Planarity Variants for Directed Graphs

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)
genehmigte

Dissertation

von

Guido Brückner

Tag der mündlichen Prüfung:	19. Februar 2021
Erste Referentin:	Prof. Dr. Dorothea Wagner
Zweiter Referent:	Prof. Dr. Ignaz Rutter
Dritter Referent:	Prof. Dr. Giuseppe Di Battista

Danksagung

Gleich zu Beginn meines Studiums der Informatik stellte ich fest, dass mich vor allem die theoretischen Aspekte der Informatik interessierten. Ich vertiefte mich in diese Richtung weiter und hörte so die meisten theoretisch geprägten Informatikvorlesungen, die am KIT angeboten wurden. Während der Promotion durfte ich nun mehrere Jahre lang selber theoretische Informatik betreiben – dadurch ist ein großer Traum wahr geworden.

Bei Ignaz Rutter möchte ich mich für die absolut erstklassige Betreuung bedanken. Die regelmäßigen Forschungstreffen (ob virtuell über Skype oder persönlich in Eindhoven und Passau), die hunderten ausführlich rot annotierten PDF-Seiten und die unzähligen Stunden, in denen wir gemeinsam im Deadlinestress bis tief in die Nacht an unseren Einreichungen gearbeitet haben, haben ganz entscheidend zu dieser Dissertation beigetragen.

Ich danke Dorothea Wagner dafür, dass sie mir die Möglichkeit geboten hat, zu promovieren, und dass ich in ihrer Gruppe immer gut aufgehoben war. Besonders gut werden mir die zahlreichen Konferenzen und Summer Schools, zu denen ich reisen durfte, in Erinnerung bleiben. Sehr dankbar bin ich auch für ihre große Flexibilität, die es mir ermöglichte, für ein paar Monate über den Tellerrand hinaus auf die angewandte Seite der Informatik zu blicken.

I want to thank Giuseppe Di Battista for kindly accepting to referee my thesis and for navigating the formalities of my defense during the pandemic. More broadly, I want to thank the whole graph drawing and algorithms community for being very welcoming, very down to earth, and for being truly dedicated to the spirit of science. It has been a great pleasure to be a part of this community.

Ganz herzlich möchte ich mich außerdem bei meinen Kollegen Lukas Barth, Moritz Baum, Valentin Buchhold, Lars Gottesbüren, Sascha Gritzbach, Michael Hamann, Paul Jungeblut, Tamara Mchedlidze, Laura Merker, Benjamin Niedermann, Roman Prutkin, Marcel Radermacher, Jonas Sauer, Ben Strasser, Torsten Ueckerdt, Franziska Wegner, Matthias Wolf, Tim Zeitz und Tobias Zündorf bedanken. Es war eine sehr schöne Zeit mit euch! Ich denke sehr gerne an die regelmäßigen Kaffeerunden, die Streifzüge der antikapitalistischen Automatenfront und die stets aufreibenden wie fairen Kickerrunden zurück. Besonderer Dank geht an meinen langjährigen Bürokollegen Jonas: zusammen konnten wir im Detail Freude und Leid über die Forschung, die Lehre (im Allgemeinen und über TGI im Besonderen) und über die Studenten teilen. Ich danke Marcel für die zahlreichen Reisen zusammen, und Valentin für das gemeinsame wackere Standhalten gegen Corona.

Schließlich danke ich Lilian Beckert, Isabelle Junge, Laurette Lauffer und Tanja Wehrmann für ihre ausdauernde Hilfe im Kampf mit der Bürokratie. Bei Ralf Kölmel bedanke ich mich für die Pflege der technischen Infrastruktur, auf die ich mich immer verlassen konnte.

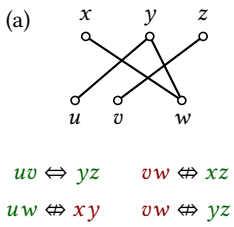
Deutsche Zusammenfassung

Diese Dissertation trägt zur Theorie von planaren Graphen bei, insbesondere zu Planaritätsvarianten für gerichtete Graphen. Sie ist in drei Teile gegliedert. Im ersten Teil behandeln wir das Testen eines Graphen auf (radiale) Levelplanarität. Wir räumen einige der Probleme, die aktuelle Ansätze mit sich bringen, aus dem Weg und untersuchen den Fall, wenn die Einbettung des zu testenden Graphen fest ist. Der zweite Teil befasst sich mit eingeschränkten Einbettungsproblemen. Hier untersuchen wir zwei Ansätze, nämlich erstens die PQ-Baum Datenstruktur zu erweitern, um mit Teileinbettungen levelplanarer Graphen umgehen zu können, und zweitens SPQR-Baumartige Einbettungsrepräsentation für gerichtete planare Graphen zu entwickeln, mithilfe derer wir Algorithmen für ungerichtete planare Graphen einfach auf den gerichteten Fall übertragen können. Schließlich werden im dritten Teil neue Zeichenstile eingeführt und analysiert. Wir betrachten besonders symmetrische Zeichnungen mit wenigen Steigungen, eine Verallgemeinerung von Level- und Aufwärtsplanarität und Zeichnungen, die gleichzeitig zwei Ordnungen auf den Knoten eines Graphen darstellen, wobei eine Ordnung von unten nach oben und die andere Ordnung von links nach rechts visualisiert wird.

Teil I — Levelplanaritätsteste

Im ersten Teil dieser Arbeit betrachten wir drei Aspekte davon, einen Graphen auf (radiale) Levelplanarität zu testen. Zunächst untersuchen wir das Testen von Graphen auf Levelplanarität unter der Bedingung, dass die Einbettung des Graphen fest ist. Dabei betrachten wir drei unterschiedliche Definition einer Einbettung, nämlich

das Fixieren der zyklischen Ordnung von Kanten um jeden Knoten, das Fixieren der linearen Ordnungen von ein- und ausgehenden Kanten um jeden Knoten, und schließlich das Fixieren der linearen Ordnung von Knoten und Kanten auf jedem Level. Wir geben für jeden dieser Fälle einen Linearzeitalgorithmus an, der das entsprechende Problem löst. Außerdem charakterisieren wir radiale aufwärtsplanare Einbettungen, was uns unsere Ergebnisse für radiale Levelplanarität erweitern lässt.



2-SAT Formulierung von Levelplanarität (a) und Hanani-Tutte-Zeichnung (b).

Der besonders einfache Levelplanaritätstest mit quadratischer Laufzeit von Randerath et al. [Ran+01] reduziert das Testen auf Levelplanarität auf das Problem 2-SAT. Allerdings haben manche Autoren Zweifel an der Vollständigkeit des dazugehörigen Korrektheitsbeweises geäußert [FPSŠ13]. Kürzlich haben Fulek et al. [FPSŠ13, FPS17, FPS16] eine Hanani-Tutte-artige topologische Charakterisierung von (radialer) Levelplanarität gegeben, d.h., ein Graph ist (radial) levelplanar, wenn er eine (radiale) Levelzeichnung besitzt, in der sich alle Paare von unabhängigen Kanten gerade oft kreuzen. Wir zeigen, dass die 2-SAT-Formulierung von Levelplanarität von Bachmaier et al. äquivalent zu dem starken Hanani-Tutte Theorem für Levelplanarität ist. Unser Resultat kann entweder als vollständiger Korrektheitsbeweis des Algorithmus von Randerath et al. interpretiert werden, oder, wenn man von dessen Korrektheit bereits überzeugt war, als alternativer, einfacher Beweis des Hanani-Tutte Theorems für Levelplanarität angesehen werden. Wir zeigen außerdem, dass im radialen Fall ein ähnlicher Zusammenhang besteht. Dadurch ergibt sich ein neuer, sehr einfacher Testalgorithmus für radiale Levelplanarität in quadratischer Zeit.

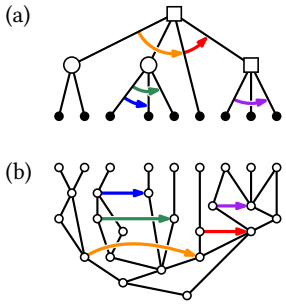
Danach beschreiben wir einen Linearzeitalgorithmus, der einen Levelgraphen auf (radiale) Levelplanarität testet und, sollte das Ergebnis positiv sein, eine (radiale) levelplanare Einbettung berechnet. Unser Algorithmus folgt dem etablierten Ansatz, den Graphen Level für Level zu bearbeiten und dabei einen PQ-Baum pro Zusammenhangskomponente zu verwalten. Im Gegensatz zu den bekannten Algorithmen nutzen wir die deutlich vereinfachte Variante des PQ-Baums von Hsu und McConnell [HM03] und beschriften diese anders, nämlich mit Knoten aus G die bezeugen, wie viel Platz zwischen zwei benachbarten Knoten einer Zusammenhangskomponente ist. Wachsen mehrere Zusammenhangskomponenten zusammen, so müssen die entsprechenden PQ-Bäume mitsamt ihrer Beschriftungen verschmolzen werden.Im Vergleich zu den bekannten, sehr komplexen Algorithmen ist unser Al-

gorithmus vergleichsweise einfach. Außerdem behandelt er einen wichtigen Fall, der von Bachmaier et al. [BBF05] scheinbar übersehen wurde. Damit ist unser Algorithmus der erste vollständige Linearzeitalgorithms für das Testen von radialer Levelplanarität.

Teil II — Eingeschränkte Einbettungen

Oftmals sucht man nicht nach einer beliebigen planaren Einbettung eines Graphen, sondern einer, die zusätzlichen Einschränkungen genügt. Eine gängige Frage ist etwa, ob sich eine Einbettung eines Teils eines Graphen auf eine planare Einbettung des gesamten Graphen erweitern lässt, ohne die Teileinbettung dabei zu verändern. Eine weitere Frage ist, ob zwei Graphen G_1, G_2 planare Einbettungen besitzen, deren Einschränkungen auf den gemeinsamen Graphen $G_1 \cap G_2$ identisch sind. Im planaren Fall spielt der SPQR-Baum eine entscheidene Rolle bei der Entwicklung von Algorithmen, die diese Fragen beantworten.

Im zweiten Teil dieser Arbeit betrachten wir eingeschränkte Einbettungen von gerichteten planaren Graphen. Eine eingeschränkte Einbettung eines Levelgraphen besteht aus Halbordnungen der Knoten und Kanten auf jedem Level. Wir verfolgen zwei Ansätze. Als Erstes erweitern wir PQ-Bäume, die sich als nützliches Werkzeug für Algorithmen für levelplanare Graphen erwiesen haben, um in Polynomialzeit eingeschränkte Einbettungen zu finden; siehe die Abbildung rechts. Außerdem zeigen wir, dass der allgemeine Fall NP-schwer ist.



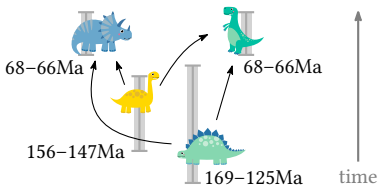
Levelgraph mit eingeschränkter Einbettung (a) und entsprechender PQ-Baum (b).

Ein Nachteil von PQ-Bäumen ist, dass sie nur eine lokale Sichtweise auf Graphen und deren Einbettungen ermöglichen. Im Gegensatz dazu bieten SPQR-Bäume eine globale Sichtweise auf alle planaren Einbettungen eines Graphen. Als Zweites entwickeln wir deshalb SPQR-Baumartige Einbettungsrepräsentationen für zweifachzusammenhängende levelplanare und aufwärtsplanare Graphen mit nur einer Quelle, die wir als LP-Baum und UP-Baum bezeichnen. Beide Bäume teilen wichtige Eigenschaften von SPQR-Bäumen. Dies ermöglicht es uns Algorithmen, die auf SPQR-Bäumen basieren, auf einfache Art und Weise für die gerichteten Planaritätsvarianten zu adaptieren. So konstruieren wir einen Algorithms für das eingeschränkte Einbettungsproblem für levelplanare Graphen, dessen Laufzeit besser als die des auf PQ-Bäumen basierenden Algorithmus ist. Außerdem lösen wir einen Fall des simultanen Einbettungsproblems

für levelplanare Graphen und, mit UP-Bäumen, auch das eingeschränkte Einbettungsproblem für aufwärtsplanare Graphen.

Teil III — Neue Zeichenstile

Im dritten Teil dieser Arbeit betrachten wir neue Zeichenstile für gerichtete planare Graphen. Zunächst führen wir geradlinige levelplanare Zeichnugen mit einer festen Anzahl λ von Steigungen ein. Für $\lambda = 2$ entspricht dieser Zeichenstil in etwa dem verbreiteten orthogonalen Zeichenstil für planare Graphen. Für Levelgraphen bei denen alle Kanten Knoten benachbarter Level verbinden, geben wir einen Algorithmus mit beinahe linearer Laufzeit an, der entweder eine solche Zeichnung findet, oder feststellt, dass es keine solche Zeichnung gibt. Wir betrachten außerdem das Erweitern von Teilzeichnung und das Finden simultaner Zeichnungen, und kontrastieren diese algorithmischen Ergebnisse mit NP-Schwerebeweisen für den allgemeinen Fall.



Ein phylogenetisches Netzwerk, das die evolutionären Beziehungen zwischen Dinosaurierspezies unter Berücksichtigung der Zeitalter, zu denen sie lebten visualisiert.

Danach betrachten wir das Konzept von *Multilevelplanarität*, einer Verallgemeinerung von Levelplanarität und Aufwärtsplanarität. Sei $G = (V, E)$ ein gerichteter Graph und sei $\ell : V \to \mathcal{P}(\mathbb{Z})$ eine Funktion die jedem Knoten von G eine Menge von ganzen Zahlen zuordnet. Eine aufwärtsplanare Zeichnung von G ist multilevelplanar, wenn die y-Koordinate jedes Knotens $v \in V$ in der Menge $\ell(v)$ liegt; siehe die Abbildung rechts für ein Beispiel. Wir präsentieren Linearzeitalgorithmen für Kreise und eingebette Graphen mit nur einer Quelle Fälle, und NP-Schwereresultate für den allgemeinen Fall.

Schließlich untersuchen wir , ob zwei Halbordnungen über derselben Menge so gezeichnet werden können, dass die eine Halbordnung von unten nach oben und die andere Halbordnung von links nach rechts dargestellt ist. Die Eingabe besteht aus einem gerichteten Graphen $G = (V, E)$ und zwei Mengen X, Y mit $X \cup Y = E$, wobei X und Y als partielle Ordnungen von V interpretiert werden können. Die Aufgabe ist es, eine Zeichnung von G zu finden, in der jede gerichtete Kante in X bzw. Y als x- bzw. y-monotone Kurve gezeichnet ist. Eine solche Zeichnung ist *xy-planar*.

Im Allgemeinen ist es NP-schwer, zu testen, ob eine Graph eine xy-planare Zeichnung erlaubt. Wir untersuchen dne Fall, dass die Einbettung von G fest ist und der durch die Kantenmenge Y induzierte Teilgraph von G ein zusammenhängender und

aufspannender Teilgraph von G ist, dessen aufwärtsplanare Einbettung fest ist. Wir nutzen die Zusammenhänge zwischen xy-Planarität, Aufwärtsplanarität und einer neuartigen Charakterisierung von Windrosenplanarität aus, um einen Linearzeitalgorithmus zu entwickeln, der einen Graphen auf xy-Planarität testet, und im positiven Fall eine xy-planare Zeichnung mit höchstens drei Knicken pro Kante ausgibt.

Contents

1 Introduction

Planarity is a concept that has been extensively studied. Since the 1930s characterizations of planar graphs in terms of the forbidden minors are known [Kur30, Wag37]. The Hanani-Tutte theorem provides a topological characterization of planarity [Cho34, Tut70], and there also exists an algebraic characterization [Ver90]. From an algorithmic perspective, testing a graph for planarity, and, if the graph is planar, finding a planar embedding of it, are classic problems [HT74, SH99, BM04, HT08, dM12]. For visualization purposes, it is often desirable to find a graph drawing that is not just planar, but has additional qualities. Examples include drawing an embedded graph orthogonally [Tam87], with few slopes [DESW07], or symmetrically [HME06]; or inserting an edge or a vertex into a planar drawing so that the number of edge crossings is minimized [GMW05]. Historically, many of these problems have been considered for embedded graphs. More recent research has attempted to optimize not only one fixed embedding, but to optimize across all possible planar embeddings of a graph [GMW05, BRW16]. A common tool in many of these algorithms is the SPQR-tree data structure, which offers a global view on all planar embeddings of a biconnected planar graph [Mac37, Tut66, HT73, DT89, DT90, DT96]. It essentially decomposes the many and complex choices in choosing a planar embedding into fewer and well-structured choices. The SPQR-tree is also a key tool for solving constrained embedding problems such as extending partial embeddings [Ang+15c] and finding simultaneous embeddings [BKR13].

There also exist planarity variants for directed graphs. A drawing of a directed graph is upward if each directed edge is drawn as a y-monotone curve. Level drawings are upward drawings where additionally the y-coordinate of each vertex is fixed.

Planarity variants for directed graphs such as level planarity or upward planarity are not understood quite as well. A characterization of level-planar graphs in terms of minimal forbidden subgraphs is still missing, even for trees [HKL04, FK07, EFK09]. There exists significant research into level planarity testing algorithms. This problem was initially studied by Di Battista and Nardelli who solved this problem for graphs with a single source [DN88]. Their idea is to process the graph in a bottom-up sweep while maintaing all necessary information in form of a PQ-tree. This approach was extended to the general case with multiple sources in a series of publications [HP95, HP99, JLM97, JLM98]. The idea is to perform a bottom-up sweep, maintaining one PQ-tree per connected component and merging them as their components are merged together. This algorithm was augmented to also output a level-planar embedding if one exists [JL02], and also to the radial level planar setting, i.e., on the standing cylinder [BBF05]. Overall, this makes for more than 150 pages of literature. However, the resulting algorithms are notoriously complex and a convincing proof of their correctness is, in our opinion, currently not available. In fact, it seems like the algorithm for radial level planarity testing does not cover an important case. For these reasons, simpler but slower algorithms have been proposed even long after the publication of the linear-time algorithm [Ran+01, HK04, HH07, FPS17]. One particularly simple algorithm is due to Randerath et al. [Ran+01]. However, some authors have expressed doubts about the completeness of the correctness proof of this algorithm [FPSŠ13].

The linear-time level planarity algorithms use the PQ-tree data structure, which offers a localized view of the embeddings of a level graph. It is difficult to qualify how a localized embedding choice interrelates with embedding choices made elsewhere. This makes it challenging to develop algorithms for some constrained embedding problems. Therefore, it would be desirable to have a tool that offers a global view on all level planar / upward planar embeddings, much like the SPQR-tree does for planar embeddings.

1.1 Contribution

This thesis contributes to the theory of planar graphs, in particular to the study of planarity variants tailored to directed graphs. It is split into three parts. In the first part, we study the problem of level planarity testing. We clear up the problems of some of the existing approaches and study the very restricted case when the (upward) planar embedding of the input graph is fixed. The second part of this thesis is concerned with finding planar embeddings of directed graphs subject to certain constraints. Here, we explore two approaches, namely extending the PQ-tree data structure to be able to handle partial and constrained level planar embeddings, and

adapting the SPQR-tree data structure to represent exactly the level planar or the upward planar embeddings of a single-source graph and then translating SPQR-tree-based algorithms to the level planar and upward planar settings. Finally, the third part of this thesis is dedicated to the study of new drawing styles tailored specifically to directed graphs. We investigate particularly symmetric drawings with few slopes, a generalization of level planarity and upward planarity, and drawings that simultaneously capture two relationships on the vertices of a graph, one of which is visualized from the bottom up and the other from left to right.

Part I — Level Planarity Testing

In the first part of this thesis, we study three aspects of (radial) level planarity testing. First, we consider a quite restricted setting, namely when the embedding of the graph is fixed. Second, we show that the simple level planarity testing algorithm due to Randerath et al. is indeed correct, and then extend it to the radial level planar setting. Third, we present a simplified linear-time (radial) level planarity testing algorithm together with a rigorous but accessible proof of correctness.

Level Planarity Testing with Fixed Embedding

In Chapter 3, we consider the level planarity testing problem when the embedding is fixed for three different notions of embedding, namely the level embedding (i.e., the order of vertices and edges on each level is fixed), the upward embedding (i.e., the linear order of incoming and outgoing edges around each vertex is fixed) and the planar embedding (i.e., the cyclic order of edges around each vertex is fixed). These three notions allow for increasing flexibility in the corresponding level planar drawings. We show how to make all faces small by untangling level planar embeddings in order to insert new edges; see Figure 1.1. Graphs with such small faces and fixed (upward) planar embedding can be tested for level planarity in linear time. Moreover, we provide a new characterization of radial upward planar embeddings, which lets us extend these results to radial level planarity.

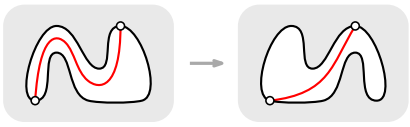


Figure 1.1: Untangling a level planar embedding to insert an edge.

Level Planarity Testing and Hanani-Tutte

The simple quadratic-time algorithm for (non-radial) level planarity testing due to Randerath et al. [Ran+01] reduces level planarity testing to the 2-SATISFIABILI-

TY (2-SAT) problem; see Figure 1.2 (a). However, some authors have expressed doubts about the completeness of the correctness proof of this reduction [FPSŠ13]. Recently, Fulek et al. [FPSŠ13, FPS17, FPS16] have presented a Hanani-Tutte-style topological characterization of (radial) level planarity, i.e., a graph is (radial) level planar if it admits a (radial) level drawing where any two independent edges cross an even number of times; see Figure 1.2 (b) for such a drawing.

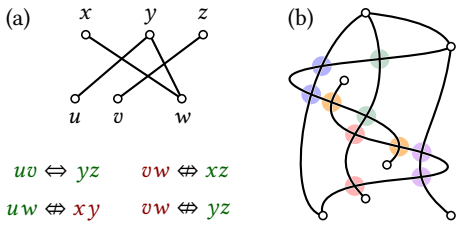


Figure 1.2: 2-SAT formulation of level planarity (a) and Hanani-Tutte drawing (b).

In Chapter 4, we show that the 2-SAT formulation of level planarity testing due to Randerath et al. [Ran+01] is equivalent to the strong Hanani-Tutte theorem for level planarity [FPSŠ13]. Our result can be interpreted either as a complete correctness proof for the algorithm of Randerath et al., or, assuming completeness of their proof, as an alternative, simple proof to the Hanani-Tutte result for level planarity. In addition to this, we show that this relationship carries over to radial level planarity, which yields a novel very simple polynomial-time algorithm for testing radial level planarity.

Level Planarity Testing in Linear Time

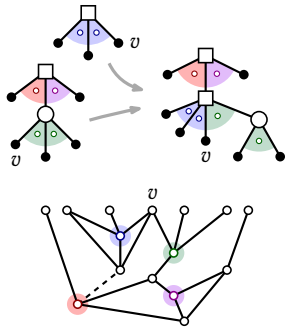


Figure 1.3: Merging PQ-trees to represent a multi-source graph.

