u <_S v$ if and only if $\text{relation}(u, v, \phi) < \text{relation}(u, v, \psi)$

In order to show that $(S, <_S)$ is a strong partially ordered set, we have to show that "$<_S$" is irreflexive, transitive and asymmetric.

Irreflexivity We have to show that not $x <_S x$ for all $x \in S$.

Let $x \in S$. Since $\text{relation}(x, x, \phi) = 0$ and $\text{relation}(x, x, \psi) = 0$, it holds that not $x <_S x$.

Transitivity We have to show that if $x <_S y$ and $y <_S z$ then $x <_S z$ for all $x, y, z \in S$.

Let $x, y, z \in S$ and let $x <_S y$ and $y <_S z$. It holds that

$$x <_S y \Leftrightarrow \text{relation}(x, y, \phi) < \text{relation}(x, y, \psi)$$

$$\Leftrightarrow \Big(\phi(x) < \phi(y) \text{ and } \psi(x) \geq \psi(y)\Big) \text{ or } \Big(\phi(x) = \phi(y) \text{ and } \psi(x) > \psi(y)\Big).$$

Similarly, it holds that

$$y <_S z \Leftrightarrow \Big(\phi(y) < \phi(z) \text{ and } \psi(y) \geq \psi(z)\Big) \text{ or } \Big(\phi(y) = \phi(z) \text{ and } \psi(y) > \psi(z)\Big).$$

This gives us four cases.

1. $\phi(x) < \phi(y) < \phi(z)$ and $\psi(x) \geq \psi(y) \geq \psi(z)$
2. $\phi(x) < \phi(y) = \phi(z)$ and $\psi(x) \geq \psi(y) > \psi(z)$
3. $\phi(x) = \phi(y) < \phi(z)$ and $\psi(x) > \psi(y) \geq \psi(z)$
4. $\phi(x) = \phi(y) = \phi(z)$ and $\psi(x) > \psi(y) > \psi(z)$

In each of the four cases, it holds that $\text{relation}(x, z, \phi) < \text{relation}(x, z, \psi)$. Hence, it holds that $x <_S z$.

Asymmetry We have to show that if $x <_S y$ then not $y <_S x$ for all $x, y \in S$.

Let $x, y \in S$. Assume that $x <_S y$ and $y <_S x$. By the transitivity of "$<_S$", it follows that $x <_S x$. This cannot be since "$<_S$" is irreflexive. Thus, "$<_S$" is asymmetric.

Thus, $(S, <_S)$ is a strong partially ordered set.

We show that the comparability graph $C = (V_C, E_C)$ of $(S, <_S)$ and K are isomorphic. Since both $V_C = V_G$ and $V_K = V_G$, it suffices to show that $\{u, v\} \in E_C \Leftrightarrow \{u, v\} \in E_K$.

$$\begin{gathered}\{u, v\} \in E_C \\ \Leftrightarrow u <_S v \text{ or } v <_S u \\ \Leftrightarrow \text{relation}(u, v, \phi) < \text{relation}(u, v, \psi) \text{ or relation}(u, v, \phi) > \text{relation}(u, v, \psi) \\ \Leftrightarrow \text{relation}(u, v, \phi) \neq \text{relation}(u, v, \psi) \\ \Leftrightarrow u \in \text{Forbidden}(v) \text{ or } v \in \text{Forbidden}(u) \\ \Leftrightarrow \{u, v\} \in E_K.\end{gathered}$$

Thus, C and K are isomorphic. □

Maximum Common Subsequence for y-Orderings

If ϕ and ψ are y-orderings, the maximum common subsequence of ϕ and ψ can be reduced to computing the longest common subsequence of two sequences, both of size $|V|$, as we show in the next paragraph. This can be done via a dynamic programming algorithm in $\mathcal{O}(|V|^2)$ time [Hir75].

Observation 4.12. *Let ϕ and ψ the two y-orderings of a graph $G = (\{v_1, \ldots, v_n\}, E)$. Let σ_1 and σ_2 be two sequences of size n such that $\sigma_1(i) = \phi(v_i)$ and $\sigma_2(i) = \psi(v_i)$ for $i = 1, \ldots, n$. The maximum common subsequence of ϕ and ψ is the same as the longest common subsequence of σ_1 and σ_2.*

Our results lead to the following theorem.