In Chapter 5, we present a linear-time algorithm that tests a level graph G for (radial) level planarity and, if the outcome is positive, outputs a (radial) level-planar embedding of G. Our algorithm follows the well-established idea of processing G in a bottom-up sweep, maintaining one PQ-tree per connected component. In contrast to the existing algorithms we use a dramatically simpler PQ-tree data structure due to Hsu and McConnell [HM03] and annotate it differently, namely with vertices of G that witness how much space there is between two vertices of a connected component. As connected components are merged together, their PQ-trees need to be merged as well; see Figure 1.3. Compared with the state of the art, our algorithm is simple. Very importantly, our contribution includes a rigorous proof of correctness of our algorithm. In developing this proof, we found that the algorithm for radial level planarity testing and radial level planar embedding by Bachmaier et al. [BBF05] seems to not handle an important

case. This would make our algorithm the first correct linear-time algorithm for radial level planarity testing and radial level planar embedding.

Part II — Constrained Embeddings

Often, we are not looking for just any embedding of a graph, but one that satisfies certain additional constraints. One popular question asks whether a given partial embedding of a graph G, i.e., an embedding of a subgraph of G, can be extended to an embedding of the entire graph G without modifying the given partial embedding. Another question asks whether two graphs G_1, G_2 admit planar drawings that are identical on the shared graph $G_1 \cap G_2$. In the planar setting, the SPQR-tree plays a crucial role in devising algorithms that answer these questions.

In the second part of this thesis, we consider constrained embeddings of level planar and upward planar graphs. We attack these problems from two angles. First, we study how PQ-trees, which have proven to be a very useful tool to solve problems related to level planarity, can be augmented to find constrained embeddings of level planar graphs. One downside of PQ-trees is that they offer only a localized view of the graph and its embeddings. In comparison, the SPQR-tree offers a global view of all embeddings of a planar graph. Therefore, our second angle of attack is to develop SPQR-tree-like embedding representations for single-source level planar graphs and upward planar graphs and use them to translate SPQR-tree-based constrained embedding algorithms to the directed settings.

Partial and Constrained Level Planarity

In Chapter 6, we study the problem of level planarity testing with partial or constrained level embeddings. A partial level embedding consists of total left-to-right orders of a subset of the vertices and edges on each level. A constrained level embedding consists of a partial left-to-right order of the vertices and edges on each level. We show that for single-source level graphs both of these problems can be solved in polynomial time. To this end, we start with the original level planarity testing algorithm for single-source graphs of Di Battista and Nardelli, and then augment the PQ-tree data structure to also handle partial orders of its leaves; see Figure 1.4.

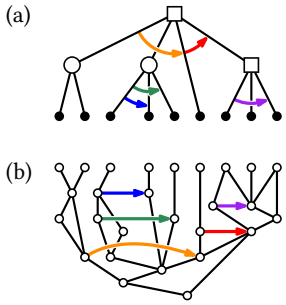


Figure 1.4: Constrained level graph (b) and PQ-tree (a).

We complement this positive result for single-source level graphs by showing that the partial and constrained level planarity problems are NP-hard in general.

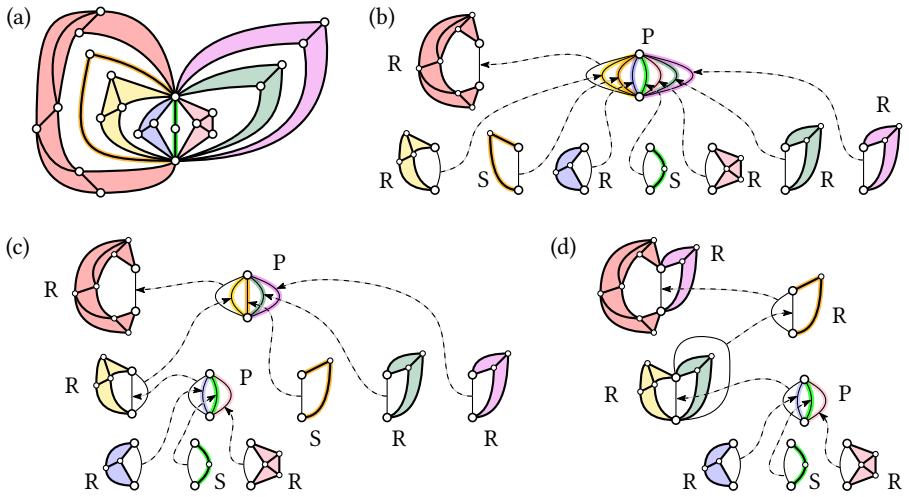


Figure 1.5: A level planar graph (a), the SPQR-tree of its underlying planar graph (b), the UP-tree of its underlying upward planar graph (c), and its LP-tree (d).

SPQR-Tree-Like Embedding Representations

In Chapter 7, we show that biconnected single-source level graphs admit an SPQR-tree-like embedding representation. We formalize this representation under the name LP-tree. LP-trees share many of the important properties of SPQR-trees. This makes it possible to use LP-trees almost as a drop-in replacement for SPQR-trees in algorithms for planar graphs, thereby translating these algorithms from the planar setting to the level-planar setting with only very few modifications. In particular, we translate an algorithm for extending partial and constrained planar embeddings and an algorithm for finding simultaneous planar embeddings to the level-planar setting. The former algorithm improves upon the running time of the algorithm for the same problem presented in Chapter 6, the latter algorithm is the first solution to a previously open problem.

In Chapter 8, we show that an SPQR-tree-like embedding representation also exists for biconnected single-source upward planar graphs. We dub this representation UP-tree. We demonstrate the usefulness of the UP-tree by translating the partial embedding algorithm for planar graphs to the upward planar setting, using the UP-tree as a drop-in replacement for the SPQR-tree.

See Figure 1.5 for an example of a graph, the SPQR-tree, the LP-tree and the UP-tree.

Part III — New Drawing Styles

In the third part of this thesis we consider three new drawing styles for planar directed graphs.

Level Planar Drawings with Few Slopes

In Chapter 10, we introduce and study level-planar straight-line drawings with a fixed number λ of slopes; see Figure 1.6. For $\lambda = 2$ this drawing style is somewhat akin to the popular orthogonal drawing style for planar graphs. For proper level graphs (all edges connect vertices of adjacent levels), we give an almost-linear-time algorithm that either finds such a drawing or determines that no such drawing exists. Moreover, we consider the partial drawing extension problem, where we seek to extend an immutable drawing of a subgraph to a drawing of the whole graph, and the simultaneous drawing problem, which asks about the existence of drawings of two graphs whose restrictions to their shared subgraph coincide. We present efficient algorithms for these respective problems on proper level-planar graphs. These positive results are complemented by our result that testing whether non-proper level graphs admit level-planar drawings with λ slopes is NP-hard even in restricted cases.

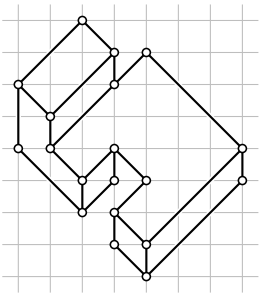


Figure 1.6: A drawing with three slopes.

Multilevel Planarity

In Chapter 9, we introduce and study *multilevel planarity*, a generalization of upward planarity and level planarity. Let $G = (V, E)$ be a directed graph and let $\ell : V \to \mathcal{P}(\mathbb{Z})$ be a function that assigns a finite set of integers to each vertex. A multilevel-planar drawing of G is an upward planar drawing of G such that for each vertex $v \in V$ its y-coordinate $y(v)$ is in $\ell(v)$. See Figure 1.7 for an example. We present linear-time algorithms for testing multilevel planarity of embedded graphs with a single source and of oriented cycles. Complementing these algorithmic results, we show that multilevel-planarity testing is NP-complete even in very restricted cases.

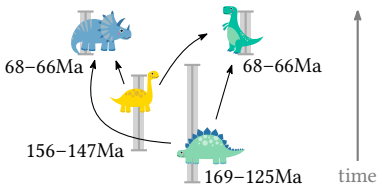


Figure 1.7: Phylogenetic network showing the evolutionary relationships of dinosaur species with respect to the times when they roamed the earth.

Drawing Two Posets

In Chapter 11, we investigate the problem of drawing two posets of the same ground set so that one is drawn from left to right and the other one is drawn from the bottom up. The input to this problem is a directed graph $G = (V, E)$ and two sets X, Y with $X \cup Y = E$, each of which can be interpreted as a partial order of V. The task is to find a planar drawing of G such that each directed edge in X is drawn as an x-monotone edge, and each directed edge in Y is drawn as a y-monotone edge. Such a drawing is called an *xy-planar* drawing.

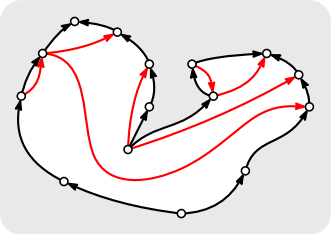


Figure 1.8: Inserting rightward edges (red) into a face of an upward planar graph.

Testing whether a graph admits an xy-planar drawing is NP-complete in general. We consider the case that the planar embedding of G is fixed and the subgraph of G induced by the edges in Y is a connected spanning subgraph of G whose upward embedding is fixed. Exploiting the relationships between xy-planarity, upward planarity and a new combinatorial characterization of windrose planarity, we present a linear-time algorithm that determines whether the edges in X can be inserted into the upward planar drawing of the graph spanned by the edges in Y (see Figure 1.8) and, if so, produces an xy-planar polyline drawing with at most three bends per edge.

2 Terminology

In this chapter we introduce some basic terminology used throughout this thesis.

Graphs. An *undirected graph* is a tuple $G = (V, E)$ that consists of a set of vertices V and a set of edges $E \subseteq \{\{u, v\} \mid u, v \in V\}$. A *directed graph* is a tuple $G = (V, E)$ that consists of a set of vertices V and a set of (directed) edges $E \subseteq \{(u, v) \mid u, v \in V\}$. An edge whose endpoints are identical is a *loop*. In this work we consider graphs without loops. A *simple* graph has at most one edge between any pair of distinct vertices (although if the graph is directed, two vertices u, v may be connected by two edges (u, v) and (v, u) in opposite directions). In contrast, in a *multigraph* there may exist more than one edge between the same two vertices. A *subgraph* of G is a graph $G' = (V', E')$ with $V' \subseteq V$ such that the edges in $E' \subseteq E$ connect vertices in V'. In a directed graph, a vertex v of G is a *source* if all edges incident to v originate from v. Symmetrically, a vertex v of G is a *sink* if all edges incident to v have v as their endpoint. A *cutvertex* of G is a vertex whose removal disconnects G. We say that G is *biconnected* if it has no cutvertex. Further, $\{u, v\}$ is a *cutpair* if there are connected subgraphs H_1, H_2 of G with $H_1 \cup H_2 = G$ and $H_1 \cap H_2 = \{u, v\}$ that both contain at least two edges. If a graph has no cutpair it is *triconnected*.

Drawings and Embeddings of Graphs. A *drawing* of G maps each vertex to a point in the plane and each edge to a finite polygonal chain (i.e., a connected series of line segments) connecting its endpoints such that for each $v \in V$ the line segments incident to v of pairwise distinct edges have pairwise distinct slopes. A drawing Γ of G *extends* a drawing Δ of a subgraph H of G if the restriction of Γ to H is Δ. A

drawing is *planar* if distinct vertices do not coincide and no edges coincide, except in common endpoints. A *(combinatorial) embedding* of G consists of a counter-clockwise cyclic order of edges incident to v for each $v \in V$. An embedding $\mathcal{G}$ of G *extends* an embedding $\mathcal{H}$ of a subgraph H of G if the restriction of $\mathcal{G}$ to H is $\mathcal{H}$, i.e., if all edges e, f, g in H that share a common endpoint v appear in the same cyclic order around v in $\mathcal{G}$ and $\mathcal{H}$. Note that a drawing Γ of G induces a (combinatorial) embedding of G, which we refer to as the embedding of Γ. An embedding of G is *planar* if it is induced by a planar drawing of G. A planar embedding defines faces bounded by *facial walks* $v_1, v_2, \ldots, v_n, v_{n+1} = v_1$ where for each i with $1 \leq i \leq n$ the edge $\{v_{i-1}, v_i\}$ immediately precedes the edge $\{v_i, v_{i+1}\}$ in the counter-clockwise cyclic order of edges incident to v_i.

The terms *drawing* and *planar drawing* extend to directed graphs without modifications. A drawing Γ is *upward* if each edge $(u, v) \in E$ increases strictly y-monotonically from u to v. It is *upward planar* if it is both upward and planar. A *(combinatorial) upward embedding* of G consists of two left-to-right linear orders for each vertex $v \in V$, namely one order each of the incoming and outgoing edges incident to v. Observe that an upward drawing Γ induces an upward embedding of G, which we refer to as the upward embedding of Γ. An upward embedding of G is *upward planar* if it is induced by an upward planar drawing of G.

There is another, alternative view of upward embeddings due to Bertolazzi et al. [BDLM94]. A *source/sink assignment* ψ of G assigns to each source or sink v of G an ordered pair (e, e') of edges where e immediately precedes e' in the counter-clockwise order of edges incident to v. There exists a unique face f on whose facial walk e immediately precedes e', we also say that ψ assigns v to f. An upward planar embedding induces a source/sink assignment by assigning each source v to (e, e'), where e and e' are the leftmost and rightmost outgoing edge incident to v, respectively, and by assigning each sink v to (e, e'), where e and e' are the rightmost and leftmost outgoing edge incident to v, respectively. Conversely, a planar embedding together with a source/sink assignment induces an upward embedding. We therefore use these concepts interchangeably. Let f be a face and let $v_1, v_2, \ldots, v_n, v_{n+1} = v_1$ denote the facial walk that bounds f. For $1 \leq i \leq n$ the vertex v_i is a *face source* of f if both edges $\{v_{i-1}, v_i\}$ and $\{v_i, v_{i+1}\}$ originate from v_i. Symmetrically, v_i is a *face sink* of f if both edges $\{v_{i-1}, v_i\}$ and $\{v_i, v_{i+1}\}$ have v_i as their endpoint. For each face f let n_f denote the number of face sources of f (note that cutvertices of G may be counted multiple times), which equals the number of face sinks of f.

For $k \in \mathbb{N}$ a *k-level graph* (or simply a *level graph*) is a directed graph $G = (V, E)$ together with a *level assignment* $\ell : V \to \{1, 2, \ldots, k\}$ that satisfies $\ell(u) < \ell(v)$ for each edge $(u, v) \in E$. It is *proper* if for each $(u, v) \in E$ it is $\ell(u) = \ell(v) - 1$. It admits a *proper subdivision* that is obtained from G by subdividing each edge (u, v) where $\ell(u) < \ell(v) - 1$ with additional vertices until it becomes proper. A drawing Γ

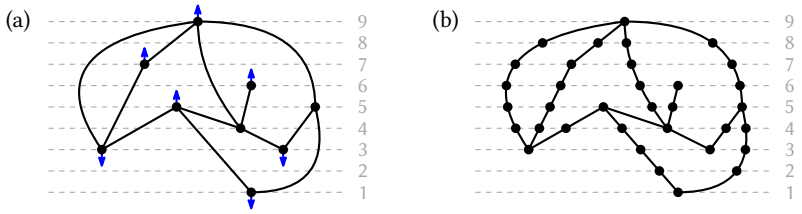


Figure 2.1: A level-planar drawing of a level graph (a) where the blue arrows indicate the source/sink assignment of the underlying upward planar embedding, and its proper subdivision.

of G is a *level drawing* if for each vertex v of G the y-coordinate of v in Γ is $\ell(v)$. A level drawing that is planar is *level planar*. A *(combinatorial) level embedding* of G is an upward embedding of G together with for each level a left-to-right linear order of vertices that lie on that level and edges that cross that level. Each level drawing Γ induces a (combinatorial) level embedding, which we refer to as the level embedding of Γ. A level embedding of G is *level planar* if it is induced by a level planar drawing of G.

In a *radial* drawing the vertices of G are not mapped onto the plane, but onto the standing cylinder. The terms defined above extend to radial drawings without modifications, except for (combinatorial) level embeddings. A *(combinatorial) radial level embedding* of G defines for each level a counter-clockwise cyclic order of vertices that lie on that level and edges that cross that level.

Decomposition Trees. Our description of decomposition trees follows Angelini et al. [ABR14]. Let G be a biconnected graph. Let $\{u, v\}$ be a cutpair and let H_1, H_2 be two subgraphs of G with $H_1 \cup H_2 = G$ and $H_1 \cap H_2 = \{u, v\}$. Define the tree $\mathcal{T}$ that consists of two *nodes* μ_1 and μ_2 connected by an undirected *arc* as follows. For $i = 1, 2$ node μ_i is equipped with a multigraph $\text{skel}(\mu_i) = H_i + e_i$, called its *skeleton*, where $e_i = (u, v)$ is called a *virtual* edge. The arc (μ_1, μ_2) links the two virtual edges e_i in $\text{skel}(\mu_i)$ with each other. We also say that the virtual edge e_1 *corresponds* to μ_2 and likewise that e_2 corresponds to μ_1. The idea is that $\text{skel}(\mu_1)$ provides a more abstract view of G where e_1 serves as a placeholder for H_2. More generally, there is a bijection $\text{corr}_\mu \colon E(\text{skel}(\mu)) \to N(\mu)$ that maps every virtual edge of $\text{skel}(\mu)$ to a neighbor of μ in $\mathcal{T}$, and vice versa. If it is $\text{corr}_\mu((u, v)) = \nu$, then ν is said to have *poles* u and v in μ. If μ is clear from the context we simply say that ν has poles u, v. When the underlying graph is a level graph, we assume $\ell(u) \leq \ell(v)$ without loss of generality. For an arc (ν, μ) of $\mathcal{T}$, the virtual edges e_1, e_2 with $\text{corr}_\mu(e_1) = \nu$

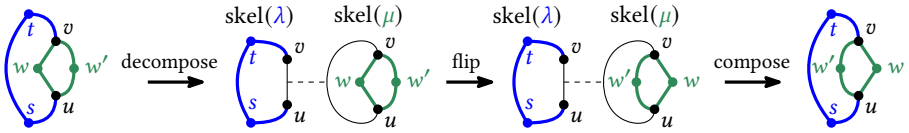


Figure 2.2: Decompose the embedded graph G on the left at the cutpair u, v. This gives the center-left decomposition tree whose skeletons are embedded as well. Reflecting the embedding of skel(μ) or, equivalently, flipping (λ, μ), yields the same decomposition tree with a different embedding of skel(μ). Then contract (λ, μ) to obtain the embedding on the right.

and $\text{corr}_\nu(e_2) = \mu$ are called *twins*, and e_1 is called the *twin* of e_2 and vice versa. This procedure is called a *decomposition*, see Fig. 2.2 on the left. It can be re-applied to skeletons of the nodes of $\mathcal{T}$, which leads to larger trees with smaller skeletons. A *decomposition tree* of G is an unrooted tree obtained in this way. A decomposition can be undone by *contracting* an arc (μ_1, μ_2) of $\mathcal{T}$, forming a new node μ with a larger skeleton as follows. Let e_1, e_2 be twin edges in skel(μ_1), skel(μ_2). The skeleton of μ is the union of skel(μ_1) and skel(μ_2) without the two twin edges e_1, e_2. We call this a *composition*. Contracting all arcs of a decomposition tree of G results in a decomposition tree consisting of a single node whose skeleton is G. See Fig. 2.2 on the right. Let μ be a node of a decomposition tree with a virtual edge e with $\text{corr}_\mu(e) = \nu$. The *expansion graph* of e and ν in μ, denoted by $G(e)$ and $G(\mu, \nu)$, respectively, is the graph obtained by removing the twin of e from skel(ν) and contracting all arcs in the subtree that contains ν.

Decomposition Trees and Planar Embeddings. Each skeleton of a decomposition tree of G is a minor of G. So if G is planar, each skeleton of a decomposition tree $\mathcal{T}$ of G is planar as well. In fact, decomposition trees can be used to decompose not only a graph, but also an embedding of it. Consider a biconnected graph G together with a planar embedding $\mathcal{E}$ and a decomposition tree $\mathcal{T}$. For each node μ of $\mathcal{T}$ the embedding of skel(μ) is obtained from $\mathcal{E}$ by contracting for each virtual edge e of skel(μ) the expansion graph $G(e)$ into a single edge. These embeddings of the skeletons of the nodes of $\mathcal{T}$ are referred to as a *configuration*. Conversely, if (μ, μ) is an arc of $\mathcal{T}$, and skel(μ) and skel(μ) have fixed planar embeddings $\mathcal{E}_\mu$ and $\mathcal{E}_\nu$, respectively, then the skeleton of the node λ obtained from contracting (μ, ν) can be equipped with an embedding $\mathcal{E}$ by merging these embeddings along the twin edges corresponding to (μ, ν). If we equip every skeleton with a planar embedding and contract all arcs, we obtain a planar embedding of G. The resulting embedding is independent of the order of the edge contractions. Thus, every decomposition

tree $\mathcal{T}$ of G represents (not necessarily all) planar embeddings of G by choosing a planar embedding of each skeleton and contracting all arcs. We call this embedding representation the *skeleton-based* embedding representation.

SPQR-Trees. As described in the previous paragraph, decomposition trees separate independent choices in finding planar embeddings of a graph. We may either choose an embedding of the entire graph, which is generally very complex, or we may decompose the graph into smaller skeletons, independently choose embeddings of these skeletons and compose them into an embedding of the entire graph. In this sense decomposition trees implement a trade off between making few complex choices or many simple choices.

The *SPQR-tree* is a decomposition tree that makes this trade off in favor of many simple choices. SPQR-trees have four kinds of nodes, all of whose skeletons offer only few and well-structured embedding choices. (i) R-nodes are nodes whose skeleton is triconnected. Such skeletons have a unique planar embedding up to flipping. (ii) S-nodes are nodes whose skeleton is a simple cycle. Such skeletons offer no embedding choice (not counting the choice of the outer face, which is typically made by the rooting of the SPQR-tree). Adjacent S-nodes are contracted into one larger S-node, i.e., an S-node whose skeleton is a larger simple cycle. This means that in SPQR-trees no two S-nodes are adjacent. (iii) P-nodes are nodes whose skeleton is a multigraph that consists of two vertices connected by three or more edges. The order of these edges may be arbitrarily permuted. Again, adjacent P-nodes are contracted into one larger P-node, i.e., no two P-nodes are adjacent. (iv) Q-nodes are nodes whose skeleton consists of two vertices connected by two edges, namely one virtual edge and one non-virtual edge. They serve mostly to ensure that the skeletons of all non-Q-nodes consist entirely of virtual edges. In particular, they offer no embedding choice. See Figure 8.6 (a) and (b) for a graph and its SPQR-tree decomposition.

Let e_{ref} be an edge of G. Rooting $\mathcal{T}$ at the unique node μ_{ref} whose skeleton contains the real edge e_{ref} identifies a unique parent virtual edge in each of the remaining nodes; all other virtual edges are called *child virtual edges*. The arcs of $\mathcal{T}$ become directed from the parent node to the child node. Let μ be a node of $\mathcal{T}$ and let e be a child virtual edge in $\text{skel}(\mu)$ with $\text{corr}_\mu(e) = \nu$. Then the expansion graph $G(\mu, \nu)$ is simply referred to as $G(\nu)$. For rooted SPQR-trees there is also an *arc-based* embedding representation. Here the embedding choices are (i) the linear order of the children in each P-node, and (ii) for each arc (λ, μ) whose target μ is an R-node whether the embedding of the expansion graph $G(\mu)$ should be *flipped*. To obtain the embedding of G, we contract the edges of $\mathcal{T}$ from the bottom up. Consider the contraction of an arc (λ, μ) whose child μ used to be an R-node in $\mathcal{T}$. At this point, $\text{skel}(\mu)$ is equipped with a planar embedding $\mathcal{E}_\mu$. If the embedding should be flipped, we reflect

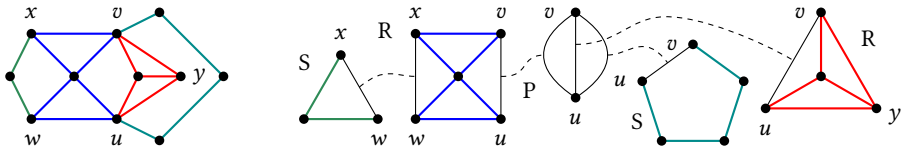


Figure 2.3: An embedded planar graph (left) and its SPQR-tree (right). The five nodes of the SPQR-tree are represented by their respective skeleton graphs. Dashed edges connect twin virtual edges and colored edges correspond to Q-nodes (omitted).

the embedding $\mathcal{E}_\mu$ before contracting (λ, μ), otherwise we simply contract (λ, μ). The arc-based and the skeleton-based embedding representations are equivalent. See Fig. 2.3 and Fig. 7.6 (a,b) for examples of a planar graph and its SPQR-tree.

PC-Trees and PQ-Trees. Roughly following the description due to Hsu and McConnell [HM03], a *PC-tree* T is a rooted tree of nodes connected by arcs directed from the parent to the children. Nodes of T that have no child are *leaves* of T. All other nodes are *inner nodes* of T. Each inner node is explicitly labeled as either a *P-node* or a *C-node*. The node that has no parent is the *root* of T. PC-trees are *ordered* trees, i.e., the arcs incident to each node are fixed in a counter-clockwise cyclic order. For every non-root node μ splitting the counter-clockwise cyclic order of arcs around μ at the parent arc yields a linear left-to-right order of the child arcs of μ. This also makes it possible to define a *leftmost* and *rightmost* leaf of the subtree of T rooted at μ. A counter-clockwise traversal of the leaves of T gives a counter-clockwise cyclic order of the leaves called the *frontier* of T.

There are two equivalence transformations for PC-trees. Namely, the order of arcs around P-nodes may be arbitrarily permuted and the order of arcs around C-nodes may be reversed. A PC-tree S obtained from T by a sequence of such equivalence transformations is *equivalent* to T, we write $S \equiv T$. The PC-tree T *represents* the set of cyclic orders of its leaves $\{\pi \mid \pi \text{ is the frontier of some } S \equiv T\}$.

The usefulness of a PC-tree arises from the update operation. Let U denote a subset of the leaves of T. The update operation yields a new PC-tree T' that represents exactly those cyclic orders of its leaves that are represented by T and in which the leaves in U appear consecutively. If there are no such orders, then the result is *null tree*, which formally represents the empty set of cyclic orders of a ground set.

Updating works in three steps. First, mark each leaf in U as black and then iteratively mark each inner node of T as black if all of its neighbors except for one are black. Mark all other nodes of T as white. Call an arc of T *terminal* if both subtrees induced by its removal from T contain both black and white leaves. If the terminal

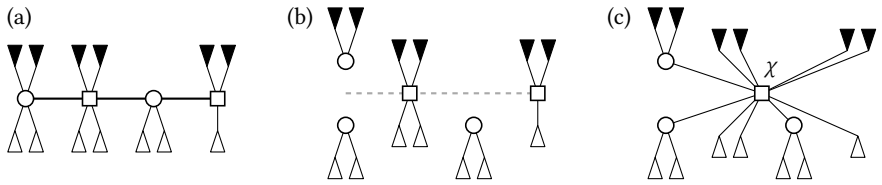


Figure 2.4: The three-step update procedure for PC-trees; C-nodes are represented by squares, P-nodes by disks. (a) The terminal path is bold, the subtrees consisting of only black and of only white vertices are represented by black and white triangles, respectively. (b) The terminal path is split. (c) A new C-node χ is connected to all vertices of the terminal path.

arcs form a path in T, call it the *terminal path.* Otherwise, T represents no cyclic order in which $u_1, \ldots, u_n$ appear consecutively, so define the result of the update as the null tree. Arrange T so that the terminal path is horizontal, and, if necessary, perform equivalence transformations on nodes of T such that all black neighbors of nodes on the terminal path lie below the terminal path and all white neighbors lie above; see Figure 2.4 (a). If this is not possible, then T represents no cyclic order in which $u_1, \ldots, u_n$ appear consecutively, again define the result as the null tree. In the second step, the terminal path is split; see Figure 2.4 (b). For every P-node μ on the terminal path that has both black and white neighbors, create two split P-nodes, one black split node and one white split node. Otherwise, create only the split node of the color of the neighbors. Attach all black (white) neighbors of the original node to the black (white) split node. In the third step, create a new C-node χ and attach all split nodes and all neighbors of C-nodes on the original terminal path to χ in the order as they appear around the terminal path; see Figure 2.4 (c). If the root of T lies on the terminal path, then χ becomes the new root. If the root of T does not lie on the terminal path, the root remains unchanged by the update.

Lemma 1 ([HM03, Lemma 4.7])**.** *Let T be a PC-tree and let U be a subset of its leaves. Updating T with U takes $O(p + |U|)$ time, where p is the length of the terminal path in T.*

PC-trees were initially developed by Shih and Hsu [SH99] and later simplified by Hsu and McConnell [HM03]. They generalize the closely related PQ-tree data structure due to Booth and Lueker [BL76]. In contrast to PC-trees, which represent sets of cyclic orders, the PQ-tree represents sets of linear orders. To implement the analogue of the update procedure for PQ-trees, an unwieldy collection of so-called patterns is used, each of which describes a local modification in the PQ-tree. In comparison, the update procedure due to Hsu and McConnell described above is a

lot more simple. Moreover, the running time of algorithms involving the PC-tree is sometimes easier to analyze than that of algorithms using PQ-trees. Fortunately, by inserting a special leaf into a PC-tree with the convention that all cyclic orders are split at that leaf, a PC-tree can easily simulate a PQ-tree. In this way, PC-trees can be used instead of PQ-trees in existing algorithms.

Part I

Level Planarity Testing

3 Radial Level Planarity with Fixed Embedding

We study level planarity testing of graphs with a fixed combinatorial embedding for three different notions of combinatorial embeddings, namely the *level embedding*, the *upward embedding* and the *planar embedding*. These notions allow for increasing degrees of freedom in their corresponding drawings.

For the fixed level embedding there are known and easy to test level planarity criteria. We use these criteria to prove an "untangling" lemma that plays a key role in a simple level planarity test for the case where only the upward embedding is fixed. This test is then adapted to the case where only the planar embedding is fixed. Further, we characterize radial upward planar embeddings, which lets us extend our results to radial level planarity. No algorithms were previously known for these problems.

This chapter is based on joint work with Ignaz Rutter [BR21].

3.1 Introduction

Level planarity and upward planarity are two natural planarity notions for directed graphs that enrich the notion of planarity of ordinary graphs by imposing additional requirements based on the directions of the edges. Informally, a drawing of a directed graph is upward when all edges are drawn as y-monotone curves. In the level planar setting the y-coordinates of the vertices are fixed. We refer to Section 3.2 for formal definitions. Both exist also in a radial form.

There is a natural definition of *topological equivalence* for such drawings. Namely, two planar drawings Γ_1, Γ_2 of a graph G are *equivalent* if they can be continuously transformed into each other without causing crossings. Formally, there is an (ambient) isotopy $F\colon \mathbb{R}^2 \times [0,1] \to \mathbb{R}^2$ that takes Γ_1 to Γ_2. That is, F is continuous, for each $t \in [0,1]$ the map $F_t(x) = F(x,t)$ is a homeomorphism of the plane, F_0 is the identity map and F_1 maps Γ_1 to Γ_2. Note that the property of F_t being a homeomorphism ensures that also each intermediate image is planar. In particular, there exists a family of drawings $F_t(\Gamma_1)$ for $t \in [0,1]$. For upward planar drawings it is natural to require that the intermediate drawings are also upward, and for level-planar drawings it is natural to require that the intermediate drawings are level planar. We consider connected graphs. Then, equivalence classes of such drawings can be combinatorially described by so-called *embeddings* (which give the circular order of edges around each vertex), *upward embeddings* (which give the linear orders of incoming and outgoing edges around each vertex), and *level embeddings* (specifying for each level the order of its vertices and the edges that cross it), respectively. These notions equally make sense in the radial setting, except that there level embeddings specify circular orderings rather than linear ones.

With this definition, not every embedding (upward embedding, level embedding) must correspond to a planar drawing (upward planar drawing, level planar drawing). If such a drawing exists, we call it *planar embedding* (*upward planar embedding*, *level planar embedding*). Yet a level-planar drawing is also upward-planar (forget about the levels), and an upward planar drawing is planar (forget about the edge directions). Thus, a level planar embedding uniquely fixes an upward planar embedding, which in turn uniquely fixes a planar embedding.

Given a fixed combinatorial embedding of one of the above types, it therefore makes sense to ask for which planarity notions there exists a planar drawing that induces the given embedding. In this way, each combination of planarity variant and combinatorial embedding type gives an instantiation of the planarity testing with fixed embedding problem.

Contribution and Outline. After introducing some preliminaries in Section 3.2, we present a characterization of radial upward planar embeddings in Section 3.3. This characterization can be tested in $O(n \log^3 n)$ time. We consider the level planar setting in Section 3.4, finding that in this setting we can "untangle" drawings to insert additional edges. This lets us devise linear-time algorithms for level planarity testing with fixed embedding. We note that for the non-radial level planar case there already exists a quadratic-time algorithm [ADDF17] and a paper that claims a linear-time algorithm [CD95], although that paper does not contain all proofs and there exists no full version. We close with some concluding remarks in Section 3.5. The following

table gives an overview of known results and our contribution. Green cells indicate new results.

	type of fixed embedding			
planarity notion	none	embedding	upward	level
planar	$O(n)$ [HT74]	$O(n)$ (Euler)	n.a.	n.a.
upward	NPC [GT01]	$O(n^{3/2})$ [Gab83, BDLM94]	$O(n)$ [BDLM94]	n.a.
radial upward	NPC [HRK97]	$O(n \log^3 n)$ Theorem 1	$O(n)$ Lemma 4	n.a.
level	$O(n)$ [JLM98]	$O(n)$ Theorem 3	$O(n)$ Theorem 2	$O(n)$ Lemma 5
radial level	$O(n)$ [BBF05]	$O(n)$ Theorem 3	$O(n)$ Theorem 2	$O(n)$ Lemma 5

3.2 Preliminaries

Bertolazzi et al. [BDLM94] show the following characterization of upward planar embeddings of biconnected graphs. Our definition of source/sink assignments follows the generalization of that characterization to non-biconnected graphs [BC20, Lemma 1].

Lemma 2 ([BDLM94],[BC20, Lemma 1])**.** *Let G be a directed acyclic graph. An upward embedding of G is upward planar if and only if there exists a face h (namely, the outer face) such that the source/sink assignment ψ assigns to each (inner) face $f \neq h$ exactly $n_f - 1$ sources/sinks, and to h exactly $n_h + 1$ sources/sinks.*

When the planar embedding of G is fixed, Bertolazzi et al. provide a quadratic-time algorithm to test whether a source/sink assignment subject to the conditions stated in Lemma 2 exists and if so, compute one. This running time can be improved to $O(n^{3/2})$ as follows. An upward planar source/sink assignment corresponds to a capacitated matching in the *assignment network* N, which is defined as follows; see Figure 3.1 (b). For each face f of $\mathcal{E}$ the assignment network N contains three vertices f, f_s, f_t and the edge $\{f_s, f_t\}$. The capacity of f is $n_f - 1$, and the capacity of f_s and f_t is 1. Each source/sink of G is also a vertex in N. For each incidence of a source (a sink) v to a face f the assignment network N contains an edge $\{v, f\}$ and an edge $\{v, f_s\}$ (and an edge $\{v, f_t\}$). The capacity of v is 1. An upward planar source/sink assignment exists if and only if the number of sources and sinks in G is $2 + \sum_{f \in \mathcal{E}}(n_f - 1)$ and there exists a subset M of the edges of N such that for each

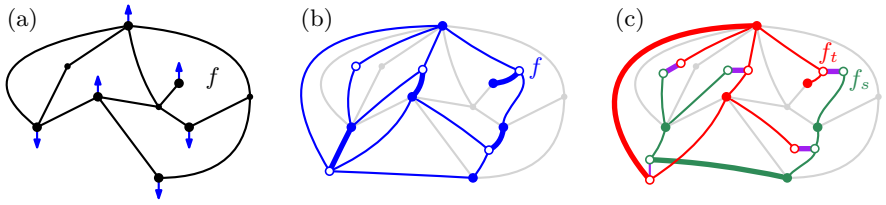


Figure 3.1: An upward planar graph G (black) together with the source/sink assignment that corresponds to its upward planar embedding (blue) (a), and the assignment network N (colored) for G (b,c). For better readability, N is shown split over (b) and (c). Bold edges belong to the capacitated matching that corresponds to the upward planar source/sink assignment in (a).

vertex v of N the number of edges in M incident to v equals the capacity of v. In particular, we interpret an edge $\{v, f\}, \{v, f_s\}, \{v, f_t\} \in M$ as v being assigned to f. The edge $\{f_s, f_t\}$ of N guarantees that either $n_f - 1$ or $n_f + 1$ sources/sinks are assigned to f. Such an edge set M can be found in $O(n^{3/2})$ running time using an algorithm due to Gabow [Gab83].

The following characterization of radial level planarity is common knowledge.

Lemma 3. *A radial level embedding of a level graph G in proper form is radial level planar if for all choices of three independent edges $(u, u'), (v, v'), (w, w')$ of G with $\ell(u) = \ell(v) = \ell(w)$ where u, v, w appear in that cyclic order the vertices u', v', w' appear in that cyclic order.*

Radial level planarity generalizes level planarity: to test a graph G for level planarity one can test G together with two vertices s, t such that s lies below all vertices of G and t lies above all vertices of G connected by the edge (s, t) for radial level planarity. Using Lemma 3 it can easily be tested in cubic time whether a (radial) level embedding is (radial) level planar. In Section 3.4, we improve this to linear time, see Lemma 5.

3.3 Radial Upward Planarity

In this section we generalize the criterion for upward planarity due to Bertolazzi et al. as stated in Lemma 2 to radial upward planarity.

Lemma 4. *Let G be a connected directed acyclic graph. An upward embedding of G is radial upward planar if and only if there exist two faces g, h such that the source/sink assignment ψ assigns to each face f exactly $n_f - 1$ sources/sinks, plus one source if $f = g$, and plus one sink if $f = h$.*

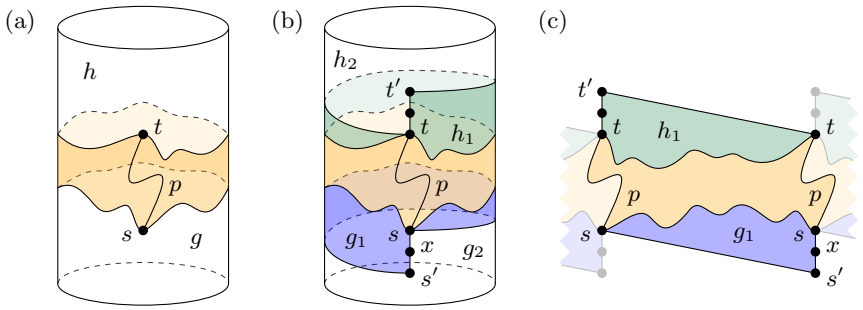


Figure 3.2: Proof of Lemma 4. Extend the radial upward drawing Γ (a) to a radial upward drawing Γ' (b), then unwrap it from the cylinder into the plane by cutting along the path p (c).

Proof. For the forward direction, consider a radial upward planar drawing Γ of G. If Γ is upward planar, there exists a source/sink assignment ψ that assigns to one face h exactly $n_h + 1$ sources/sinks, and to each other face f exactly $n_f - 1$ sources/sinks. Let $g = h$. There are exactly $n_g = n_h$ face sources and face sinks each on the facial walk that bounds $g = h$. So, at least one source of G is assigned to g, at least one sink of G is assigned to h and $n_h - 1 = n_g - 1$ further sources/sinks are assigned to $g = h$. These are exactly the claimed properties of ψ. Otherwise Γ is not upward planar. Let g denote the lower unbounded face and let $h \neq g$ denote the upper unbounded face. Let s denote the vertex with smallest y-coordinate in Γ (we may assume without loss of generality that this vertex is unique); see Figure 3.2 (a). Extend G and Γ to a new graph G' together with a radial upward planar drawing Γ' as follows; see Figure 3.2 (b). Create new vertices s', x and draw x below s and s' below x Next, create new edges (s', x) and (x, s) drawn as vertical straight-line segments. Finally, create a new edge (s', s) and draw it so that the linear order of parent edges around s' is (s', s), (s', x) and the linear order of child edges around s is (x, s), (s', s). This splits face g into one bounded face g_1 and one unbounded face g_2. Similarly, the upper unbounded face h can be split into one bounded face h_1 and one unbounded face h_2. Because G is connected there exists in G a simple path from s to t. Then in G' there exists a simple path from s' to t' that extends p. Cutting Γ' along p' makes it possible to unwrap Γ' from the cylinder and embed it into the plane; see Figure 3.2 (c). Every inner face of Γ' (in particular, g_1 and h_1) is then drawn as an inner non-radial upward planar face. Bertolazzi et al. showed that then $n_f - 1$ sources/sinks are assigned to such a face f. Now go back from Γ' to Γ. For each inner face f of Γ nothing changes, i.e., $n_f - 1$ sources/sinks are assigned to f. Assign s and every source/sink previously assigned to g_1 to g. Observe $n_g = n_{g_1}$, so $n_g - 1$ sources/sinks are assigned to g, plus

the source s. Likewise, assign t and every source/sink previously assigned to h_1 to h. Then $n_h - 1$ sources/sinks are assigned to h, plus the sink t. These are exactly the claimed properties of the source/sink assignment.

For the reverse direction, consider a source/sink assignment ψ with the stated properties. If the extra source and sink are assigned to the same face of $\mathcal{E}$, then by Lemma 2 there exists an upward planar drawing of G. Otherwise, let g denote the face to which an extra source s is assigned, and let $h \neq g$ denote the face to which an extra sink t is assigned. Extend G and $\mathcal{E}$ to a new graph G' with a planar embedding $\mathcal{E}'$, respectively, as follows. Create new vertices including s', t' and edges as described for the forward direction. In particular, insert the edges (x, s), (s', s) around s between the two edges to which s is assigned by ψ. Recall that this splits face g into one bounded face g_1 and one unbounded face g_2. Assign s' to g_2, then $n_g - 1 = n_{g_1} - 1$ sources/sinks are assigned to g_1. Similarly, face h can be split into one bounded face h_1 and one unbounded face h_2. Assign t' to h_2, then $n_h - 1 = n_{h_1} - 1$ sources/sinks are assigned to h_1. Observe that each bounded face f of G' is assigned exactly $n_f - 1$ sources/sinks. Using the result of Bertolazzi et al., an incoming (outgoing) edge can be added to each source (sink) assigned to f and embedded within f (see the if-part of the proof of Theorem 3 in [BDLM94]). After this, G' is a directed acyclic graph whose single source is s' and whose single sink is t' together with a planar embedding $\mathcal{E}'$. Hashemi shows that G' is then upward planar with planar embedding $\mathcal{E}'$ on the sphere [Has01, Theorem 1]. Auer et al. show that upward planarity on the sphere coincides with upward planarity on the standing cylinder [ABBG11, Theorem 1]. Hence, G' is radial upward planar with planar embedding $\mathcal{E}'$, and, as a direct consequence, G is radial upward planar with planar embedding $\mathcal{E}$. □

In particular, the face to which the extra source (sink) is assigned is the lower (upper) unbounded face. The (non-radial) upward planarity criterion due to Bertolazzi et al. states the special case that $g = h$. A given upward embedding of G can easily be tested for radial upward planarity in linear time using the criterion provided by Lemma 4. Also observe that in a given radial upward planar embedding, the extra source/sink can be reassigned to any incident face while maintaining radial upward planarity. In the non-radial setting, the extra source and sink can only be reassigned to faces that are incident to both of these vertices while maintaining upward planarity.

Theorem 1. *Let G be a directed graph together with a planar embedding $\mathcal{E}$. It can be tested in $O(n \log^3 n)$ time whether G admits a radial upward planar embedding with planar embedding $\mathcal{E}$.*

Proof. The criterion from Lemma 4 can be tested by solving a maximum flow problem on the following network N. The sources of N are the sources/sinks of G and the sinks of N are the faces of $\mathcal{E}$ together with two vertices s (for the extra source) and t

(for the extra sink). For each source (sink) v on the facial walk that bounds f, create one arc from v to f and one arc from v to s (one arc from v to t). Set the capacity of each source of N to one. Set the demand of f to $n_f - 1$ and of s and t to one. Note that N is not necessarily planar, but removing s and t makes it planar. This means that we can compute a maximum flow in N using the $O(n \log^3 n)$-time algorithm due to Borradaile et al. [Bor+17, Corollary 5.1]. □

Note that testing a graph for radial upward planarity with a given planar embedding is faster compared to testing a graph for (non-radial) upward planarity with a given planar embedding due to the independent treatment of the extra source and extra sink.

3.4 Level Planarity with Fixed Embedding

In this section, we investigate (radial) level planarity testing with fixed embedding. We begin with the most restricted embedding notion, which is that of (radial) level embeddings. In particular, we show that the problem of deciding whether a given radial level embedding is radial level planar can be solved in linear time using the characterization provided by Lemma 3. Recall from Section 3.2 that this generalizes the treatment of non-radial level embeddings.