Theorem 4.13. *Let ϕ and ψ be two extended layerings of a graph. The minimum number of moves needed to transform ϕ into ψ can be computed in polynomial time.*

4.2.2 Rotate the Drawing

circular ordering A *circular ordering* of a point set P is a vertical ordering of the points in P that can be obtained by rotating the point set around its center such that no two points have the same y-coordinate. (It is a well-known fact that by rotating the point set slightly around its center, we can always ensure this.)

As shown by Goodman and Pollak [Goo80], there are $\Theta(n^2)$ possible circular orderings of a point set. As shown by Biery and Schmidt [BS96], the set of circular orderings of a point set of size n can be computed in $\mathcal{O}(n^4)$ time.

Let δ be the planar straight-line drawing of G which we obtained in Step 1. The vertex locations of G in δ induce a point set P. Each circular ordering of P represents a y-ordering of the vertices of G.

In this step, we use the quartic-time algorithm from Biery and Schmidt to compute all y-orderings of G induced by rotations of δ.

Further Improvements

It remains open whether we can find parts of the drawing which we can rotate independently without destroying planarity in order to obtain more (extended) layerings. Moreover, since our current approach does only produce y-orderings, it remains further work to investigate ways to "compress" them into (extended) layerings of smaller height.

4.2.3 Time Complexity

Let $(G = (V_G, E_G), \phi)$ be the (extended) leveled graph provided by the input. Let $n = |V_G|$. We compute all possible circular orderings of the points given by the drawing in $\mathcal{O}(n^4)$ time. There are $\mathcal{O}(n^2)$ many possible circular orderings of a point set. Each circular ordering induces a y-ordering of the graph. For each induced y-ordering δ, we compute the distance of ϕ to δ.

We need $\mathcal{O}(n)$ time to find the minimum number of reassignments needed to transform an extended layering ϕ into an extended layering ψ. We need $\mathcal{O}(n^2)$ time to find the minimum number of swaps needed to transform an extended layering ϕ into an extended layering ψ. If ϕ and ψ are y-orderings, we need $\mathcal{O}(n^2)$ time to find the minimum number of moves needed to transform ϕ into ψ. Otherwise, this can also be done in polynomial time.

All in all, the Rotation Heuristic takes $\mathcal{O}(n^4)$ time for the (Extended) Layer Reassign Problem and the (Extended) Layer Swap Problem. If ϕ is an y-ordering, the Rotation Heuristic for the (Extended) Vertex Move Problem takes also $\mathcal{O}(n^4)$ time. Depending on the algorithm chosen for computing a maximum independent set in a comparability graph, the Rotation Heuristic for the (Extended) Vertex Move Problem for arbitrary (extended) layerings still takes polynomial time.

4.3 More Ideas for Heuristics

In this section, we shortly discuss further ideas for heuristics.

Greedy Heuristics After starting with some intial (extended) leveled embedding and applying a heuristic for the (Extended) Crossing Minimization Problem, we can greedily try to reassign/ swap/ move vertices between layers as long as we can reduce the total number of crossings by one single reassignment/ move/ swap.

Untangling Drawings Since testing whether a graph is planar can be done in linear time [HT74] and planarity is a necessary condition for level planarity, we can restrict us to planar graphs. We can combine a heuristic for the Untangling a Planar Graph Problem, for example the quadratic-time heuristic from Goaoc et al. [GKO+09], with a heuristic for the (Extended) Crossing Minimization Problem in order to obtain a heuristic for the (Extended) Vertex Move Problem as follows.

Given an (extended) leveled graph (G, ϕ), we choose some initial (extended) leveled embedding ϵ of G with respect to ϕ. We reorder the vertices within each layer according to a heuristic for the (Extended) Crossing Minimization Problem, resulting in an (extended) leveled embedding ϵ'. We construct a straight-line (extended) leveled drawing of G out of ϵ'. Then, we apply a heuristic for the Untangling a Planar Graph Problem and return the total number of vertices moved by this heuristic.

5. Conclusion

We have given an extensive overview on current research concerning level planarity, introduced and analyzed some new optimization problems and provided a heuristic for them. Table 5.1 sums up the complexity results of all problems mentioned in this thesis.