Lemma 5. *It can be tested in linear time whether a (radial) level embedding of a level graph in proper from is (radial) level planar.*

Proof. Let G be a k-level graph and let $\mathcal{E}$ be a radial level embedding of G. To check whether $\mathcal{E}$ is level planar it suffices to check for each $1 \leq i < k$ whether the edges between levels i and $i+1$ induce no crossings. To this end, proceed as follows. First, remove all vertices on level i (on level $i+1$) that have no neighbor on level $i+1$ (on level i). If at most one vertex remains on level i or $i+1$, then $\mathcal{E}$ is radial level planar. Otherwise, let u denote a vertex on level i. If the neighbors of u on level $i+1$ do not appear consecutively in the counter-clockwise cyclic order of vertices level $i+1$, then $\mathcal{E}$ is not radial level planar. This can be checked in linear time for each vertex u on level i. If this check passes, the counter-clockwise cyclic order induces a left-to-right linear order of the neighbors $u'_1, u'_2, \ldots, u'_n$ of u on level $i+1$. If any u'_j for $1 < j < n$ has a neighbor other than u, then $\mathcal{E}$ is not radial level planar. Next, contract all neighbors of u into a single vertex u' on level $i+1$. Iteratively perform this step for each vertex on level i, and, symmetrically, for each vertex on level $i+1$. The result is that each vertex has exactly one neighbor on the other level, i.e., we are left with a set of independent edges and counter-clockwise cyclic orders of their endpoints on each level $i, i+1$. Then $\mathcal{E}$ is radial level planar if and only if these cyclic orders can

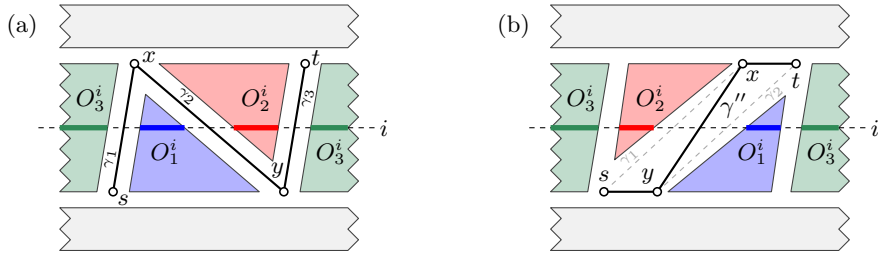


Figure 3.3: Eliminating the turns x and y in the proof of Lemma 6.

be translated into each other by renaming each vertex to its unique neighbor on the other level. This can be checked in linear time. □

Next, we consider the slightly more relaxed version where we seek to decide whether a given radial upward planar embedding is radial level planar. To this end, we provide the following lemma, which explains how to "untangle" radial level planar drawings without changing the underlying radial upward embedding. This property is partially known [Ark+04, Lemma 3.4], [ADDF17, Claim 2], but only for non-radial level planar drawings.

Lemma 6. *Let $G = (V, E)$ be a level graph, let Γ be a (radial) level planar drawing of G with (radial) upward embedding $\mathcal{E}$. Let $a, b \in V$ with $\ell(a) < \ell(b)$ be incident to a face f such that embedding (a, b) inside f yields a (radial) upward embedding $\mathcal{E}'$ of $G + (a, b)$. If there exists a curve γ inside the face f of Γ that connects a and b and does not intersect the levels $\ell(a)$ and $\ell(b)$, there exists a (radial) level planar drawing Γ' of $G + (a, b)$ with (radial) upward embedding $\mathcal{E}'$.*

Proof. Let γ be such a curve. Insert between any pair of consecutive levels so-called *half-levels*, and subdivide γ such that it becomes a path that crosses all levels and half-levels at vertices. A vertex of γ is a *turn* if its two neighbors lie on the same level. We can assume that all turns lie on half-levels. Note that such a curve has an even number of turns.

If γ has no turns, it corresponds to an upward drawing of (a, b) and we can choose Γ as Γ'. Assume that γ has at least one turn. Consider two consecutive turns $x, y, \ell(x) > \ell(y)$ that minimize $\ell(x) - \ell(y)$. We modify the drawing to eliminate these two turns. Let s be the last vertex before x with $\ell(s) = \ell(y)$ and let t be the first vertex after y with $\ell(t) = \ell(x)$ on γ; see Figure 3.3 (a). Each level in $[\ell(y)+1, \ell(x)-1]$ is intersected exactly once by each curve $\gamma_1 := \gamma[s, x], \gamma_2 := \gamma[x, y], \gamma_3 := \gamma[y, t]$, where $\gamma[a, b]$ denotes the subcurve of γ from a to b (inclusive). Correspondingly, for each such level i, the counterclockwise circular ordering of the vertices on that

level are partitioned into three linear orders O^i_j, $j = 1, 2, 3$ such that O^i_j is the linear order between γ_j and γ_{j+1} (where $\gamma_4 = \gamma_1$ for reasons of circularity), and the circular ordering on level i is the concatenation $O^i_1 \cdot O^i_2 \cdot O^i_3$. Consider the drawing Γ'' of G and γ that is obtained by changing the vertex ordering on each level $i \in [\ell(y) + 1, \ell(x) - 1]$ to the concatenation $O^i_2 \cdot O^i_1 \cdot O^i_3$. Let further γ'' be obtained from γ by replacing the subcurve $\gamma[s, t]$ by inserting a straight-line segment from s to y, then following γ in reverse from y to x and then inserting a straight-line segment from x to t; see Figure 3.3 (b).

We show that Γ'' is radial level planar using Lemma 3. Crossings could only be introduced by edges that cross level $\ell(s) = \ell(y)$ or level $\ell(x) = \ell(t)$. The order $O_2^{\ell(s)=\ell(y)}$ is empty, so $O_1^{\ell(s)} \cdot O_2^{\ell(s)} \cdot O_3^{\ell(s)} = O_2^{\ell(s)} \cdot O_3^{\ell(s)} = O_2^{\ell(s)} \cdot O_1^{\ell(s)} \cdot O_3^{\ell(s)}$. Likewise, the order $O_1^{\ell(x)=\ell(t)}$ is empty, so $O_1^{\ell(t)} \cdot O_2^{\ell(t)} \cdot O_3^{\ell(t)} = O_2^{\ell(t)} \cdot O_3^{\ell(t)} = O_2^{\ell(t)} \cdot O_1^{\ell(t)} \cdot O_3^{\ell(t)}$. Therefore Γ'' is radial level planar. A very similar argument shows that the upward embeddings of Γ and Γ'' are the same. Moreover γ'' has fewer turning points than γ. Therefore the proof follows by induction. □

We can use Lemma 6 to insert edges into a graph while maintaining (radial) upward planarity such that all faces end up having size at most 4.

Lemma 7. *Let $G = (V, E)$ be a level graph together with a (radial) upward planar embedding $\mathcal{E}$. We can compute in linear time a level graph G' with (radial) upward planar embedding $\mathcal{E}'$ whose faces have size at most 4 so that G has a (radial) level planar drawing with embedding $\mathcal{E}$ if and only if G' has a (radial) level planar drawing with embedding $\mathcal{E}'$.*

Proof. Let G be a level graph together with an upward planar embedding $\mathcal{E}$. Recall from Section 3.2 that we may assume without loss of generality that G is in star form. Insert edges into G and $\mathcal{E}$ until all faces have size at most 4 while maintaining (radial) level planarity as follows. Let f be a face of $\mathcal{E}$ with at least five vertices incident to f. Suppose that there exist successive edges (u, v), (v, w) or (v, w), (u, v) on the facial walk that bounds f. If G is level planar with upward planar embedding $\mathcal{E}$, then by Lemma 6 the edge (u, w) can be inserted into G and $\mathcal{E}$ while maintaining level planarity (other instances of the edge (u, w) may already exist in G, but this is not a problem). In particular, the curve γ can be constructed by closely following (u, v), (v, w) or (v, w), (u, v) within f. If no such successive edges exist, then each vertex incident to f is a face source or a face sink of f. Let (w, v) be an edge incident to f that minimizes $\ell(v) - \ell(w)$. Because v and w are face sources or face sinks the edge (u, v) and (w, x) precede and succeed (w, v) on the facial walk of f, respectively. Because (w, v) minimizes $\ell(v) - \ell(w)$ and because G is in star form, i.e., every vertex lies on its own level, it is $\ell(u) \leq \ell(w) < \ell(v) \leq \ell(x)$. See Figure 3.4. First, suppose

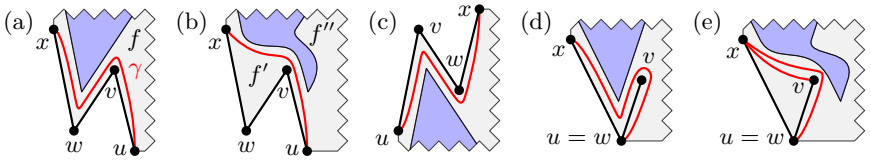


Figure 3.4: Inserting the edge (u, x) by defining the red curve γ (a) and using Lemma 6 to move the blue obstruction to the side (b). The upward embedding may be different (c), which requires moving a different obstruction. If w is a cutvertex that separates v from the rest of G, insert the edge (v, x) instead (e, d).

that $u \neq w$ and $v \neq x$. Then, because G is in star form, both inequalities are strict, i.e., $\ell(u) < \ell(w) < \ell(v) < \ell(x)$. There is a curve γ from u to x that does not intersect levels $\ell(u), \ell(x)$ except in u and x. By Lemma 6 (u, x) can be inserted into G and $\mathcal{E}$ while maintaining level planarity. In particular, (u, x) is inserted just after (u, v) in the left-to-right order of outgoing edges incident to u and just after (w, x) in the left-to-right order of incoming edges incident to x. See Figure 3.4 (a–c). Now consider the case $u = w$, i.e., w is a cutvertex that separates v from the rest of G. Again, there is a curve from u to x that does not intersect levels $\ell(u), \ell(x)$ except in u and x. By Lemma 6 (u, x) could be inserted into G and $\mathcal{E}$ while maintaining level planarity (though we do not actually insert this edge). Then the edge (v, x) can also be inserted into G and $\mathcal{E}$ while maintaining level planarity. Insert (v, x) just after (w, x) in the left-to-right order of incoming edges incident to x. See Figure 3.4 (d, e). The case $v = x$ is symmetric, in this case we insert the edge (u, w). Finally, observe that it is $u \neq w$ or $v \neq x$ because at least five vertices are incident to f.

To reach the claimed running time, we proceed slightly differently. Process the graph one face at a time. For each face f, first shortcut successive edges (u, v), (v, w) or (v, w), (u, v). This can be done in time linear in the size of f. Afterwards, every vertex incident to f is a face source or a face sink. Consider a sequence u, v, w of vertices that are consecutive on the facial walk that bounds f such that u, w are face sources and v is a face sink. Because G is in star form all vertices have distinct levels. If $\ell(u) \leq \ell(w)$, we process the face clockwise, otherwise, we go counterclockwise. We describe the clockwise case, the counterclockwise case is symmetric. Create a stack S, and push the edges (u, v) and (w, v) onto S. We will maintain the invariant that the stack contains a part of the facial walk of the current face such that, if edge (a, b) lies above edge (c, d) on S, then $\ell(a) \geq \ell(c)$ and $\ell(b) \leq \ell(d)$; see Figure 3.5. If the number of elements on S drops below two, we clear the stack and start the construction with a new sequence u, v, w of consecutive vertices.

While there are at least two elements on the stack, we proceed as follows. Let u, v, w

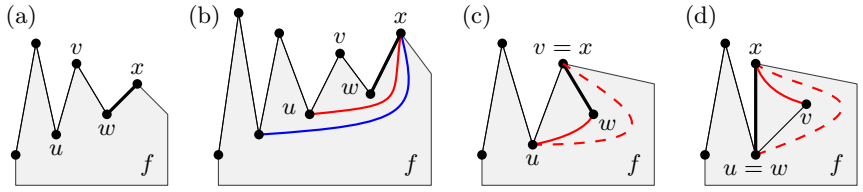


Figure 3.5: Illustration of the processing algorithm in clockwise direction. The edges on the stack S are thin. The edge following the top edge on S is bold. The illustration assumes that w is a source. If $\ell(v) > \ell(x)$ (a), then (w, x) is pushed onto S. If $\ell(v) < \ell(x)$ (b), the red edge (u, x) is inserted. In the depicted case, the following iteration inserts the blue edge. If $v = x$ (c), the edge (u, w) is inserted instead of (u, x). If $u = w$ (d), the edge (v, x) is inserted instead of (u, x).

denote the endpoints of the two edges on top of the stack in the order in which they occur along the clockwise facial walk. Let x denote the successor of w. We distinguish cases based on whether w is a source or a sink. First assume that w is a source (the case where w is a sink is symmetric). If $\ell(x) < \ell(v)$, we push (w, x) onto the stack, which maintains the invariant; see Figure 3.5 (a). If $\ell(x) = \ell(v)$, then $v = x$ and $u \neq w$, i.e., $\ell(u) < \ell(w)$, and we insert (u, w) into the face; see Figure 3.5 (c). If $\ell(x) > \ell(v)$, then either $u = w$ and we insert (v, x) into the face, or $\ell(u) < \ell(w)$, and Lemma 6 allows us to insert (u, x) into the face. See Figure 3.5 (d) and (b), respectively.

If we insert the edge (u, x), remove the top two edges from the stack. Otherwise, if (u, w) or (v, x) is inserted, remove the top edge from the stack. Then, we continue with the next iteration. The removed edges are then incident to faces of size 4 or 3, respectively. This means that every edge incident to f is pushed onto S at most once. Moreover, since the edges on S become shorter and shorter, the algorithm terminates. It follows that the face can be augmented in time that is linear in its size. □

Note that we may insert edges that already existed in G, turning the initially simple graph into a multigraph. To apply the upward planarity criterion due to Bertolazzi et al., which is only formulated for simple graphs, we can subdivide edges that occur multiple times. The subdivision vertices are neither (face) sources nor (face) sinks, so the upward planar embeddings of this subdivided graph correspond bijectively to the upward planar embeddings of our multigraph.

Theorem 2. *Let $G = (V, E)$ be a level graph together with a (radial) upward planar embedding $\mathcal{E}$. It can be tested in linear time whether G admits a (radial) level planar drawing with (radial) upward planar embedding $\mathcal{E}$.*

Proof. Lemma 7 lets us assume that each face has size at most 4. No source/sink is

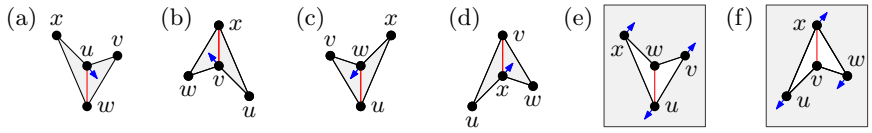


Figure 3.6: Inserting the red augmentation edges into inner faces (a–d), and the level planarity criterion for the outer face (e, f).

assigned to faces of size at most 3, then nothing needs to be shown. Let f be an inner face of size 4 and let u, v, w, x denote the facial walk that bounds f. Without loss of generality let u, w be face sources and let v, x be face sinks. See Figure 3.6 (a–d). Exactly one of u, v, w, x is assigned to f. Bertolazzi et al. [BDLM94] show that if u is assigned to f, then the edge (w, u) can be inserted into f in any upward planar drawing. For level planarity, this additionally requires $\ell(w) < \ell(u)$; see Figure 3.6 (a). Thus, if $\ell(w) \geq \ell(u)$, then G is not (radial) level planar with (radial) upward embedding $\mathcal{E}$. The cases when v, w or x are assigned to f are symmetric; see Figure 3.6 (b–d). If the outer face h has size 4, either two sources and one sink is assigned to h, or one source and two sinks are assigned to h. Level planarity requires that the lowest vertex u and the highest vertex x are assigned to the outer face. This leaves two cases; see Figure 3.6 (e, f). If v is assigned to h, then it must be $\ell(u) < \ell(w)$, otherwise w is assigned to h and it must be $\ell(v) < \ell(x)$. Inserting all of these edges gives a level graph G' with a single source u and a single sink x together with a embedding $\mathcal{E}'$. As in the proof of 4, this means that G' is (radial) upward planar with upward embedding $\mathcal{E}'$. Because G' has a single source and a single sink this means that G' is also (radial) level planar with upward embedding $\mathcal{E}'$. To see this, consider a (radial) upward planar drawing Γ' of G' with upward planar embedding $\mathcal{E}'$. Assume without loss of generality that the y-coordinate of u in Γ' is $\ell(u)$. Scale up Γ' so that for each vertex $v \neq u$ its y-coordinate is at least $\ell(v)$. Next, process each vertex v of G' in topological order. If the y-coordinate of v is greater than $\ell(v)$, move v down along its leftmost incoming edge until its y-coordinate is $\ell(v)$ while maintaining (radial) upward planarity. Once all vertices are processed Γ' is a (radial) level planar drawing of G' with upward embedding $\mathcal{E}'$. Since G is a subgraph of G' and $\mathcal{E}$ is the restriction of $\mathcal{E}'$ to G this means that G is (radial) level planar with upward embedding $\mathcal{E}$. Reducing the faces to size 4 takes linear time (Lemma 7); the remaining steps take $O(1)$ time per face. □

Finally, consider the least restricted version of level planarity testing with fixed embedding. Namely, suppose that only a planar embedding $\mathcal{E}$ of G is fixed. We translate Lemma 7 to this setting as follows.

Lemma 8. *Let G be a level graph with a planar embedding $\mathcal{E}$. We can compute in linear time a level graph G' with planar embedding $\mathcal{E}'$ whose faces have size at most 4 so that G has a (radial) level planar drawing with embedding $\mathcal{E}$ if and only if G' has a (radial) level planar drawing with embedding $\mathcal{E}'$.*

Proof. The only difference to the proof of Lemma 7 is that we do not insert the edges into the upward embedding, i.e., into the linear lists of outgoing or incoming edges, but rather into embedding, i.e., the cyclic order of edges. □

To test a planar embedding for (radial) level planarity in linear time, we need one last ingredient. Recall that (radial) level planarity requires (radial) upward planarity, which can be tested using the assignment network. We show that for a certain kind of assignment network, testing (radial) upward planarity is feasible in linear time.

Lemma 9. *Let $G = (A \cup B, E)$ be a bipartite graph with n vertices and m edges such that every vertex in A has degree at most 2. It can be decided $O(n + m)$ time whether G admits a perfect matching and, if so, one can be computed within the same running time.*

Proof. We proceed similar to the algorithm of Karp and Sipser [KS81]. We first preprocess the graph G in linear time by iteratively matching a degree-1 vertex $u \in A$ to its unique neighbor $v \in B$ and then removing both u and v from the graph. The resulting graph $G' = (A' \cup B', E')$ has a perfect matching if and only if G has one.

Next, we construct an auxiliary graph $H = (B', E'')$ such that E'' contains an edge v, w if there exists a $u \in A'$ that is incident to both $v, w \in B'$. A perfect matching of G' corresponds to an orientation of H such that each vertex has in-degree exactly 1. For example, interpret the orientation (v, w) as $\{u, v\} \in E'$ being matched and interpret the orientation (w, v) as $\{u, w\} \in E'$ being matched. Such an orientation of H exists if and only if each connected component of G' is a pseudo-tree, that is, it is a tree plus one edge. This can be tested in linear time. □

This lets us prove the following.

Theorem 3. *Let G be a level graph together with a planar embedding $\mathcal{E}$. It can be tested in linear time whether G admits a (radial) level planar drawing with planar embedding $\mathcal{E}$.*

Proof. Lemma 8 lets us assume that each face has size at most 4. We check whether there exists an upward planar embedding with planar embedding $\mathcal{E}$ that is level planar by computing an assignment of vertices to faces similar to Bertolazzi et al. [BDLM94]. Recall that no sources/sinks are assigned to faces of size 3, and exactly one source/sink is assigned to each inner face of size 4. From the assignment of a source or sink to an

inner face f, we know which edge must be inserted in the augmentation procedure of Bertolazzi et al., and we can test beforehand, whether this augmentation respects the given level assignment. If it does not, then we remove the possibility to assign the corresponding source/sink to face f by removing the corresponding edge from the assignment network. If the remaining network does not allow to assign all sources and sinks to the faces of the graph, then there is no level-planar embedding with the given embedding. Otherwise, adding the augmentation edges yields a level st-graph that is planar, and therefore level-planar. Note that in this case the assignment network is bipartite because it connects face nodes with vertex nodes and each face node has degree at most two. Lemma 9 shows that in such a network, an assignment can be computed in linear time. □

3.5 Conclusion

In this chapter, we have studied the (radial) upward and (radial) level planarity testing problems with fixed embedding for three notions of fixed embeddings, namely embeddings, upward embeddings and level embeddings. We provided a new combinatorial characterization of radial upward planarity with fixed embedding, which can be tested in $O(n \log^3 n)$ time. Somewhat curiously, testing for radial upward planarity is faster than testing for (non-radial) upward planarity, which takes $O(n^{3/2})$ time. It would be interesting to see whether these running times can be improved, possibly down to linear time. For the level planar setting, we have shown that a well-known characterization of radial level planar embeddings can be tested in linear time. For a fixed (upward) planar embedding, we have shown how to insert edges into the graph while maintaining (upward) planarity such that all faces have size at most 4. This lets us test a level graph with a fixed ((radial) upward) planar embedding for (radial) level planarity in linear time.

4 Level-Planarity: Transitivity vs. Even Crossings

Recently, Fulek et al. [FPS17, FPS16, FPSŠ13] have presented Hanani-Tutte results for (radial) level-planarity, i.e., a graph is (radial) level-planar if it admits a (radial) level drawing where any two independent edges cross an even number of times. We show that the 2-SAT formulation of level-planarity testing due to Randerath et al. [Ran+01] is equivalent to the strong Hanani-Tutte theorem for level-planarity [FPSŠ13]. Further, we show that this relationship carries over to radial level planarity, which yields a novel polynomial-time algorithm for testing radial level-planarity in the spirit of Randerath et al.

This chapter is based on joint work with Ignaz Rutter and Peter Stumpf [BRS18].

4.1 Introduction

Planarity of graphs is a fundamental concept for graph theory as a whole, and for graph drawing in particular. Naturally, variants of planarity tailored specifically to directed graphs have been explored. A planar drawing is *upward planar* if all edges are drawn as monotone curves in the upward direction. A special case are *level-planar drawings* of level graphs, where the input graph $G = (V, E)$ comes with a level assignment $\ell\colon V \to \{1, 2, \ldots, k\}$ for some $k \in \mathbb{N}$ that satisfies $\ell(u) < \ell(v)$ for all $(u, v) \in E$. One then asks whether there is an upward planar drawing such that each vertex v is mapped to a point on the horizontal line $y = \ell(v)$ representing the level of v. There are also radial variants of these concepts, where edges are drawn as

curves that are monotone in the outward direction in the sense that a curve and any circle centered at the origin intersect in at most one point. Radial level-planarity is derived from level-planarity by representing levels as concentric circles around the origin.

Despite the similarity, the variants with and without levels differ significantly in their complexity. Whereas testing upward planarity and radial planarity are NP-complete [GT01], level-planarity and radial level-planarity can be tested in polynomial time. In fact, linear-time algorithms are known for both problems [JLM98, BBF05]. However, both algorithms are quite complicated, and subsequent research has led to slower but simpler algorithms for these problems [HH07, Ran+01]. Recently also constrained variants of the level-planarity problem have been considered [BR17, KR19].

One of the simpler algorithms is the one by Randerath et al. [Ran+01]. It only considers *proper* level graphs, where each edge connects vertices on adjacent levels. This is not a restriction, because each level graph can be subdivided to make it proper, potentially at the cost of increasing its size by a factor of k. It is not hard to see that in this case a drawing is fully specified by the vertex ordering on each level. To represent this ordering, define a set of variables $\mathcal{V} = \{uw \mid u, w \in V, u \neq w, \ell(u) = \ell(w)\}$ where uw being true means u is left of w on level $\ell(w)$. Randerath et al. observe that there is a trivial way of specifying the existence of a level-planar drawing by the following consistency (4.1), transitivity (4.2) and planarity constraints (4.3):

$$\forall uw \in \mathcal{V} : uw \Leftrightarrow \neg wu \tag{4.1}$$

$$\forall uw, wy \in \mathcal{V} : uw \wedge wy \Rightarrow uy \tag{4.2}$$

$$\forall uw, vx \in \mathcal{V} \text{ with } (u,v), (w,x) \in E \text{ independent} : uw \Leftrightarrow vx \tag{4.3}$$

The surprising result due to Randerath et al. [Ran+01] is that the satisfiability of this system of constraints (and thus the existence of a level-planar drawing) is equivalent to the satisfiability of a *reduced constraint system* obtained by omitting the transitivity constraints (4.2). That is, transitivity is irrelevant for the satisfiability. Note that a satisfying assignment of the reduced system is not necessarily transitive. Rather Randerath et al. prove that a solution can be made transitive without invalidating the other constraints. Since the remaining conditions (4.1) and (4.3) can be easily expressed in terms of 2-SAT, which can be solved efficiently, this yields a polynomial-time algorithm for level-planarity.

A very recent trend in planarity research are Hanani-Tutte style results. The (strong) Hanani-Tutte theorem [Cho34, Tut70] states that a graph is planar if and only if it can be drawn so that any two independent edges (i.e., not sharing an endpoint) cross an even number of times. One may wonder for which other drawing styles such a statement is true. Pach and Tóth [PT04, PT11] showed that the weak

Hanani-Tutte theorem (which requires even crossings for all pairs of edges) holds for a special case of level-planarity and asked whether the result holds in general. This was shown in the affirmative by Fulek et al. [FPSŠ13], who also established the strong version for level-planarity. Most recently, both the weak and the strong Hanani-Tutte theorem have been established for radial level-planarity [FPS17, FPS16].

Contribution. We show that the result of Randerath et al. [Ran+01] from 2001 is equivalent to the strong Hanani-Tutte theorem for level-planarity.

A key difference is that Randerath et al. consider proper level graphs, whereas Fulek et al. [FPSŠ13] work with graphs with only one vertex per level. For a graph G we define two graphs $G^\star$, G^+ that are equivalent to G with respect to level-planarity. The graph $G^\star$ has only vertex per level and the graph G^+ is the proper subdivision of $G^\star$. We show how to transform a Hanani-Tutte drawing of a graph $G^\star$ into a satisfying assignment for the constraint system of G^+ and vice versa. Since this transformation does not make use of the Hanani-Tutte theorem nor of the result by Randerath et al., this establishes the equivalence of the two results.

Moreover, we show that the transformation can be adapted to the case of radial level-planarity. This results in a novel polynomial-time algorithm for testing radial level-planarity by testing satisfiability of a system of constraints that, much like the work of Randerath et al., is obtained from omitting all transitivity constraints from a constraint system that trivially models radial level-planarity. Currently, we deduce the correctness of the new algorithm from the strong Hanani-Tutte theorem for radial level planarity [FPS16]. However, also this transformation works both ways, and a new correctness proof of our algorithm in the style of the work of Randerath et al. [Ran+01] may pave the way for a simpler proof of the Hanani-Tutte theorem for radial level-planarity. We leave this as future work.

4.2 Preliminaries

For any level graph G let $\overline{G}$ denote its proper subdivision; see G^+ in Figure 4.1. Two independent edges (u, v), (w, x) are *critical* if $\ell(u) \leq \ell(x)$ and $\ell(v) \geq \ell(w)$. Note that any pair of independent edges that can cross in a level drawing of G is a pair of critical edges. Throughout this chapter, we consider drawings that may be non-planar, but we assume at all times that no two distinct vertices are drawn at the exact same point, no edge passes through a vertex, and no three (or more) edges cross in a single point. If any two independent edges cross an even number of times in a drawing Γ of G, it is called a *Hanani-Tutte drawing* of G.

We now describe how to obtain a proper level graph for the constraint system from one for Hanani-Tutte drawings and vice-versa. Note that a level graph that has

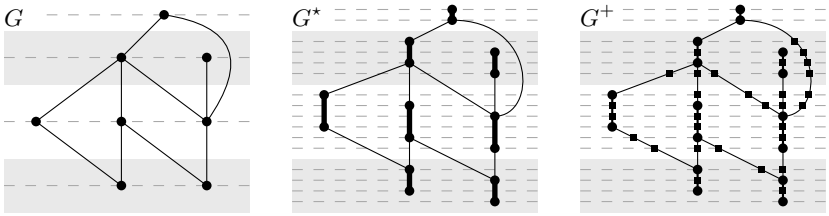


Figure 4.1: A level graph G (a) modified to a graph $G^\star$ so as to have only one vertex per level; stretch edges are bold (b) and its proper subdivision G^+ (c).

only one vertex per level can be considered as an ordered graph (where the vertices are ordered as their levels are ordered), and vice versa. For any k-level graph G we define a *narrow form* $G^\star$ that is an ordered graph. The construction is similar to the one used by Fulek et al. [FPSŠ13]. Let $n(i)$ denote the number of vertices on level i for $1 \leq i \leq k$. Further, let $v_1, v_2, \ldots, v_{n(i)}$ denote the vertices on level i. Subdivide every level i into $2n(i)$ sublevels $(i,1), (i,2), \ldots, (i,2n(i))$ and order them all lexicographically. For $1 \leq j \leq n(i)$, replace vertex v_j by two vertices v'_j, v''_j with $\ell(v'_j) = (i,j)$ and $\ell(v''_j) = (i, j+n(i))$ and connect them by an edge (v'_j, v''_j), referred to as the *stretch edge* $e(v_j)$. Connect all incoming edges of v_j to v'_j instead and connect all outgoing edges of v_j to v''_j instead. Let $e = (u,v)$ be an edge of G. Then let e' denote the edge of $G^\star$ that connects the endpoint of $e(u)$ with the starting point of $e(v)$; see Figure 4.1. Note that a Hanani-Tutte drawing of G induces a Hanani-Tutte drawing of $G^\star$. Define G^+ as the graph obtained by subdividing the edges of $G^\star$ so that the graph becomes proper; again, see Figure 4.1. For any vertex u in G we say the vertices u', u'' and all subdivision vertices of (u', u'') *originate* from u. Let $(u,v), (w,x)$ be critical edges in $G^\star$. Define their *limits* in G^+ as $(u', v'), (w', x')$ where u', v' are endpoints or subdivision vertices of (u,v), w', x' are endpoints or subdivision vertices of (w,x) and it is $\ell(u') = \ell(w') = \max(\ell(u), \ell(w))$ and $\ell(v') = \ell(x') = \min(\ell(v), \ell(x))$. Note that we have $\overline{G^+} = G^+$ where for an edge e in G we have subdivision vertices of e in G^+ where in $\overline{G^+}$ we have subdivision vertices of stretch edges in $\overline{G}$ that are subdivision vertices of e. Finally, we define the function L that maps each level (i,j) of $G^\star$ or G^+ to the level i of G.

Lemma 10. *Let G be a level graph. Then*

$$G \text{ is level-planar} \Leftrightarrow G^\star \text{ is level-planar} \Leftrightarrow G^+ \text{ is level-planar}$$

Proof. The first equivalence is due to Fulek et al. [FPSŠ13]. The forward direction is trivially true. For the reverse direction, the key insight is that for every level i of G there is a level i' of $G^\star$ so that for each vertex v with $\ell(v) = i$ its stretch edge $e(v)$

crosses level i' in $G^\star$. The second equivalence is obvious, because G^+ is the proper subdivision of $G^\star$. □

4.3 Level-Planarity

Recall from the introduction that Randerath et al. formulate level planarity of a proper level graph G as a Boolean satisfiability problem $\mathcal{S}'(G)$ on the set of variables $\mathcal{V} = \{uw \mid u \neq w, \ell(u) = \ell(w)\}$ and the clauses given by Eq. (4.1)–(4.3).

It is readily observed that G is level-planar if and only if $\mathcal{S}'(G)$ is satisfiable. Now let $\mathcal{S}(G)$ denote the SAT instance obtained by removing the transitivity clauses (4.2) from $\mathcal{S}'(G)$. Note that $(uw \Rightarrow \neg wu) \equiv (\neg uw \vee \neg wu)$ and $(uw \Rightarrow vx) \equiv (\neg uw \vee vx)$, i.e., $\mathcal{S}(G)$ is an instance of 2-SAT, which can be solved efficiently. The key claim of Randerath et al. is that $\mathcal{S}'(G)$ is satisfiable if and only if $\mathcal{S}(G)$ is satisfiable, i.e., dropping the transitivity clauses does not change the satisfiability of $\mathcal{S}'(G)$. In this section, we show that $\mathcal{S}(G)$ is satisfiable if and only if $G^\star$ has a Hanani-Tutte level drawing (Theorem 4). Of course, we do not use the equivalence of both statements to level-planarity of G. Instead, we show that $\mathcal{S}(\overline{G})$ and $\mathcal{S}(G^+)$ are equivalent (Lemma 13) and then construct a satisfying truth assignment of $\mathcal{S}(G^+)$ from a given Hanani-Tutte level drawing (Lemma 12), and vice versa (Lemma 14). This implies the equivalence of the results of Randerath et al. and Fulek et al. (Corollary 1).

The common ground for our constructions is the constraint system $\mathcal{S}'(G^+)$, where a Hanani-Tutte drawing implies a variable assignment that does not necessarily satisfy the planarity constraints (4.3), though in a controlled way, whereas a satisfying assignment of $\mathcal{S}(G)$ induces an assignment for $\mathcal{S}'(G^+)$ that satisfies the planarity constraints but not the transitivity constraints (4.2). Thus, in a sense, our transformation trades planarity for transitivity and vice versa.

A (not necessarily planar) drawing Γ of G *induces* a truth assignment φ of $\mathcal{V}$ by defining for all $uw \in \mathcal{V}$ that $\varphi(uw)$ is true if and only if u lies to the left of w in Γ. Note that this truth assignment must satisfy the consistency clauses, but does not necessarily satisfy the planarity constraints. The following lemma describes a relationship between certain truth assignments of $\mathcal{S}(G)$ and crossings in Γ that we use to prove Lemmas 12 and 14.

Lemma 11. *Let $(u,v),(w,x)$ be two critical edges of $G^\star$ and let $(u',v'),(w',x')$ be their limits in G^+. Further, let $\Gamma^\star$ be a drawing of $G^\star$, let Γ^+ be the drawing of G^+ induced by $\Gamma^\star$ and let φ^+ be the truth assignment of $\mathcal{S}(G^+)$ induced by Γ^+. Then (u,v) and (w,x) intersect an even number of times in $\Gamma^\star$ if and only if $\varphi^+(u'w') = \varphi^+(v'x')$.*

Proof. We may assume without loss of generality that any two edges cross at most

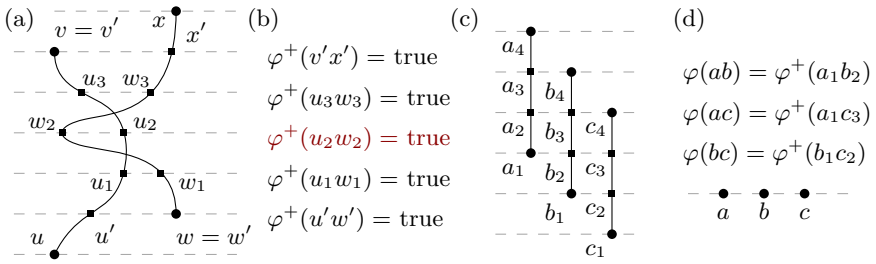


Figure 4.2: A Hanani-Tutte drawing (a) induces a truth assignment φ^+ that satisfies $\mathcal{S}(G^+)$ (b), the value where φ^+ differs from ψ^+ is highlighted in red. Using the subdivided stretch edges of G^+ (c), translate φ^+ to a satisfying assignment φ of $\mathcal{S}(G)$ (d).

once between consecutive levels by introducing sublevels if necessary. Let X be a crossing between (u, v) and (w, x) in $G^\star$; see Figure 4.2 (a). Further, let u_1, w_1 and u_2, w_2 be the subdivision vertices of (u, v) and (w, x) on the levels directly below and above X in $G^\star$, respectively. It is $\varphi^+(u_1w_1) = \neg\varphi^+(u_2w_2)$. In the reverse direction, $\varphi^+(u_1w_1) = \neg\varphi^+(u_2w_2)$ implies that (u, v) and (w, x) cross between the levels $\ell(u_1)$ and $\ell(u_2)$. Due to the definition of limits, any crossing between (u, v) and (w, x) in $G^\star$ must occur between the levels $\ell(u') = \ell(w')$ and $\ell(v') = \ell(x')$. Therefore, it is $\varphi^+(u'w') = \varphi^+(v'x')$ if and only if (u, v) and (w, x) cross an even number of times. □

We obtain that if $\Gamma^\star$ is a Hanani-Tutte drawing, then for critical edges (u, v), (w, x) the limits u', w' are ordered the same way as the limits v', x'. To obtain a satisfying assignment for $S(G)$, we then choose on each level the same ordering for the corresponding subdivision vertices. For any vertex v of $G^\star$ we use for every subdivision vertex on level $\ell(v)$ the order induced by $\Gamma^\star$ with regards to v. Note that this orders are only assigned for pairs of vertices and may be non-transitive on some layers.

Lemma 12. *Let G be a proper level graph and let $\Gamma^\star$ be a Hanani-Tutte drawing of $G^\star$. Then $\mathcal{S}(G^+)$ is satisfiable.*

Proof. We use $\Gamma^\star$ to define a truth assignment φ^+ that satisfies all clauses of $\mathcal{S}(G^+)$. Let Γ^+ be the drawing of G^+ induced by $\Gamma^\star$ and let ψ^+ denote the truth assignment induced by Γ^+. Note that ψ^+ does not necessarily satisfy the crossing clauses. Define φ^+ so that it satisfies all clauses of $\mathcal{S}(G^+)$ as follows. Let u'', w'' be two vertices of G^+ with $\ell(u'') = \ell(w'')$. If one of them is a vertex in $G^\star$, set $\varphi^+(u'', w'') = \psi^+(u'', w'')$. Otherwise u'', w'' are subdivision vertices of two edges (u, v), $(w, x) \in E(G^\star)$. If

they are independent, then they are critical. In that case their limits $(u', v'), (w', x')$ are already assigned consistently by Lemma 11. Then set $\varphi^+(u''w'') = \psi^+(u'w')$. If (u, v), (w, x) are adjacent, then we have $u = w$ or $v = x$. In the first case, we set $\varphi^+(u''w'') = \psi^+(v'x')$. In the second case, we set $\varphi^+(u''w'') = \psi^+(u'w')$.

So for critical edges $(u'', v''), (w'', x'') \in E(G^+)$ it is $\varphi^+(u''w'') = \varphi^+(v''x'')$ and also $\varphi^+(u''w'') = \neg\varphi^+(w''u'')$. Hence, assignment φ^+ satisfies $\mathcal{S}(G^+)$. See Figure 4.2 for a drawing Γ^+ (a) and the satisfying assignment of $\mathcal{S}(G^+)$ derived from it (b). □

Lemma 13. *Let G be a level graph. Then $\mathcal{S}(\overline{G})$ is satisfiable if and only if $\mathcal{S}(\overline{G}^+)$ is satisfiable.*

Proof. First assume there is a satisfying assignment φ for $\mathcal{S}(\overline{G})$. We define a mapping $O\colon V(\overline{G}^+) \to V(\overline{G})$ that maps each vertex of $V(\overline{G}^+)$ to a vertex in $\overline{G}$. Thereby we can describe φ^+ in relation to φ. For each vertex $v \in V(\overline{G}^+)$ that is part of a stretch edge $e(w)$ of $\overline{G}^\star$ for some vertex w of $\overline{G}$, we set $O(v) = w$. Each other vertex $v \in V(\overline{G}^+)$ is a subdivision vertex of an edge xx' of $\overline{G}^\star$ that is not a stretch edge. We then map v to vertex $w \in \{x, x'\}$ with $L(\ell(v)) = L(\ell(w))$. Now for distinct $x, y \in V(\overline{G}^+)$ with $O(x) \neq O(y)$ we set $\varphi^+(xy) = \varphi(O(x)O(y))$. If $O(x) = O(y)$, then x and y are subdivision vertices of non-stretch edges e_x, e_y in $\overline{G}^\star$. Then let v_x, v_y be the disjoint ends of e_x, e_y and we set $\varphi^+(xy) = \varphi(O(v_x), O(v_y))$.

Because φ satisfies the consistency constraint, we obtain directly that φ^+ satisfies the consistency constraint as well. Finally, consider the planarity constraint. Let $(u, v), (w, x) \in E(\overline{G}^+)$ be independent edges with $\ell(u) = \ell(w)$. If $L(\ell(u))$ equals $L(\ell(v))$, then we have $O(u) = O(v)$ and further $O(w) = O(x)$, and then it follows directly that $\varphi^+(uw) = \varphi(O(u)O(w)) = \varphi(O(v)O(x)) = \varphi^+(vx)$. Otherwise, we have $L(\ell(u)) \neq L(\ell(v))$. Then (u, v), (w, x) are subdivision edges of non-stretch edges e_u, e_w in $G^\star$. If e_u, e_w are not dependent, then $\varphi^+(uw) = \varphi(yz) = \varphi^+(vx)$ where y, z are the distinct ends of e_u, e_w. Otherwise, e_u, e_w are independent. Then we have $\varphi^+(uw) = \varphi(O(u)O(w)) = \varphi(O(v)O(x)) = \varphi^+(vx)$ since φ satisfies the planarity constraint.

Next assume there is a satisfying assignment φ^+ of $\mathcal{S}(\overline{G}^+)$. Let u, w be two vertices of G with $\ell(u) = \ell(w)$. Then the stretch edges $e(u), e(w)$ in $G^\star$ are critical by construction. Let $(u', u''), (w', w'')$ be their limits in G^+. Set $\varphi(uw) = \varphi^+(u'w')$. Because φ^+ is a satisfying assignment, all crossing clauses of $\mathcal{S}(G^+)$ are satisfied, which implies $\varphi^+(u'w') = \varphi^+(u''w'')$. The same is true for all subdivision vertices of $e(u)$ and $e(w)$ in G^+. Because φ^+ also satisfies the consistency clauses of $\mathcal{S}(G^+)$, this means that φ satisfies the consistency clauses of $\mathcal{S}(G)$. See Figure 4.2 for how $\mathcal{S}(G^+)$ is translated from G^+ (c) to G (d). Note that the resulting assignment is not necessarily transitive, e.g., it could be $\varphi(uv) = \varphi(vw) = \neg\varphi(uw)$.

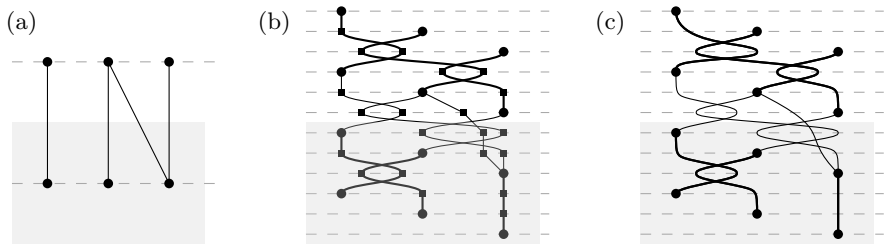


Figure 4.3: A proper level graph G together with a satisfying variable assignment φ (a) induces a drawing of G^+ (b), which induces a Hanani-Tutte drawing of $G^\star$ (c).

Consider two edges (u, v), (w, x) in G with $\ell(u) = \ell(w)$. Because G is proper, we do not have to consider other pairs of edges. Let (u', u''), (w', w'') be the limits of $e(u), e(w)$ in G^+. Further, let (v', v''), (x', x'') be the limits of $e(v), e(x)$ in G^+. Because there are disjoint directed paths from u' and w' to v' and x' and φ^+ is a satisfying assignment, it is $\varphi^+(u'w') = \varphi^+(v'x')$. Due to the construction of φ described in the previous paragraph, this means that it is $\varphi(uw) = \varphi(vx)$. Therefore, φ is a satisfying assignment of $\mathcal{S}(G)$ □

Lemma 14. *Let G be a level graph together with a satisfying truth assignment φ^+ of $\mathcal{S}(G^+)$. Then there exists a Hanani-Tutte drawing $\Gamma^\star$ of $G^\star$.*

Proof. We construct a drawing Γ^+ of G^+ from φ^+ as follows. Recall that by construction, every level of G^+ contains exactly one non-subdivision vertex. Let u denote the non-subdivision vertex of level i. Draw a subdivision vertex w on level i to the right of u if $\varphi^+(uw)$ is true and to the left of u otherwise. The relative order of subdivision vertices on either side of u can be chosen arbitrarily. Let $\Gamma^\star$ be the drawing of $G^\star$ induced by Γ^+. To see that $\Gamma^\star$ is a Hanani-Tutte drawing, consider two critical edges (u, v), (w, x) of $G^\star$. Let (u', v'), (w', x') denote their limits in G^+. One vertex of u' and v' (w' and x') is a subdivision vertex and the other one is not. The planarity constraint gives $\varphi^+(u'w') = \varphi^+(v'x')$ and by construction u', w' and v', x' are placed consistently on their respective levels. Moreover, Lemma 11 yields that (u, v) and (w, x) cross an even number of times in $\Gamma^\star$. □

Theorem 4. *Let G be a level graph. Then*

$$\begin{aligned} & \mathcal{S}(\overline{G}) \text{ is satisfiable} \\ \Leftrightarrow\ & \mathcal{S}(G^+) \text{ is satisfiable} \\ \Leftrightarrow\ & G^\star \text{ has a Hanani-Tutte level drawing} \end{aligned}$$

With Lemma 10 we obtain the claimed equivalence of the result of Randerath et al. [Ran+01] and the strong Hanani-Tutte Theorem. Namely, on the one hand, if every level graph with a Hanani-Tutte level drawing is level-planar, then we obtain for every level graph G where $\mathcal{S}(\overline{G})$ is satisfied that $G^\star$ has a Hanani-Tutte level drawing and is thus level-planar. This implies that G is level-planar. On the other hand, if every level graph H where $\mathcal{S}(H)$ is satisfiable is level-planar, then for every proper level-planar graph G with a Hanani-Tutte level drawing we trivially obtain a Hanani-Tutte level drawing for $G^\star$ which means $\mathcal{S}(\overline{G})$ is satisfiable. This implies $\overline{G}$ is level-planar and thus G is also level-planar.