We have shown that if the EXTENDED k-LEVEL PLANARITY PROBLEM is $\mathcal{NP}$-complete, then it is also $\mathcal{NP}$-complete to determine the minimum number of layer reassignments or swaps needed in order to make an extended leveled graph extended level planar.

We have given a polynomial-time heuristic that we call the Rotation Heuristic. This heuristic can be applied to the new problems we defined in this thesis, namely the (EXTENDED) LAYER REASSIGN PROBLEM, the (EXTENDED) LAYER SWAP PROBLEM and the (EXTENDED) VERTEX MOVE PROBLEM.

We have shown that finding the minimum number of vertices needed to move in order to transform one extended layering into another can be reduced to finding a maximum independent set in a comparability graph which is in $\mathcal{P}$. If we restrict us to y-orderings (i.e., each vertex is mapped to a different layer), we have observed that the problem can be reduced to finding a longest common subsequence of the permutations that are given by the y-orderings.

5.1 Open Problems and Further Work

To our knowledge, it still remains open whether the (EXTENDED) k-LEVEL PLANARITY problem is $\mathcal{NP}$-complete or not. Since the $\mathcal{NP}$-completeness of most of the related problems (proper level planarity, crossing minimization and untangling straight-line nonplanar drawings of planar graphs) is shown via a reduction from the PLANAR 3-SAT PROBLEM, the PLANAR 3-SAT PROBLEM seems to be a promising candidate for the reduction in an $\mathcal{NP}$-hardness proof.

It also remains open whether the (EXTENDED) LAYER CHANGE problem, the (EXTENDED) LAYER SWAP problem and the (EXTENDED) VERTEX MOVE problem are $\mathcal{NP}$-complete in the case that we restrict us to y-orderings instead of general (extended) layerings. It remains further work to analyze the complexity of the open problems mentioned above. Also, the quality of the rotation heuristic can still be improved, as explained in Section 4.2.2.

Problem	Time complexity	Shown by
Planar 3-SAT Problem	A	Lichtenstein [Lic82]
Maximum Independent Set Problem	A	Cook [Coo71]
Proper Level Planarity Problem	A	Heath and Rosenberg [HR89]
k-Level Planarity Problem	C	
Extended k-Level Planarity Problem	C	
Level Planarity Testing Problem	B	Juenger, Leipert and Mutzel [JLM98]
Extended Level Planarity Testing Problem	E	Hong and Nagamochi [HN09]
Crossing Minimization Problem	A	Garey and Johnson [GJ83]
Extended Crossing Minimization Problem	A	Bachmeier et al. [BBFH10]
Maximum Level Planar Subgraph Problem	A	Eades and Whitesides [EW94]
Maximum Extended Level Planar Subgraph Problem	A	Trivial result
Maximum Induced Level Planar Subgraph Problem	C	
Maximum Induced Extended Level Planar Subgraph Problem	C	
Untangling Planar Graph Problem	A	Goaoc et al. [GKO+09]
Partial Drawing Extensibility Problem	A	Patrignani [Pat06]
Layer Reassign Problem	C	
Extended Layer Reassign Problem	D	This thesis
Layer Swap Problem	C	
Extended Layer Swap Problem	D	This thesis
Vertex Move Problem	C	
Extended Vertex Move Problem	C	

Table 5.1: Complexity results. A : $\mathcal{NP}$-complete, B : $\mathcal{P}$, C : unknown, D : unknown but $\mathcal{NP}$-complete if the Extended k-Level Planarity Problem is $\mathcal{NP}$-complete, E : unknown but in $\mathcal{P}$ if the number of isolated components is bounded by a constant.

Bibliography

[AL91] Stefan Arnborg and Jens Lagergren. Easy problems for tree-decomposable graphs, 1991.

[BBFH10] Christian Bachmaier, Hedi Buchner, Michael Forster, and Seok-Hee Hong. Crossing minimization in extended level drawings of graphs. *Discrete Applied Mathematics*, 158(3):159–179, 2010.

[BDH$^+$07] Prosenjit Bose, Vida Dujmovic, Ferran Hurtado, Stefan Langerman, Pat Morin, and David R. Wood. A polynomial bound for untangling geometric planar graphs. *CoRR*, abs/0710.1641, 2007.