Corollary 1. *The level-planar graphs are exactly the level graphs with a Hanani-Tutte level drawing if and only if they are exactly the level graphs G where $\mathcal{S}(G)$ is satisfiable.*

4.4 Radial Level-Planarity

In this section we present an analogous construction for radial level planarity. In contrast to level-planarity, we now have to consider cyclic orders on the levels, and even those may still leave some freedom for drawing the edges between adjacent levels. In the following we first construct a constraint system of radial level planarity for a proper level graph G, which is inspired by the one of Randerath et al. (Section 4.4.1). Afterwards, we slightly modify the construction of $G^\star$ (Section 4.4.2). Finally, in analogy to the level-planar case, we show that a satisfying assignment of our constraint system defines a satisfying assignment of the constraint system of G^+ (Section 4.4.3), and that this in turn corresponds to a Hanani-Tutte radial level drawing of $G^\star$ (Section 4.4.4) and vice versa (Section 4.4.5).

4.4.1 A Constraint System for Radial Level-Planarity

To deal with the increased complexity in the radial case, we state the constraints aiming for a linear equation system over $\mathbb{F}_2$. First observe, that we can formulate the constraints for the linear case as follows. We again use the set of variables $\mathcal{V} = \{uw \mid u, w \in V, u \neq w, \ell(u) = \ell(w)\}$ where now $uw \equiv 0$ means u is left of w on level $\ell(w)$.

$$
\begin{aligned}
&\forall uw \quad \in \mathcal{V} && : && uw && \equiv wu + 1 \\
&\forall uw, wy \in \mathcal{V} && : && uw \equiv 0 \wedge wy \equiv 0 \Rightarrow uy \equiv 0 \\
&\forall uw, vx \ \in \mathcal{V} \text{ with } (u,v), (w,x) \in E \text{ ind.} && : && uw && \equiv vx
\end{aligned}
$$

To obtain a constraint system for the radial case, we start with a special case that bears a strong similarity with the level-planar case. Namely, assume that G is a proper level graph that contains a directed path $P = \alpha_1, \ldots, \alpha_k$ that has exactly one vertex α_i on each level i. We now express the cyclic ordering on each level as linear orders whose first vertex is α_i. To this end, we introduce for each level the variables $\mathcal{V}_i = \{\alpha_i uv \mid u, v \in V_i \setminus \{\alpha_i\}\}$, where $\alpha_i uv \equiv 0$ means α_i, u, v are arranged clockwise on the circle representing level i. We further impose the following necessary and sufficient *linear ordering constraints* $\mathcal{L}_G(\alpha_i)$.

$$\forall \text{ distinct } u, v \in V \setminus \{\alpha_i\} : \quad \alpha_i uv \equiv \alpha_i vu + 1 \tag{4.4}$$

$$\forall \text{ pw. distinct } u, v, w \in V \setminus \{\alpha_i\} : \quad \alpha_i uv \equiv 0 \wedge \alpha_i vw \equiv 0 \Rightarrow \alpha_i uw \equiv 0 \tag{4.5}$$

It remains to constrain the cyclic orderings of vertices on adjacent levels so that the edges between them can be drawn without crossings. For two adjacent levels i and $i+1$, let $\varepsilon_i = (\alpha_i, \alpha_{i+1})$ be the *reference edge*. Let E_i be the set of edges (u, v) of G with $\ell(u) = i$ that are not adjacent to an endpoint of ε_i. Let $E_i^+ = \{(\alpha_i, v) \in E \setminus \{\varepsilon_i\}\}$ and $E_i^- = \{(u, \alpha_{i+1}) \in E \setminus \{\varepsilon_i\}\}$ denote the edges between levels i and $i+1$ adjacent to the reference edge ε_i.

In the context of the constraint formulation, we only consider drawings of the edges between levels i and $i+1$ where any pair of edges crosses at most once and, moreover, ε_i is not crossed. Note that this can always be achieved, independently of the orderings chosen for levels i and $i+1$. Then, the cyclic orderings of the vertices on the levels i and $i+1$ determine the drawings of all edges in E_i. In particular, two edges (u, v), $(u', v') \in E_i$ do not intersect if and only if $\alpha_i uu' \equiv \alpha_{i+1} vv'$; see Figure 4.4 (a). Therefore, we introduce constraint (4.6). For each edge $e \in E_i^+ \cup E_i^-$ it remains to decide whether it is embedded locally to the left or to the right of ε_i. We introduce for each such edge e a variable $l(e)$ where $l(e) \equiv 0$ represents the former case. Two edges $e \in E_i^-$, $f \in E_i^+$ do not cross if and only if $l(e) \equiv l(f) + 1$; see Figure 4.4 (b). This gives us constraint (4.7). It remains to forbid crossings between edges in E_i and edges in $E_i^+ \cup E_i^-$. An edge $e = (\alpha_i, v'') \in E_i^+$ and an edge $(u', v') \in E_i$ do not cross if and only if $l(e) \equiv \alpha_{i+1} v'v''$; see Figure 4.4 (c). Crossings with edges $(v, \alpha_{i+1}) \in E_i^-$ can be treated analogously. This yields constraints (4.8) and (4.9). We denote the planarity constraints (4.6)–(4.9) by $\mathcal{P}_G(\varepsilon_i)$, where $\varepsilon_i = (\alpha_i, \alpha_{i+1})$.

$$\forall \text{ independent } (u, v), (u', v') \in E_i : \quad \alpha_i uu' \equiv \alpha_{i+1} vv' \tag{4.6}$$

$$\forall e \in E_i^+, f \in E_i^- : \quad l(e) \equiv l(f) + 1 \tag{4.7}$$

$$\forall \text{ independent } (\alpha_i, v'') \in E_i^+, (u, v) \in E_i : \quad l(\alpha_i, v'') \equiv \alpha_{i+1} vv'' \tag{4.8}$$

$$\forall \text{ independent } (u'', \alpha_{i+1}) \in E_i^-, (u, v) \in E_i : \quad l(u'', \alpha_{i+1}) \equiv \alpha_i uu'' \tag{4.9}$$

It is not difficult to see that the transformation between Hanani-Tutte drawings and

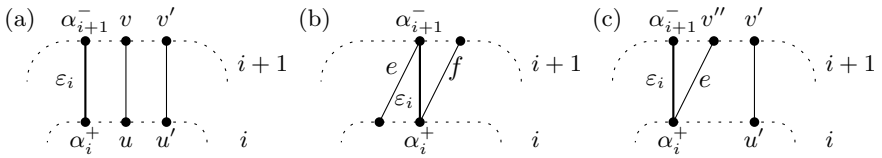


Figure 4.4: Illustration of the planarity constraints for radial planarity for the case of two edges in E_i (a), constraint (4.6); the case of an edge in $e \in E_i^-$ and an edge $f \in E_i^+$ (b), constraint (4.7); and the case of an edge in E_i and an edge $e \in E_i^+$ (c), constraint (4.8).

solutions of the constraint system without the transitivity constraints (4.5) can be performed as in the previous section. The only difference is that one has to deal with edges that share an endpoint with a reference ε_i.

In general, however, such a path P from level 1 to level k does not necessarily exist. Instead, we use an arbitrary reference edge between any two consecutive levels. More formally, we call a pair of sets $A^+ = \{\alpha_1^+, \dots, \alpha_k^+\}$, $A^- = \{\alpha_1^-, \dots, \alpha_k^-\}$ *reference sets* for G if we have $\alpha_1^- = \alpha_1^+$ and $\alpha_k^+ = \alpha_k^-$ and for $1 \le i \le k$ the *reference vertices* α_i^+, α_i^- lie on level i and for $1 \le i < k$ graph G contains the *reference edge* $\varepsilon_i = (\alpha_i^+, \alpha_{i+1}^-)$ unless there is no edge between level i and level $i+1$ at all. In that case, we can insert into every radial drawing of G the edge $(\alpha_i^+, \alpha_{i+1}^-)$ without creating new crossings. We therefore assume from now on that this case does not occur.

To express radial level-planarity, we express the cyclic orderings on each level twice, once with respect to the reference vertex α_i^+ and once with respect to the reference vertex α_i^-. To express planarity between adjacent levels i and $i+1$, we use the planarity constraints with respect to the reference edge ε_i. It only remains to specify that, if $\alpha_i^+ \neq \alpha_i^-$, the linear ordering with respect to these reference vertices must be linearizations of the same cyclic ordering. This is expressed by the following *cyclic ordering constraints* $\mathcal{C}_G(\alpha_i^+, \alpha_i^-)$; see Figure 4.5.

$$\forall \text{ distinct } u, v \in V_i \setminus \{\alpha_i^-, \alpha_i^+\} : \alpha_i^- uv + \alpha_i^+ uv + \alpha_i^- u\alpha_i^+ + \alpha_i^- v\alpha_i^+ \equiv 0 \qquad (4.10)$$

$$\forall \quad v \in V_i \setminus \{\alpha_i^-, \alpha_i^+\} : \alpha_i^- v\alpha_i^+ + \alpha_i^+\alpha_i^- v \equiv 0 \qquad (4.11)$$

The constraint set $\mathcal{S}'(G, A^+, A^-)$ consists of the linearization constraints $\mathcal{L}_G(\alpha_i^+)$ and $\mathcal{L}_G(\alpha_i^-)$, the cyclic ordering constraints $\mathcal{C}_G(\alpha_i^+, \alpha_i^-)$ for $i = 1, 2, \dots, k$ if $\alpha_i^+ \neq \alpha_i^-$, plus the planarity constraints $\mathcal{P}_G(\varepsilon_i)$ for $i = 1, 2, \dots, k-1$. This completes the definition of our constraint system.

Theorem 5. *Let G be a proper level graph with reference sets A^+, A^-. Then the constraint system $\mathcal{S}'(G, A^+, A^-)$ is satisfiable if and only if G is radial level-planar.*

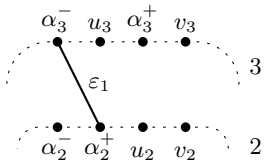


Figure 4.5: Illustration of the cyclic ordering constraints with $\alpha_2^- u_2 v_2 \equiv \alpha_2^+ u_2 v_2$ in level 2 and with $\alpha_3^- u_3 v_3 \not\equiv \alpha_3^+ u_3 v_3$ in level 3. Regarding Constraint (4.11), we have for example $\alpha_3^- u_3 \alpha_3^+$ and $\alpha_3^+ \alpha_3^- u_3$.

Moreover, the radial level planar drawings of G correspond bijectively to the satisfying assignments of $\mathcal{S}'(G, A^+, A^-)$.

Proof. Clearly, the radial level-planar drawings of G correspond injectively to the satisfying assignments of $\mathcal{S}'(G, A^+, A^-)$. For the reverse direction, consider a satisfying assignment φ of $\mathcal{S}'(G, A^+, A^-)$. Start by observing that the constraints (4.4) and (4.5) ensure that the orders defined for all $u, v \in V_i \setminus \{\alpha_i^-, \alpha_i^+\}$ by the variables $\alpha_i^- uv$ and $\alpha_i^+ uv$ are indeed linear. Define $\mathbb{A} = \{v \in V_i \setminus \{\alpha_i^-, \alpha_i^+\} \mid \alpha_i^- v \alpha_i^+\}$ and $\mathbb{B} = \{v \in V_i \setminus \{\alpha_i^-, \alpha_i^+\} \mid \alpha_i^- \alpha_i^+ v\}$. Let σ_i^-, σ_i^+ denote the cyclic orders of the vertices on level i induced by the assignments for variables starting with a reference vertex in A^- and A^+, respectively. We have to show $\sigma_i^- = \sigma_i^+$. To do so, consider pairwise distinct $u, v, w \in V_i$ and show $(u, v, w) \in \sigma_i^- \Rightarrow (u, v, w) \in \sigma_i^+$. We distinguish four cases.

1. $u, v, w \notin \{\alpha_i^-, \alpha_i^+\}$. Assume $\alpha_i^- uv \equiv \alpha_i^- vw \equiv \alpha_i^- uw \equiv 0$ without loss of generality, i.e., $(u, v, w) \in \sigma_i^-$. We distinguish four cases based on how the vertices u, v and w are distributed over the sets $\mathbb{A}$ and $\mathbb{B}$.
   a) $u, v, w \in \mathbb{A}$. It is $\alpha_i^- u \alpha_i^+ \equiv \alpha_i^- v \alpha_i^+ \equiv \alpha_i^- w \alpha_i^+ \equiv 0$. With constraint (4.10) we conclude $\alpha_i^+ uv \equiv \alpha_i^+ vw \equiv \alpha_i^+ uw \equiv 0$, which gives $(u, v, w) \in \sigma_i^+$.
   b) $u, v, \in \mathbb{A}, w \in \mathbb{B}$. It is $\alpha_i^- u \alpha_i^+ \equiv \alpha_i^- v \alpha_i^+ \equiv \alpha_i^- w \alpha_i^+ + 1 \equiv 0$. Using constraint (4.10) we conclude $\alpha_i^+ uv \equiv \alpha_i^+ vw + 1 \equiv \alpha_i^+ uw + 1$, which gives $(u, v, w) \in \sigma_i^+$.
   c) $u, \in \mathbb{A}, v, w \in \mathbb{B}$; similar to case 1a.
   d) $u, v, w \in \mathbb{B}$; similar to case 1b.
2. $u, v \notin \{\alpha_i^-, \alpha_i^+\}$, $w = \alpha_i^-$. We distinguish three cases based on how the vertices u and v are distributed over the sets $\mathbb{A}$ and $\mathbb{B}$.
   a) $u, v \in \mathbb{A}$. I.e., it is $\alpha_i^- u \alpha_i^+ \equiv \alpha_i^- v \alpha_i^+ \equiv 0$. Constraint (4.10) gives $\alpha_i^+ uv \equiv 0$. Constraint (4.11) gives $\alpha_i^+ v \alpha_i^- \equiv \alpha_i^+ u \alpha_i^- \equiv 0$. Hence, $(u, v, \alpha_i^- = w) \in \sigma_i^+$.

b) $u \in \mathbb{A}, v \in \mathbb{B}$. I.e., it is $\alpha_i^- u \alpha_i^+ \equiv \alpha_i^- \alpha_i^+ v \equiv \alpha_i^- uv \equiv 0$. Using constraints (4.4) and (4.11) we obtain $\alpha_i^+ v \alpha_i^- \equiv 0$ from $\alpha_i^- \alpha_i^+ v \equiv 0$. Using constraint (4.11) we get $\alpha_i^+ \alpha_i^- u \equiv 0$ from $\alpha_i^- u \alpha_i^+ \equiv 0$. Finally, we use constraint (4.10) to obtain $\alpha_i^+ uv \equiv 0$. This means $(u, v, \alpha_i^- = w) \in \sigma_i^+$.

c) $u, v \in \mathbb{B}$. Use constraint (4.11) to obtain $\alpha_i^+ u \alpha_i^- \equiv \alpha_i^+ v \alpha_i^-$ from $\alpha_i^- \alpha_i^+ u \equiv 0$ and $\alpha_i^- \alpha_i^+ v \equiv 0$, respectively. Then use constraint (4.10) to obtain $\alpha_i^+ uv \equiv 0$, which means $(u, v, \alpha_i^- = w) \in \sigma_i^+$.

3. $u, v \notin \{\alpha_i^-, \alpha_i^+\}$, $w = \alpha_i^+$. We distinguish three cases based on how the vertices u and v are distributed over the sets $\mathbb{A}$ and $\mathbb{B}$.

a) $u, v \in \mathbb{A}$. Constraint (4.11) gives $\alpha_i^+ \alpha_i^- v \equiv \alpha_i^+ \alpha_i^- u \equiv 0$ from $\alpha_i^- \alpha_i^+ v \equiv 0$ and $\alpha_i^- \alpha_i^+ u \equiv 0$, respectively. Then use constraint (4.10) to obtain $\alpha_i^+ uv \equiv 0$, which means $(u, v, \alpha_i^+ = w) \in \sigma_i^+$.

b) $u \in \mathbb{B}, v \in \mathbb{A}$. Use constraint (4.11) to obtain $\alpha_i^+ \alpha_i^- v \equiv \alpha_i^+ u \alpha_i^- \equiv 0$ from $\alpha_i^- v \alpha_i^+ \equiv 0$ and $\alpha_i^- \alpha_i^+ u \equiv 0$, respectively. Then use constraint (4.10) to obtain $\alpha_i^+ uv \equiv 0$, which means $(u, v, \alpha_i^+ = w) \in \sigma_i^+$.

c) $u, v \in \mathbb{B}$. Use constraints (4.4) and (4.11) to obtain $\alpha_i^+ u \alpha_i^- \equiv \alpha_i^+ v \alpha_i^- \equiv 0$ from $\alpha_i^- \alpha_i^+ u \equiv 0$ and $\alpha_i^- \alpha_i^+ v \equiv 0$, respectively. Then use constraint (4.10) to obtain $\alpha_i^+ uv \equiv 0$, which means $(u, v, \alpha_i^+ = w) \in \sigma_i^+$.

4. $v \notin \{\alpha_i^-, \alpha_i^+\}, u, w \in \{\alpha_i^-, \alpha_i^+\}$. We may assume $u = \alpha_i^-$ and $w = \alpha_i^+$ without loss of generality. Then $(u, v, w) \in \sigma_i^-$ gives $\alpha_i^- v \alpha_i^+ \equiv 0$ and constraint (4.11) yields $\alpha_i^+ \alpha_i^- v \equiv 0$, which means $(\alpha_i^- = u, v, \alpha_i^+ = w) \in \sigma_i^+$.

Therefore, φ induces well-defined cyclic orders of the vertices on all levels. It remains to be shown that no two edges cross. Recalling that G is proper, it is sufficient to show that no two edges $e, f \in (E_i \cup E_i^- \cup E_i^+)$ cross for $i = 1, 2, \ldots, k$. We distinguish five cases based on how the edges e and f are distributed across the sets E_i, E_i^- and E_i^+.

1. $e, f \in E_i$. Then constraint (4.6) together with the fact that no edge may cross ε_i implies that e and f do not cross.

2. $e \in E_i, f \in E_i^-$. Let $e = (u, v)$ and $f = (u'', \alpha_{i+1}^-)$ and suppose $l(f) \equiv 0$ (the case $l(f) \equiv 1$ works symmetrically). Then constraint (4.9) ensures that e and f do not cross.

3. $e \in E_i, f \in E_i^+$; works symmetrically to case 2 with constraint (4.8).

4. $e \in E_i^-, f \in E_i^+$. Then constraint (4.7) ensures that e and f are embedded locally to the left and right of ε_i, respectively, or vice versa. Together with the fact that no edge may cross ε_i, this means that e and f do not cross.

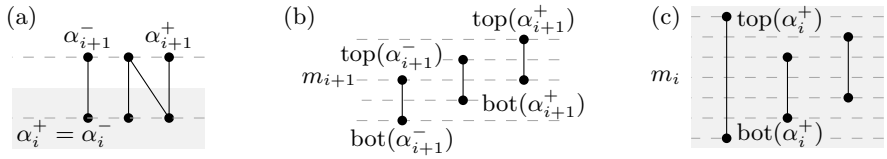


Figure 4.6: Illustration of the modified construction of the stretch edges for $G^\star$ for the graph G in (a). The stretch edges for level $i+1$ where $\alpha^+_{i+1} \neq \alpha^-_{i+1}$ (b) and for level i where $\alpha^+_i = \alpha^-_i$ (c).

5. $e, f \in E^-_i$ or $e, f \in E^+_i$. Because e and f share an endpoint they do not cross.

Therefore, no two edges cross, which means that φ induces a radial level-planar drawing. □

Similar to Section 4.3, we define a reduced constraints system $\mathcal{S}(G, A^+, A^-)$ obtained from $\mathcal{S}'(G, A^+, A^-)$ by dropping constraint (4.5). Our main result is that $S(G, A^+, A^-)$ is satisfiable if and only if G is radial level-planar. The proof works by showing equivalence to the existence of a Hanani-Tutte Drawing of $G^\star$.

4.4.2 Modified Star Form

We slightly modify the splitting and perturbation operation in the construction of the star form $G^\star$ of G for each level i. This is necessary since we need a special treatment of the reference vertices α^+_i and α^-_i on each level i. Namely, we want the stretch edge of α^+_i to end at the highest level and the stretch edge of α^-_i to end at the lowest level. Further, for $\alpha^-_i \neq \alpha^+_i$ we want the other ends of those two stretch edges to lie on the same level m_i. In each case, we want level m_i to intersect all stretch edges.

Consider the level i containing the n_i vertices $v_1, \ldots, v_{n_i}$. If $\alpha^+_i \neq \alpha^-_i$, then we choose the numbering of the vertices such that $v_1 = \alpha^-_i$ and $v_{n_i} = \alpha^+_i$. We replace i by $2n_i - 1$ levels $(i, 1), (i, 2), \ldots, (i, 2n_i - 1)$, which is one level less than previously. Similar to before, we replace each vertex v_j by two vertices $\mathrm{bot}(v_j)$ and $\mathrm{top}(v_j)$ with $\ell(\mathrm{bot}(v_j)) = (i, j)$ and $\ell(\mathrm{top}(v_j)) = (i, n_i - 1 + j)$ and the corresponding stretch edge $(\mathrm{bot}(v_j), \mathrm{top}(v_j))$; see Figure 4.6 (b). This ensures that the construction works as before, except that the middle level $m_i = (i, n_i)$ contains two vertices, namely $\alpha^{+\prime\prime}_i$ and $\alpha^{-\prime}_i$.

If, on the other hand, $\alpha^+_i = \alpha^-_i$, then we choose $v_1 = \alpha^+_i$. But now we replace level i by $2n_i + 1$ levels $(i, 1), \ldots, (i, 2n_i + 1)$. Replace v_1 by vertices $\mathrm{bot}(v_1), \mathrm{top}(v_1)$ with $\ell(\mathrm{bot}(v_1)) = (i, 1)$ and $\ell(\mathrm{top}(v_1)) = (i, 2n_i + 1)$. Replace all other v_j with vertices $\mathrm{bot}(v_j), \mathrm{top}(v_j)$ with $\ell(\mathrm{bot}(v_j)) = (i, j)$ and $\ell(\mathrm{top}(v_j)) = (i, n_i + 1 + j)$. For

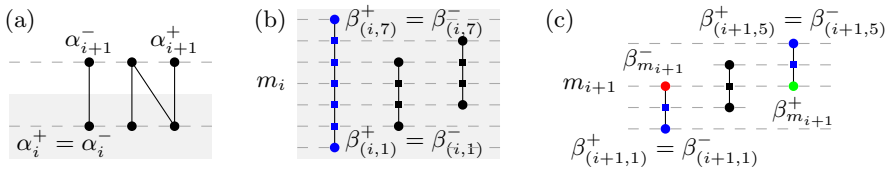


Figure 4.7: Definition of β^+, β^- in the assignment for G^+ for the same graph as in Figure 4.6 (a). Vertices β^+ (β^-) are drawn in green (red), or in blue if they coincide.

all j, we add the stretch edge $(\text{bot}(v_j), \text{top}(v_j))$ as before; see Figure 4.6 (c). This construction ensures that the stretch edge of $\alpha^+_i = \alpha^-_i$ starts in the first new level $(i, 1)$ and ends in the last new level $(i, 2n_i+1)$, and the middle level $m_i = (i, n_i + 1)$ contains no vertex.