[BFR12] Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Sorting by transpositions is difficult. *SIAM J. Discrete Math.*, 26(3):1148–1180, 2012.

[Bie13] Therese Biedl. A 4-approximation for the height of drawing 2-connected outer-planar graphs. In Thomas Erlebach and Giuseppe Persiano, editors, *Approximation and Online Algorithms*, volume 7846 of *Lecture Notes in Computer Science*, pages 272–285. Springer Berlin Heidelberg, 2013.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976.

[BM99] Oliver Bastert and Christian Matuszewski. Layered drawings of digraphs. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs*, volume 2025 of *Lecture Notes in Computer Science*, pages 87–120. Springer, 1999.

[BS96] Hanspeter Bieri and Peter-Michael Schmidt. On the permutations generated by rotational sweeps of planar point sets. In Frank Fiala, Evangelos Kranakis, and Jörg-Rüdiger Sack, editors, *CCCG*, pages 179–184. Carleton University Press, 1996.

[CC13] Vincent A. Cicirello and Robert Cernera. Profiling the distance characteristics of mutation operators for permutation-based genetic algorithms. In Chutima Boonthum-Denecke and G. Michael Youngblood, editors, *FLAIRS Conference*. AAAI Press, 2013.

[CLZ10] Gary Chartrand, Linda Lesniak, and Ping Zhang. *Graphs & Digraphs, Fifth Edition*. Chapman & Hall/CRC, 5th edition, 2010.

[CN95] Marek Chrobak and Shin-ichi Nakano. Minimum-width grid drawings of plane graphs extend abstract. In Roberto Tamassia and IoannisG. Tollis, editors, *Graph Drawing*, volume 894 of *Lecture Notes in Computer Science*, pages 104–110. Springer Berlin Heidelberg, 1995.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[Cou90] Bruno Courcelle. The monadic second-order logic of graphs i. recognizable sets of finite graphs. *Information and Computation*, pages 12–75, 1990.

[DFK+08] Vida Dujmović, Michael R. Fellows, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Frances A. Rosamond, Sue Whitesides, and David R. Wood. On the parameterized complexity of layered graph drawing. *Algorithmica*, 52(2):267–292, 2008.

[Dil90] R.P. Dilworth. A decomposition theorem for partially ordered sets. In KennethP. Bogart, Ralph Freese, and JosephP.S. Kung, editors, *The Dilworth Theorems*, Contemporary Mathematicians, pages 7–12. Birkhäuser Boston, 1990.

[DL11] Vida Dujmović and Stefan Langerman. A center transversal theorem for hyperplanes and applications to graph drawing. In *Proceedings of the Twenty-seventh Annual Symposium on Computational Geometry*, SoCG '11, pages 117–124, New York, NY, USA, 2011. ACM.

[EBFK09a] Alejandro Estrella-Balderrama, J. Joseph Fowler, and Stephen G. Kobourov. Characterization of unlabeled level planar trees. *Comput. Geom.*, 42(6-7):704–721, 2009.

[EBFK09b] Alejandro Estrella-Balderrama, J. Joseph Fowler, and Stephen G. Kobourov. On the characterization of level planar trees by minimal patterns. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing*, volume 5849 of *Lecture Notes in Computer Science*, pages 69–80. Springer, 2009.

[EFLN06] Peter Eades, Qingwen Feng, Xuemin Lin, and Hiroshi Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. *Algorithmica*, 44(1):1–32, 2006.

[Eul58] Leonhard Euler. Demonstratio nonnullarum insignium proprietatum, quibus solida hedris planis inclusa sunt praedita. *Novi Commentarii academiae scientiarum Petropolitanae*, 4:140–160, 1758. Presented to the St. Petersburg Academy on April 6, 1752. *Opera Omnia* I.26:94–108.

[EW94] Peter Eades and Sue Whitesides. Drawing graphs in two layers. *Theor. Comput. Sci.*, 131(2):361–374, 1994.

[Far48] Fary. On straight lines representation of planar graphs. *Acta Sci. Math. Szeged*, 11:229 – 233, 1948.