As before, we replace each original edge (u, v) of the input graph G by the edge $(\text{top}(u), \text{bot}(v))$ connecting the upper endpoint of the stretch edge of u to the lower endpoint of the stretch edge of v. Observe that the construction preserves the properties that for each level i the middle level m_i of the levels that replace i intersects all stretch edges of vertices on level i. Therefore, Lemma 10 also holds for this modified version of $G^\star$ and its proper subdivision G^+. For each vertex v of G we use $e(v) = (\text{bot}(v), \text{top}(v))$ to denote its stretch edge.

Again, we use the function L that maps each level (i, j) of $G^\star$ or G^+ to the level i of G. As suitable, we treat the tuples of levels as natural numbers. For an edge e of $G^\star$ and a level i that intersects e, we denote by e_i the subdivision vertex of e at level i in G^+. For two levels i and j that both intersect an edge e of $G^\star$, we denote by e^j_i the path from e_i to e_j in G^+.

4.4.3 Constraint System and Assignment for G^+

Let G be a proper level graph with reference sets A^+, A^-. We now choose reference sets B^+, B^- for G^+ that are based on the reference sets A^+, A^-. Let j be any level of $G^\star$ and let $L(j) = i$ be the corresponding level of G. Define two vertices β^+_j, β^-_j as follows. If $\alpha^-_i = \alpha^+_i$, set $\beta^-_j = \beta^+_j = e(\alpha^-_i)_j$; see Figure 4.7 (b). Otherwise, the choice is based on whether j is the middle level $m = m_i$ of the levels $L^{-1}(i)$ that replace level i of G, or whether j lies above or below m. Choose $\beta^-_m = \text{top}(\alpha^+_i)$ and $\beta^+_m = \text{bot}(\alpha^-_{i+1})$. For $j < m$, choose $\beta^-_j = \beta^+_j = e(\alpha^-_i)_j$ and for $j > m$, choose $\beta^-_j = \beta^+_j = e(\alpha^+_i)_j$; see Figure 4.7 (c).

We set B^+ to be the set containing all β^+_j and likewise for B^-. Our next step is to construct from a satisfying assignment φ of $\mathcal{S}(G, A^+, A^-)$ a corresponding satisfying assignment φ^+ of $\mathcal{S}(G^+, B^+, B^-)$. The construction follows the approach from

Lemma 14 and makes use of the fact that G^+ is essentially a stretched and perturbed version of G.

Lemma 15. *If $\mathcal{S}(G, A^+, A^-)$ is satisfiable, then $\mathcal{S}(G^+, B^+, B^-)$ is satisfiable.*

For the proof of this result, we consider a satisfying assignment φ for $\mathcal{S}(G, A^+, A^-)$. We now derive an assignment φ^+ for $\mathcal{S}(G^+, B^+, B^-)$ from φ. Afterwards, we show that φ^+ satisfies $\mathcal{S}(G^+, B^+, B^-)$.

Construction of φ^+. Let $e = uu', f = vv'$ be two edges between level i and level $i+1$ of G and let $\varepsilon_i = (\alpha_i^+, \alpha_{i+1}^-)$ denote the reference edge between these levels. We introduce a function $\psi(\varepsilon_i, e, f)$ to deduce the order of the edges ε_i, e, f in a drawing that corresponds to φ.

$$\psi(\varepsilon_i, e, f) \equiv \begin{cases} \varphi(\alpha_i^+ uv) & , \text{ if } \alpha_i^+, u, v \text{ are pairwise distinct} \\ \varphi(\alpha_{i+1}^- u'v') & , \text{ if } \alpha_{i+1}^-, u', v' \text{ are pairwise distinct} \\ \varphi(l(f)) & , \text{ if } \alpha_i^+ = v \neq u \text{ or } \alpha_{i+1}^- = v' \neq u' \\ \varphi(l(e)) + 1 & , \text{ if } \alpha_i^+ = u \neq v \text{ or } \alpha_{i+1}^- = u' \neq v' \end{cases}$$

Note that if α_i^+, u, v and α_{i+1}^-, u', v' are pairwise distinct, then by Eq. (4.6) we have that $\varphi(\alpha_i^+ uv) \equiv \varphi(\alpha_{i+1}^- uv)$. Similarly, if $\alpha_i^+ = v$ and $\alpha_{i+1}^- = u'$ (or $\alpha_i^+ = u$ and $\alpha_{i+1}^- = v'$), then by Eq. (4.7) it is $\varphi(l(f)) \equiv \varphi(l(e) + 1$. Therefore ψ is well-defined.

Based on this, we can now define the orderings of triples of subdivision vertices of G^+, which leads to an assignment φ^+ for $\mathcal{S}(G^+, B^+, B^-)$. To this end, we define a mapping $O\colon V(G^+) \to V$ that maps each vertex of $V(G^+)$ to a vertex in G. For each vertex v of G^+ that is part of a stretch edge $e(w)$ of $G^\star$ for some vertex w of G, we set $O(v) = w$. For an example, see the orange vertex w of G in Figure 4.8. The encircled orange vertices in G^+ are mapped to w. It remains to define $O(v)$ for vertices $v \in V(G^+)$ that are subdivision vertices of an edge xx' of $G^\star$ that its not a stretch edge. Then the original levels satisfy $L(\ell(x')) = L(\ell(x)) + 1$, and we map v to the vertex w with $x = \text{top}(w)$ if $L(\ell(v)) = L(\ell(x))$ and to the vertex w' with $x' = \text{bot}(w')$, otherwise. For an example, see the edge (x, x') of G in Figure 4.8. The encircled purple (green) subdivision vertices in G^+ are mapped to x (to x').

We are now ready to define the assignment φ^+ for $\mathcal{S}(G^+, B^+, B^-)$ for a satisfying assignment φ of $\mathcal{S}(G, A^+, A^-)$. Let $x, y, z \in V_j$ be three distinct vertices on level (i, j) and such that $x \in \{\beta_j^-, \beta_j^+\}$. Observe that $O(x) \in \{\alpha_i^+, \alpha_{i+1}^-\}$. If the vertices $O(x), O(y), O(z)$ are pairwise distinct, we set

$$\varphi^+(xyz) = \varphi\left(O(x)O(y)O(z)\right) .$$

Otherwise, two of the corresponding vertices are the same. Hence, two vertices of x, y, z are subdivision vertices of original edges in G. Note that x can by definition

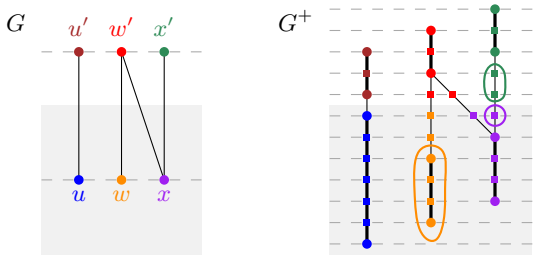


Figure 4.8: Definition of the mapping $O\colon V(G^+) \to V$. Vertices of G^+ are mapped to the vertex of G with the matching color.

of $G^\star$ not be such a subdivision vertex. Hence, vertex y is a subdivision vertex of an edge $e \in G$ and vertex z is a subdivision vertex of an edge $f \in G$, both of which connect a vertex of level i to a vertex on level $i+1$. Then we set

$$\varphi^+(xyz) \equiv \psi(\varepsilon_i, e, f)\,,$$

where $\varepsilon_i = (\alpha_i^+, \alpha_{i+1}^-)$ is the reference edge between these levels. Moreover, we set $\varphi^+(l(e)) \equiv \varphi(l(O(u), O(v)))$ for each edge $e = (u, v)$ of G^+ with $u = \beta_j^+$ or $v = \beta_{j+1}^-$ but not both for some level j of G^+.

Lemma 16. *The assignment φ^+ satisfies $\mathcal{S}(G^+, B^+, B^-)$.*

Proof. We check that φ^+ satisfies each part of the constraint system. We denote by V_j the vertices of G^+ on level j.

φ^+ satisfies $\mathcal{L}'_{G^+}(\beta_j^+)$. Let $\beta_j^+ \in B^+$. We aim to show that φ^+ satisfies $\mathcal{L}'_{G^+}(\beta_j^+)$. Let $y, z \in V_j \setminus \{\beta_j^+\}$ be distinct. If $O(\beta_j^+), O(y), O(z)$ are distinct, then we obtain that $\varphi^+(\beta_j^+ yz) \equiv \varphi(O(\beta_j^+)O(y)O(z))$ as well as $\varphi^+(\beta_j^+ zy) \equiv \varphi(O(\beta_j^+)O(z)O(y))$. Note that $O(\beta_j^+) \in \{\alpha_{L(j)}^+, \alpha_{L(j)}^-\}$. Since φ satisfies $\mathcal{L}'_G(\alpha_{L(j)}^+)$ and $\mathcal{L}'_G(\alpha_{L(j)}^-)$, we obtain $\varphi^+(\beta_j^+ yz) \equiv \varphi^+(\beta_j^+ zy) + 1$.

If $O(\beta_j^+), O(y), O(z)$ are not distinct, then we must have $O(y) = O(z)$, since β_j^- cannot be a subdivision vertex of an edge that is not a stretch edge. Hence, the vertices y, z are subdivision vertices of edges e, f of $G^\star$ connecting vertices on levels i and $i+1$. Note that this also implies $O(\beta_j^+) \neq O(y)$. We obtain $\varphi^+(\beta_j^-, y, z) \equiv \psi(\varepsilon_i, e, f)$ and $\varphi^+(\beta_j^-, z, y) \equiv \psi(\varepsilon_i, f, e)$. Since $O(y) = O(z) \neq O(\beta_j^-)$, we have with the definition of ψ that $\psi(\varepsilon_i, e, f) \equiv \psi(\varepsilon_i, f, e) + 1$. This yields $\varphi^+(\beta_j^-, y, z) \equiv \varphi^+(\beta_j^-, z, y) + 1$.

φ^+ satisfies $\mathcal{L}'_{G^+}(\beta_j^-)$. This can be argued analogously to $\mathcal{L}'_{G^+}(\beta_j^+)$.

φ^+ satisfies $\mathcal{C}_{G^+}(\beta_j^+, \beta_j^-)$. We next show that φ^+ satisfies the constraints $\mathcal{C}_{G^+}(\beta_j^+, \beta_j^-)$ for all j with $\beta_j^- \neq \beta_j^+$. By definition of $G^\star$ and B^+, B^-, these constraints are non-trivial only for levels $j = m_i$ for a level i of G with $\alpha_i^+ \neq \alpha_i^-$.

Note that due to the construction of $G^\star$, level m_i is only crossed by stretch edges, which implies that the restriction of O to the vertices of G^+ on level m_i is injective. Moreover, $O(\beta_j^+) = \alpha_i^+$ and $O(\beta_j^-) = \alpha_i^-$. Hence, the triple $(x, y, z) \in V_j$ of distinct vertices map injectively to triples of distinct vertices of G on level i. Since the value of $\varphi^+(xyz)$ is defined in terms of $\varphi(O(x)O(y)O(z))$, it follows that φ^+ satisfies Eq. (4.10) and (4.11) if φ does.

φ^+ satisfies $\mathcal{P}_{G^+}(\beta_j^+, \beta_{j+1}^-)$. We finally show that φ^+ satisfies the constraints $\mathcal{P}_{G^+}(\delta_j)$ for any two consecutive levels j and $j+1$ of G^+. We distinguish two cases, based on whether $L(j) = L(j+1)$ or $L(j+1) = L(j) + 1$.

1. $L(j) = L(j+1) = i$. First observe that, except for $(\beta_j^+, \beta_{j+1}^-)$, the reference vertex β_j^+ has no outgoing edges and β_{j+1}^- has no incoming edges. Namely, vertex β_j^+ can only have more outgoing edges, if $\beta_j^+ = \mathrm{top}(\alpha_i^+)$, and vertex β_{j+1}^- can only have more incoming edges, if $\beta_{j+1}^- = \mathrm{bot}(\alpha_i^-)$. But $\mathrm{bot}(\alpha_i^-)$ and $\mathrm{top}(\alpha_i^+)$ occupy the first and the last level of G^+ corresponding to level i of G. Hence this is not possible since $L(j) = L(j+1)$. Therefore the equations (4.7)–(4.9) are trivially satisfied.

   Now consider two independent edges $(y, y'), (z, z')$ between levels j and $j+1$ that are different from $(\beta_j^+, \beta_{j+1}^-)$.

   Since the vertices y, y' are adjacent in G^+, they either both subdivide the same edge of $G^\star$, or one of them is a vertex of $G^\star$ and the other one subdivides an incident edge. As $L(j) = L(j+1)$, we obtain $O(y) = O(y')$ in both cases. Similarly, we obtain $O(z) = O(z')$. If $j \geq m_i$, then we have $O(\beta_j^+) = O(\beta_{j+1}^-) = \alpha_i^+$. If $j < m_i$, then we have $O(\beta_j^+) = O(\beta_{j+1}^-) = \alpha_i^-$. In both cases we have $O(\beta_j^+) = O(\beta_{j+1}^-)$.

   Observe that $O(y) = O(\beta_j^+) = \alpha_i^-$ implies that $j < m_i$, and hence y is a subdivision vertex of an outgoing edge of $\mathrm{bot}(\alpha_i^-)$ in $G^\star$. This is, however, impossible since the only outgoing edge of $\mathrm{bot}(\alpha_i^-)$ is the stretch edge of α_i^-. Symmetrically, $O(y) = O(\beta_j^+) = \alpha_i^+$ implies $j \geq m_i$, and hence that y is a subdivision vertex of an incoming edge of $\mathrm{top}(\alpha_i^+)$, which is again impossible. Thus we find that $O(\beta_j^+) \neq O(y)$. Likewise, it is $O(\beta_j^+) \neq O(z)$.

   If $O(y) \neq O(z)$, then the three vertices $O(\beta_j^+) = O(\beta_j^-)$, $O(y) = O(y')$ and $O(z) = O(z')$ are pairwise distinct. Thus O maps the triples $t_1 = \beta_j^+ yz$ and $t_2 = \beta_{j+1}^- y'z'$ to the same triple t. Since $\varphi^+(t_i) \equiv \varphi(t)$ for $i = 1, 2$, it follows that $\varphi^+(t_1) \equiv \varphi^+(t_2)$, i.e., φ^+ satisfies Eq. (4.6).

If $O(y) = O(z)$, then $O(y') = O(z')$, and $(y, y'), (z, z')$ originate from two edges e, f of G. Now, if $j < m_i$, then e and f connect vertices on level $i-1$ to vertices on level i. In this case, $\varphi^+(\beta_j^+ ef)$ and $\varphi^+(\beta_j^- ef)$ are both defined in terms of $\psi(\varepsilon, e, f)$, where $\varepsilon = (\alpha_{i-1}^+, \alpha_i^-)$. If $j \geq m_i$, then e and f connect vertices on levels i and $i+1$. Then φ^+ of both triples is defined as $\psi(\varepsilon, e, f)$ for $\varepsilon = (\alpha_i^+, \alpha_{i+1}^-)$. In both case Eq. (4.6) is satisfied.

2. $i = L(j) < L(j+1) = i+1$. In this case $\beta_j^+ = \text{top}(\alpha_i^+)$ and $\beta_{j+1}^- = \text{bot}(\alpha_{i+1}^-)$. Let $xx' \in G^+$ be any edge between level j and level $j+1$. Since $L(j) \neq L(j+1)$, we have that $e = (O(x), O(x'))$ is an edge of G. Further, we obtain

$$\begin{aligned}
(x, x') \in E_i(G^+) &\Leftrightarrow e \in E_i \\
(x, x') \in E_i^+(G^+) \Leftrightarrow x = \beta_j^+ &\Leftrightarrow v(x) = \alpha_i^+ \Leftrightarrow e \in E_i^+ \\
(x, x') \in E_i^-(G^+) \Leftrightarrow x' = \beta_{j+1}^- &\Leftrightarrow v(x') = \alpha_{i+1}^- \Leftrightarrow e \in E_i^- .
\end{aligned}$$

Let yy', zz' be distinct edges between levels j and $j+1$ in G^+ that are different from $(\beta_j^+, \beta_{j+1}^-)$. We have that $e = (O(y), O(y'))$ and $f = (O(z), O(z'))$ are edges of G. We further distinguish cases based on whether e and f are independent.

   a) e and f are independent. If $e, f \in E_i$, then we have $(y, y'), (z, z') \in E_i(G^+)$ and by definition of φ^+ it is $\varphi^+(\beta_j^+ yz) \equiv \varphi(\alpha_i^+ O(y)O(z))$ and furthermore $\varphi^+(\beta_{j+1}^- y'z') \equiv \varphi(\alpha_{i+1}^- O(y')O(z'))$. Since φ satisfies $\mathcal{P}_G(\alpha_i^+, \alpha_{i+1}^-)$, it follows that φ^+ satisfies Eq. (4.6) for G^+ and edges $(y, y'), (z, z')$.

   If $e \in E_i^+$ and $f \in E_i^-$, then $(y, y') \in E_j^+(G^+)$ and $(z, z') \in E_i^-(G^+)$. Then, by definition of φ^+, we have $\varphi^+(l(y, y')) \equiv \varphi(l(\alpha_i^+ O(y'))$ and also $\varphi^+(l(z, z')) \equiv \varphi(l(O(z)\alpha_{i+1}^-))$. Since φ satisfies $\mathcal{P}_G(\varepsilon_i)$, it follows that φ^+ satisfies Eq. (4.7) for G^+ and edges $(y, y'), (z, z')$.

   If $e \in E_i^+$ and $f \in E_i$, then $(y, y') \in E_i^+(G^+)$ and $(z, z') \in E_i(G^+)$. It then follows from the definition of φ^+ that $\varphi^+(l(\beta_j^+, y')) \equiv \varphi(l(\alpha_i^+, O(y')))$ and $\varphi^+(\beta_{j+1}^- z'y') \equiv \varphi(\alpha_{i+1}^- O(z')O(y'))$. Since φ satisfies $\mathcal{P}_G(\varepsilon_i)$, it follows that φ^+ satisfies Eq. (4.8) for G^+ and edges $(y, y'), (z, z')$. The case $e \in E_i^-$ and $f \in E_i$ can be argued analogously.

   b) e, f are dependent. If $O(y) = O(z)$, then y and z are subdivision vertices of two edges e', f' in $G^\star$ that share their source, which hence lies on a level strictly below j. In particular, it is $y, z \neq \text{top}(\alpha_i^+) = \beta_j^+$. Moreover, it is $O(y') \neq O(z')$. Then $\varphi^+(\beta_j^+ yz) \equiv \varphi^+(\beta_{j+1}^- y'z') \equiv \psi(\varepsilon_i, e, f)$ by definition, where $\varepsilon_i = (\alpha_i^+, \alpha_{i+1}^-)$.

   If the vertices $O(\beta_j^+) = \alpha_{i+1}^-$, $O(y')$, $O(z')$ are pairwise distinct, then we have $(y, y'), (z, z') \in E_i(G^+)$. Then $\varphi^+(\beta_j^+ yz) \equiv \varphi(\alpha_{i+1}^- O(y')O(z'))$

and $\varphi^+(\beta^-_{j+1}y'z') \equiv \varphi(\alpha^-_{i+1}O(y')O(z'))$. Therefore φ^+ satisfies Eq. (4.6) for G^+ and edges (y, y') and (z, z').

If $\alpha^-_{i+1} = O(y') \neq O(z')$, then $(y, y') \in E^+_i(G^+)$ and $(z, z') \in E_i(G^+)$. Then $\varphi^+(\beta^+_j yz) \equiv \varphi(l(e))$ and $\varphi^+(\beta^-_{j+1}y'z') \equiv \varphi(l(e))$. Therefore φ^+ satisfies Eq. (4.8) for G^+ and edges $(y, y)', (z, z')$.

Analogously, we obtain for $\alpha^-_{i+1} = O(z') \neq O(y')$ that φ^+ satisfies Eq. (4.8) for G^+ and edges (y, y') and (z, z').

Finally, the case $O(y') = O(z')$ can be handled analogously to the case $O(y) = O(z)$.

This concludes the proof that φ^+ satisfies $\mathcal{S}(G^+, B^+, B^-)$. □

4.4.4 From a Satisfying Assignment to a Hanani-Tutte Drawing

Let G be a level graph with reference sets A^+, A^- and let B^+, B^- be corresponding reference sets for G^+. Finally, let φ^+ be a satisfying assignment for $\mathcal{S}(G^+, B^+, B^-)$. We construct a radial drawing Γ^+ of G^+, from which we obtain the drawing $\Gamma^\star$ of $G^\star$ by smoothing the subdivision vertices. Afterwards we show that $\Gamma^\star$ is a Hanani-Tutte drawing.

We construct Γ^+ as follows. Consider a level j of G^+ and let $i = L(j)$ be the original level of G. First assume $j = m_i$. If $\beta^-_j = \beta^+_j$, then we place all vertices of $V_j(G^+)$ in arbitrary order. Otherwise, we place β^-_j and β^+_j arbitrarily on the circle representing the level m_i. We then place each vertex $v \in V_j(G^+) \setminus \{\beta^-_j, \beta^+_j\}$ such that β^-_j, v, β^+_j are ordered clockwise if and only if $\varphi(\beta^-_j v \beta^+_j) \equiv 0$ (i.e., we place v on the correct side of β^-_j and β^+_j and arrange the vertices on both sides of β^-_j and β^+_j arbitrarily).

Next assume $j \neq m_i$. Then there is exactly one vertex $\xi \in V_j(G^+) \cap V(G^\star)$. If $\xi \in B^-$, then we place all vertices of $V_j(G^+)$ in arbitrary order on the circle representing the level j. Otherwise, we place β^-_j and ξ arbitrarily. We then place any vertex $v \in V_j(G^+) \setminus \{\beta^-_j, \xi\}$ such that β^-_j, ξ, v are ordered clockwise if and only if $\varphi^+(\beta^-_j \xi v) \equiv 0$. Again, we arrange the vertices on either side of β^-_j and ξ arbitrarily. We have now fixed the positions of all vertices and it remains to draw the edges.

Consider two consecutive levels j and $j+1$ of G^+. We draw the edges in $E_j(G^+)$ such that they do not cross the reference edges in $E(G^+) \cap (B^+ \times B^-)$. We draw an edge $e = (\beta^+_j, x') \in E^+_j(G^+)$ such that it is locally left of (β^+_j, β^-_j) if and only if $\varphi^+(l(e)) \equiv 0$. By reversing the subdivisions of the edges in G^+ we obtain $G^\star$ and along with that we obtain a drawing $\Gamma^\star$ of $G^\star$ from Γ^+.