[FK07a] J. Joseph Fowler and Stephen G. Kobourov. Characterization of unlabeled level planar graphs. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Graph Drawing*, volume 4875 of *Lecture Notes in Computer Science*, pages 37–49. Springer, 2007.

[FK07b] J. Joseph Fowler and Stephen G. Kobourov. Minimum level nonplanar patterns for trees. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Graph Drawing*, volume 4875 of *Lecture Notes in Computer Science*, pages 69–75. Springer, 2007.

[FPSS13] Radoslav Fulek, MichaelJ. Pelsmajer, Marcus Schaefer, and Daniel Stefankovic. Hanani–tutte, monotone drawings, and level-planarity. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 263–287. Springer New York, 2013.

[GJ83] M. Garey and D. Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983.

[GJS74] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pages 47–63, New York, NY, USA, 1974. ACM.

[GKO+09] Xavier Goaoc, Jan Kratochvíl, Yoshio Okamoto, Chan-Su Shin, Andreas Spillner 0001, and Alexander Wolff. Untangling a planar graph. *Discrete & Computational Geometry*, 42(4):542–569, 2009.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization.* Springer4060 XII, 362 S, 1988.

[GNPR05] Arvind Gupta, Naomi Nishimura, Andrzej Proskurowski, and Prabhakar Ragde. Embeddings of k-connected graphs of pathwidth k. *Discrete Applied Mathematics*, 145(2):242 – 265, 2005. Structural Decompositions, Width Parameters, and Graph Labelings.

[Goo80] Jacob E. Goodman. On the combinatorial classification of nondegenerate configurations in the plane. *J. Comb. Theory, Ser. A*, 29(2):220–235, 1980.

[Hal76] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8(1-2):171–186, 1976.

[Hir75] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343, June 1975.

[HK99a] Patrick Healy and Ago Kuusik. The vertex-exchange graph: A new concept for multi-level crossing minimisation. In Jan Kratochvíl, editor, *Graph Drawing*, volume 1731 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 1999.

[HK99b] Patrick Healy and Ago Kuusik. The vertex-exchange graph and its use in multi-level graph layout. In *In Kratochvil [Kra99*, pages 205–216, 1999.

[HKL04] Patrick Healy, Ago Kuusik, and Sebastian Leipert. A characterization of level planar graphs. *Discrete Mathematics*, 280(1–3):51 – 63, 2004.

[HN09] S. Hong and H. Nagamochi. Testing planarity of level graphs with intra-level edges. Technical report, Department of Applied Mathematics and Physics, Graduate School of Informatics, University of Kyoto, Japan, 2009.

[HR89] Lenwood S. Heath and Arnold L. Rosenberg. Graph layout using queues. Technical report, Blacksburg, VA, USA, 1989.

[HT74] John Hopcroft and Robert Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, October 1974.

[JKR13] Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter. A kuratowski-type theorem for planarity of partially embedded graphs. *Computational Geometry*, 46(4):466 – 492, 2013. 27th Annual Symposium on Computational Geometry (SoCG 2011).

[JL99] Michael Juenger and Sebastian Leipert. Level planar embedding in linear time, 1999.

[JLM98] Michael Jünger, Sebastian Leipert, and Petra Mutzel. Level planarity testing in linear time. In SueH. Whitesides, editor, *Graph Drawing*, volume 1547 of *Lecture Notes in Computer Science*, pages 224–237. Springer Berlin Heidelberg, 1998.

[kSH99] Wei kuan Shih and Wen-Lian Hsu. A new planarity test, 1999.

[Kur30] Casimir Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930.

[Len90] Thomas Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* John Wiley & Sons, Inc., New York, NY, USA, 1990.

[Lic82] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[Lin92] Xuemin Lin. *Analysis of algorithms for drawing graphs.* University of Queensland, 1992.

[MA11] Debajyoti Mondal, Muhammad Jawaherul Alam, and Md. Saidur Rahman 0001. Minimum-layer drawings of trees - (extended abstract). In Naoki Katoh and Amit Kumar, editors, *WALCOM*, volume 6552 of *Lecture Notes in Computer Science*, pages 221–232. Springer, 2011.

[MNA11] Debajyoti Mondal, Rahnuma Islam Nishat, Md. Saidur Rahman 0001, and Muhammad Jawaherul Alam. Minimum-area drawings of plane 3-trees. *J. Graph Algorithms Appl.*, 15(2):177–204, 2011.

[MNR13] Tamara Mchedlidze, Martin Nöllenburg, and Ignaz Rutter. Drawing planar graphs with a prescribed inner face. In Stephen Wismath and Alexander Wolff, editors, *Graph Drawing*, volume 8242 of *Lecture Notes in Computer Science*, pages 316–327. Springer International Publishing, 2013.

[MSM99] Christian Matuszewski, Robby Schönfeld, and Paul Molitor. Using sifting for k -layer straightline crossing minimization. In Jan Kratochvíl, editor, *Graph Drawing*, volume 1731 of *Lecture Notes in Computer Science*, pages 217–224. Springer, 1999.

[MT69] Dennis M. Moyles and Gerald L. Thompson. An algorithm for finding a minimum equivalent graph of a digraph. *J. ACM*, 16(3):455–460, July 1969.

[Muh14] Rashid Bin Muhammad. *Graph Planarity*, 2005 (accessed May 23, 2014). `http://www.personal.kent.edu/~rmuhamma/GraphTheory/MyGraphTheory/planarity.htm`.

[Mut01] P. Mutzel. An alternative method to crossing minimization on hierarchical graphs. *SIAM Journal on Optimization*, 11(4):1065–1080, 2001.

[OSH87] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 224–230, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[Ote14] Daniel E. Otero. *Kuratowski's Theorem*, 2001 (accessed May 23, 2014). `http://www.cs.xu.edu/~otero/math330/kuratowski.html`.

[Pat06] Maurizio Patrignani. On extending a partial straight-line drawing. In Patrick Healy and NikolaS. Nikolov, editors, *Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 380–385. Springer Berlin Heidelberg, 2006.

[Pet98] Julius Petersen. Sur le théoreme de tait. *L'intermédiaire des Mathématiciens*, 5:225–227, 1898.

[PT02] János Pach and Gézá Tóth. Monotone drawings of planar graphs. In Prosenjit Bose and Pat Morin, editors, *Algorithms and Computation*, volume 2518 of *Lecture Notes in Computer Science*, pages 647–653. Springer Berlin Heidelberg, 2002.

[RS83] Neil Robertson and P.D. Seymour. Graph minors. i. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39 – 61, 1983.

[RS84] Neil Robertson and P.D Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49 – 64, 1984.

[RS91] Neil Robertson and P.D Seymour. Graph minors. x. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153 – 190, 1991.

[RSB+01] Bert Randerath, Ewald Speckenmeyer, Endre Boros, Peter Hammer, Alex Kogan, Kazuhisa Makino, Bruno Simeone, and Ondrej Cepek. A satisfiability formulation of problems on level graphs. *Electronic Notes in Discrete Mathematics*, 9(0):269 – 277, 2001. {LICS} 2001 Workshop on Theory and Applications of Satisfiability Testing (SAT 2001).

[Rud14] Steven Rudich. *Planar Graphs*, 2006 (accessed May 23, 2014). `www.cse.yorku.ca/~jeff/courses/new_course/25planarity.ppt`.

[Sch89] Walter Schnyder. Planar graphs and poset dimension. *Order*, 5(4):323–343, 1989.

[Sch90] Walter Schnyder. Embedding planar graphs on the grid. In David S. Johnson, editor, *SODA*, pages 138–148. SIAM, 1990.

[Ski90] Steven Skiena. *Implementing discrete mathematics - combinatorics and graph theory with Mathematica.* Addison-Wesley, 1990.

[ST94] P.D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[Ste51] Sherman K. Stein. Convex maps. *Proc. Amer. Math. Soc.*, 2(3):464–466, 1951.

[STT81] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man & Cybernetics*, 11(2):109–125, 1981.

[Sud04] Matthew Suderman. Pathwidth and layered drawings of trees. *Int. J. Comput. Geometry Appl.*, 14(3):203–225, 2004.

[Wag36] K. Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

[Wag37] K. Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–590, 1937.

[wag14] *Proof of Wagner's theorem*, 2008 (accessed May 23, 2014). `http://planetmath.org/proofofwagnerstheorem`.