Let a, b, c be curves or corresponding edges. Then we write $\mathrm{cr}(a, b)$ for the number of crossings between a, b and set $\mathrm{cr}(a, b, c) = \mathrm{cr}(a, b) + \mathrm{cr}(a, c) + \mathrm{cr}(b, c)$. We consider any number of crossings only modulo 2. The following lemma is the radial equivalent

to Lemma 11 and constitutes our main tool for showing that edges in our drawing cross evenly.

Lemma 17. *Let C_1 and C_2 be distinct concentric circles and let a, b, c be radially monotone curves from C_1 to C_2 with pairwise distinct start- and endpoints that only intersect at a finite number of points. Then the start- and endpoints of a, b, c have the same order on C_1 and C_2 if and only if $\mathrm{cr}(a, b, c) \equiv 0$.*

Proof. If there are no crossings, then both sides hold. Hence, assume there is at least one crossing. By perturbations we achieve that every crossing has a unique distance to the center m. We order the crossings by distance to m. We add a concentric circle C_X^Y between any two consecutive crossings X, Y such that C_X^Y intersects a, b, c each once.

Then, the order of the intersection points of a, b, c must change between every two consecutive circles. Thus, that order is the same in C_1 and C_2, if and only if we added an odd number of circles. This in turn holds if and only if we have an even number of crossings. □

Lemma 18. *The drawing $\Gamma^\star$ is a Hanani-Tutte drawing of $G^\star$.*

Proof. We show that each pair of independent edges of $G^\star$ crosses evenly in $\Gamma^\star$. Of course it suffices to consider critical pairs of edges, since our drawing is radial by construction, and therefore non-critical independent edge pairs cannot cross. Every edge $(\alpha_i^+, \alpha_{i+1}^-)$ is subdivided into edges of the form $(\beta_j^+, \beta_{j+1}^-)$ and therefore it is not crossed.

Let e, f be two independent edges in $E(G^\star) \setminus (A^+ \times A^-)$ that are critical. Let a and b be the innermost and outermost level shared by e and f. We seek to use Lemma 17 to analyze the parity of the crossings between e and f. To this end, we construct a curve γ along the edges of the form $(\beta_j^+, \beta_{j+1}^-)$ as follows. For every level j we add a curve c_j between β_j^- and β_j^+ on the circle representing the level j (a point for $\beta_j^- = \beta_j^+$; chosen arbitrarily otherwise). The curve γ is the union of these curves c_j and the curves for the edges of the form $(\beta_j^+, \beta_{j+1}^-)$. Note that γ spans from the innermost level 1 to the outermost level $(k, 2n_k + 1)$ with endpoints $\mathrm{bot}(\alpha_1^+)$ and $\mathrm{top}(\alpha_k^-)$.

For any edge $g \in G^\star$, we denote its curve in $\Gamma^\star$ by $c(g)$. For any radial monotone curve c we denote its subcurve between level i and level j by c_i^j (using only one point on circle i and circle j each). We consider three curves $g' = \gamma_a^b, e' = c(e)_a^b, f' = c(f)_a^b$. Distinguish cases based on whether one of the edges e, f starts at the bottom end or ends at the top end of the reference edges on level a or b.

1. We have $e_a, f_a \neq \beta_a^+$ and $e_b, f_b \neq \beta_b^-$. Then for $a \leq j \leq b-1$ we have:

$$\begin{aligned}\varphi^+(\beta_{j+1}^+ e_{j+1} f_{j+1}) &\overset{\mathcal{C}}{\equiv} \varphi^+(\beta_{j+1}^- e_{j+1} f_{j+1}) + \varphi^+(\beta_{j+1}^- e_{j+1} \beta_{j+1}^+) + \varphi^+(\beta_{j+1}^- f_j \beta_{j+1}^+) \\ &\equiv \varphi^+(\beta_{j+1}^- e_{j+1} f_{j+1}) + \mathrm{cr}(e, c_{j+1}) + \mathrm{cr}(f, c_{j+1}) \\ &\overset{\mathcal{P}}{\equiv} \varphi^+(\beta_j^+ e_j f_j) \qquad + \mathrm{cr}(e, c_{j+1}) + \mathrm{cr}(f, c_{j+1})\end{aligned}$$

   This implies

$$\varphi^+(\beta_a^+ e_a f_a) + \varphi^+(\beta_b^- e_b f_b) \equiv \sum_{j=a}^{b-1} \mathrm{cr}(c_{j+1}, e) + \mathrm{cr}(c_{j+1}, f) \equiv \mathrm{cr}(e, \gamma) + \mathrm{cr}(f, \gamma) \quad ,$$

   where the third equation holds since edges of the form $(\beta_j^+, \beta_{j+1}^-)$ are not crossed. With Lemma 17 we conclude:

$$\begin{aligned}\mathrm{cr}(e, f) &\equiv \mathrm{cr}(e, f, \gamma) + \mathrm{cr}(e, \gamma) + \mathrm{cr}(f, \gamma) \\ &\equiv \varphi^+(\beta_a^+ e_a f_a) + \varphi^+(\beta_b^- e_b f_b) + \mathrm{cr}(e, \gamma) + \mathrm{cr}(f, \gamma) \equiv 0\end{aligned}$$

2. We do not have $e_a, f_a \neq \beta_a^+$ and $e_b, f_b \neq \beta_b^-$. For example, assume $e_a = \beta_a^+$; the other cases work analogously. We then have $\beta_a^+ = \mathrm{top}(\alpha_i^+)$. This means e originates from an edge in G. Since such edges do not cross middle levels, g' is a subcurve of an original edge ε_i. In particular, we have only three vertices per level between a and b that correspond to γ, e, f.

   Let $H \subseteq G^+$ be the subgraph induced by the vertices of $(\varepsilon_i)_a^b$, e_a^b, f_a^b. Then φ^+ satisfies all the constraints of $\mathcal{S}(H, V((\varepsilon_i)_a^b), V((\varepsilon_i)_a^b))$. However, each level of H contains only three vertices, and therefore the transitivity constraints are trivially satisfied, i.e., φ^+ satisfies all the constraints of $\mathcal{S}'(H, V((\varepsilon_i)_a^b), V((\varepsilon_i)_a^b))$. Thus, by Theorem 5, a drawing Γ_H of H according to φ^+ is planar. I.e., we have $\mathrm{cr}((\varepsilon_i)_a^b, e_a^b, f_a^b) = 0$ with regards to Γ_H. Let C_a, C_b be ε-close circles to levels a and b, respectively, that lie between levels a and b. With Lemma 17 we obtain that ε_i, e, f intersect C_a and C_b in the same order.

   Note that Γ^+ is drawn according to φ^+ in level a and in level b. We obtain that the curves for ε_i, e, f intersect C_a in the same order in Γ^+ and in Γ_H. The same holds for C_b. Hence, the curves intersect C_a and C_b in the same order in Γ^+. With Lemma 17 we have $\mathrm{cr}((\varepsilon_i)_a^b, e_a^b, f_a^b) \equiv 0$ with regards to Γ^+. Since γ is a subcurve of ε_i and thus not crossed in Γ^+, this yields $\mathrm{cr}(e_a^b, f_a^b) \equiv 0$ with regards to Γ^+.

Thus any two independent edges have an even number of crossings. □

4.4.5 From a Hanani-Tutte Drawing to a Satisfying Assignment

As in the level-planar case the converse also holds.

Lemma 19. *Let $G^\star$ be a level graph with reference sets A^+, A^- for G^+. If $G^\star$ admits a Hanani-Tutte drawing, then there exists a satisfying assignment φ of $\mathcal{S}(G^+, A^+, A^-)$.*

For the proof of the lemma we first construct the assignment φ and then show that it satisfies $\mathcal{S}(G^+, A^+, A^-)$. Let $G^\star = (V, E)$ have k levels and a radial Hanani-Tutte drawing Γ. Let G^+ be the level graph obtained by subdividing all edges such that G^+ is proper. For three distinct vertices x, y, z on level j with $x = \alpha_j^-$ or $x = \alpha_j^+$, we set $\psi(xyz) = 0$ if and only if x, y, z appear clockwise on the circle representing level j. If two edges e, f are adjacent in a vertex v with $\mathrm{cr}(e, f) \equiv 1$, then we say they have a *phantom crossing* at v. We denote by $\mathrm{cr}^\star$ the function counting crossings and additionally adding 1 if there is a phantom crossing. With the phantom crossings, any two edges in G cross an even number of times, even if they are not independent. For any edge $e = uv$ in G^+ between level j and $j+1$ with $u = \alpha_j^+$ or $v = \alpha_{j+1}^-$ we set $\psi(l(e)) = 0$ if and only if e is locally left of ε_j. We further set $\varphi(l(e)) \equiv \psi(l(e)) + \mathrm{cr}(e, \varepsilon_j)$ to switch that value in case of a phantom crossing.

Let $v \in V(G^+)$ be a vertex. If v is a subdivision vertex of an edge e, then we set $e(v) = e$. Otherwise we set $e(v) = \emptyset$. We say $\emptyset$ has no crossings with anything but stretches over all levels. This helps to avoid case distinctions. For an edge $e = (u, v)$ of G we write e^j for the subdivision path of e that starts at u and ends in level j. We set $\emptyset^j = \emptyset$. Let $x, y, z \in V_j(G^+)$ be disjoint with $x = \alpha_j^-$ or $x = \alpha_j^+$. We set

$$\varphi(xyz) \equiv \psi(xyz) + \mathrm{cr}^\star(e(x)^j, e(y)^j, e(z)^j)\,.$$

We thereby switch the order of x, y, z if and only if at least two of them are subdivision vertices and the corresponding edges cross an odd number of times up to level j. This finishes the construction of φ.

Lemma 20. *The assignment φ satisfies $\mathcal{S}(G^+, A^+, A^-)$.*

Proof. First note that ψ satisfies $\mathcal{L}(\alpha_j^+)$, $\mathcal{L}(\alpha_j^-)$ and $\mathcal{C}(\alpha_j^+, \alpha_j^-)$ for $1 \le j \le k$.

φ satisfies $\mathcal{L}(\alpha_j^+)$, $\mathcal{L}(\alpha_j^-)$. For three distinct vertices $x, y, z \in V_j(G^+)$ we have by definition of φ that $\varphi(xyz) + \varphi(xzy) \equiv \psi(xyz) + \psi(xzy)$. Since ψ satisfies Eq. (4.4), so does φ.

φ satisfies $\mathcal{C}(\alpha_j^+, \alpha_j^-)$. For Eq. (4.10) let $1 \leq j \leq k$ such that $\alpha_j^- \neq \alpha_j^+$. Let u, v be distinct vertices in $V_j(G^+) \setminus \{\alpha_j^-, \alpha_j^+\}$ By definition of φ we have that

$$\begin{aligned}
&(\varphi(\alpha_j^- uv) + \varphi(\alpha_j^+ uv)) + (\varphi(\alpha_j^- u\alpha_j^+) + \varphi(\alpha_j^- v\alpha_j^+)) \\
\equiv\ &(\psi(\alpha_j^- uv) + \psi(\alpha_j^+ uv)) + (\psi(\alpha_j^- u\alpha_j^+) + \psi(\alpha_j^- v\alpha_j^+)) \\
&+ 2\big(\operatorname{cr}^\star(e(\alpha_j^-), u) + \operatorname{cr}^\star(e(\alpha_j^-), v) + \operatorname{cr}^\star(e(\alpha_j^+), u) + \operatorname{cr}^\star(e(\alpha_j^+), v) \\
&+ \operatorname{cr}^\star(u, v) + \operatorname{cr}^\star(e(\alpha_j^-), e(\alpha_j^+))\big) \\
\equiv\ &(\psi(\alpha_j^- uv) + \psi(\alpha_j^+ uv)) + (\psi(\alpha_j^- u\alpha_j^+) + \psi(\alpha_j^- v\alpha_j^+))
\end{aligned}$$

Therefore, Eq. (4.10) holds.

For Eq. (4.11) let $u \in V_j(G^+) \setminus \{\alpha_j^+, \alpha_j^-\}$. With the definition of φ we obtain

$$\varphi(\alpha_j^- u\alpha_j^+) + \varphi(\alpha_j^+ \alpha_j^- u) \equiv \psi(\alpha_j^- u\alpha_j^+) + \psi(\alpha_j^+ \alpha_j^- u)\,.$$

Thereby Eq. (4.11) holds.
It remains to argue that φ satisfies $\mathcal{P}(\varepsilon_j)$ for all levels.

φ satisfies Eq. (4.6) of $\mathcal{P}(\varepsilon_j)$. Let $1 \leq j \leq k-1$ and let $(u, u'), (v, v') \in E_j(G^+)$ be independent. Then we argue that

$$\begin{aligned}
&\varphi(\alpha_j^+ uv) + \varphi(\alpha_{j+1}^- u'v') \\
\equiv\ &\psi(\alpha_j^+ uv) + \psi(\alpha_{j+1}^- u'v') + \operatorname{cr}^\star((u, u'), (v, v'), \varepsilon_j)
\end{aligned} \tag{4.12}$$

This means we change the order of the ends of $(u, u'), (v, v'), \varepsilon_j$ on exactly one of the levels j and $j+1$ if and only if they cross an odd number of times. With Lemma 17 we then obtain that Eq. (4.6) holds for $(u, u'), (v, v')$.

With the definition of φ it suffices to show the following three equations

$$\operatorname{cr}^\star(e(\alpha_{j+1}^-)^{j+1}, e(u')^{j+1}) + \operatorname{cr}^\star(e(\alpha_j^+)^j, e(u)^j) \equiv \operatorname{cr}^\star(\varepsilon_j, (u, u')) \tag{4.13}$$

$$\operatorname{cr}^\star(e(\alpha_{j+1}^-)^{j+1}, e(v')^{j+1}) + \operatorname{cr}^\star(e(\alpha_j^+)^j, e(v)^j) \equiv \operatorname{cr}^\star(\varepsilon_j, (v, v')) \tag{4.14}$$

$$\operatorname{cr}^\star(e(u')^{j+1}, e(v')^{j+1}) \quad + \operatorname{cr}^\star(e(u)^j, e(v)^j) \quad \equiv \operatorname{cr}^\star((u, u'), (v, v')) \tag{4.15}$$

Eq. (4.15), (4.13) and (4.14) can be shown analogously, noting that $\varepsilon_j = (\alpha_j^+, \alpha_{j+1}^-)$ is independent from $(u, u'), (v, v')$. We distinguish cases where u, u', v, v' are vertices of different kinds.

1. u, u' or v, v' are original vertices. Then $e(u) = e(u') = \emptyset$ or $e(v) = e(v') = \emptyset$. Then the left side equals 0. The right side also equals 0, since uu' or vv' is an edge of $G^\star$.

2. u, u' or v, v' are subdivision vertices. Without loss of generality assume u, u' are subdivision vertices. Then we obtain $e(u) = e(u')$ and the left side reduces to $\mathrm{cr}^\star(e(u)^{j+1}, e(v')^{j+1}) + \mathrm{cr}^\star(e(u)^j, e(v)^j)$.
   a) The case where v, v' are original vertices is already covered.
   b) If v, v' are subdivision vertices, we continue with
   $$\begin{aligned} & pcr(e(u)^{j+1}, e(v')^{j+1}) + \mathrm{cr}^\star(e(u)^j, e(v)^j) \\ \equiv\ & pcr(e(u)^{j+1}, e(v)^{j+1}) + \mathrm{cr}^\star(e(u)^j, e(v)^j) \\ \equiv\ & pcr(e(u)^{j+1_j}, e(v)_j^{j+1}) \\ \equiv\ & pcr((u, u'), (v, v')) \end{aligned}$$
   c) If v is an original vertex an v' is a subdivision vertex, we get
   $$\begin{aligned} & \mathrm{cr}^\star(e(u)^{j+1}, e(v')^{j+1}) + \mathrm{cr}^\star(e(u)^j, e(v)^j) \\ \equiv\ & \mathrm{cr}^\star(e(u)^{j+1}, vv') + 0 \\ \equiv\ & \mathrm{cr}^\star(uu', vv') \end{aligned}$$
   d) If v is a subdivision vertex and v' is an original vertex, we get
   $$\begin{aligned} & \mathrm{cr}^\star(e(u)^{j+1}, e(v')^{j+1}) + \mathrm{cr}^\star(e(u)^j, e(v)^j) \\ \equiv\ & 0 + \mathrm{cr}^\star(e(u)^j, e(v)^j) \\ \equiv\ & \mathrm{cr}^\star(e(u), e(v)) \\ \equiv\ & 0 \end{aligned}$$
3. u, v are original vertices and u', v' are subdivision vertices. Then we have that $e(u) = e(v) = \emptyset$ and $e(u')^{j+1} = uu'$ and $e(v')^{j+1} = vv'$ and the equivalence holds.
4. u', v' are original vertices and u, v are subdivision vertices. Noting
   $$\mathrm{cr}^\star(e(u)^j, e(v)^j) + \mathrm{cr}^\star(e(u)_j^k, e(v)_j^k) = \mathrm{cr}^\star(e(u), e(v)) \equiv 0,$$
   we can argue analogously.
5. u, v' are original vertices and u', v are subdivision vertices. Then we have that $e(u) = e(v') = \emptyset$, and the left side equals 0. Furthermore, observe that it is $\mathrm{cr}^\star((u, u'), (v, v')) = \mathrm{cr}^\star(e(u'), e(v)) \equiv 0$, i.e., both sides are even.
6. u', v are original vertices and u, v' are subdivision vertices. Then we can argue analogously to the previous case.

Hence, we have that φ satisfies Eq. (4.6).

φ satisfies Eq. (4.7) of $\mathcal{P}(\varepsilon_j)$. Let $e = (u, u') \in E_j^+(G^+)$ and let $f = (v, v') \in E_j^-(G^+)$. Then we adjust the drawing by perturbing possible phantom crossings of e, f with ε_j in α_j^+ and α_{j+1}^- to the space between circle j and circle $j+1$. Thereby, the states of e, f being locally left of ε_j change if and only if they have a phantom crossing. This is the case if and only if ψ and φ differ for l of the corresponding edge. I.e., $\varphi(l(e))$ and $\varphi(l(f))$ describe, whether e, f are locally left of ε_j. Consider the closed curve c that is the union of $c(e), c(\varepsilon_j)$ and the part d of circle $j+1$ between u' and α_{j+1}^-, such that circle j is on the outside of c. Then the edge f is outside of c at circle j. Note that ε_j has to be an original edge. Since f crosses e, ε_j an even number of times each (as their corresponding original edges cross only between level j and $j+1$), and it does not cross d, edge f has to be outside of c at ε-close distance to circle $j+1$. We thereby obtain $\varphi(l(e)) \equiv \varphi(l(f)) + 1$.

φ satisfies Eq. (4.8) of $\mathcal{P}(\varepsilon_j)$. Let $(u, u') \in E_j^+$ and let $(v, v') \in E_j$ be independent. Note that $e(\alpha_{j+1}^-)^{j+1} = \varepsilon_j$ and that $e(u')^{j+1} = (u, u')$ and likewise for v' since $\alpha_{j+1}^- = u'$ must be an original vertex. Let circle a be a circle that is ε-close outside of circle j.

Then we have

$$
\begin{aligned}
&\varphi(\alpha_{j+1}^- v'u') - \psi(\alpha_{j+1}^- v'u') \\
&\equiv \mathrm{cr}^\star(e(v')^{j+1}, e(u')^{j+1}) + \mathrm{cr}^\star(e(\alpha_{j+1}^-)^{j+1}, e(v')) + \mathrm{cr}^\star(e(\alpha_{j+1}^-)^{j+1}, e(u')) \\
&\equiv \mathrm{cr}^\star((v, v'), (u, u')) + \mathrm{cr}^\star(\varepsilon_j, (v, v')) + \mathrm{cr}^\star(\varepsilon_j, (u, u')) \\
&\equiv \mathrm{cr}^\star(\varepsilon_j, (v, v'), (u, u')) \\
&\equiv \mathrm{cr}(\varepsilon_j, (v, v'), (u, u')) + \mathrm{cr}^\star((u, u')^a, (v, v')^a, \varepsilon_j^a) \\
&\equiv (\psi(l(u, u')) - \psi(\alpha_{j+1}^- v'u')) + \mathrm{cr}^\star((u, u')^a, (v, v')^a, \varepsilon_j^a) \\
&\equiv \varphi(l(u, u')) - \psi(\alpha_{j+1}^- v'u') \,.
\end{aligned}
$$

We obtain $\varphi(l(u, u')) = \varphi(\alpha_{j+1}^- v'u')$.

φ satisfies Eq. (4.9) of $\mathcal{P}(\varepsilon_j)$. Can be argued similarly to the previous case. □

With the Lemmas 18, 19 we obtain the following equivalence.

Theorem 6. *Let G be a level graph with reference sets A^+, A^-. Let B^+, B^- be corresponding reference sets of G^+. Then*

This yields the equivalence of the two radial level-planarity characterizations as follows. Let G be a proper level graph with reference sets A^+, A^-. Assume all level graphs with a Hanani-Tutte radial level drawing are radial level-planar. Then for

every level graph G with reference sets A^+, A^- where $\mathcal{S}(\overline{G}, A^+, A^-)$ is satisfiable, constraint system $\mathcal{S}(G^+, B^+, B^-)$ is satisfiable by Lemma 15. Then by Theorem 6 there is a Hanani-Tutte drawing of $G^\star$ and thus $G^\star$ is radial level-planar. Since $G^\star$ is radial level-planarity equivalent to G, G is also radial level-planar. On the other hand, Assume each proper level graph G with reference sets A^+, A^- where $\mathcal{S}(\overline{G}, A^+, A^-)$ is satisfiable is radial level-planar. Then for every level graph G with reference sets A^+, A^- for G^+ and a Hanani-Tutte radial level drawing, we trivially obtain a Hanani-Tutte radial level drawing of $G^\star$. Then by Theorem 6 $\mathcal{S}(G^+, B^+, B^-)$ is satisfiable. Hence, G^+ is radial level-planar, and since G^+ and G are radial level-planarity equivalent, G is also radial level-planar. Note that we did not actually use the Hanani-Tutte result for the radial case [FPS16].

Corollary 2. *The radial level-planar graphs are exactly the level graphs with a Hanani-Tutte radial level drawing if and only if they are exactly level graphs G where $\mathcal{S}(G)$ is satisfiable.*

With the known Hanani-Tutte result for the radial case [FPS16], we conclude that radial level-planarity can be characterized in the spirit of Randerath et al. [Ran+01].

Corollary 3. *Let G be a proper level graph with reference sets A^+, A^-. Then*

4.5 Conclusion

We have established an equivalence of two results on level-planarity that have so far been considered as independent. The novel connection has further led us to a new testing algorithm for radial level planarity. Can similar results be achieved for level-planarity on a rolling cylinder or on a torus [Ang+20]?

5 Level Planarity Testing: A Unified Approach

The problems of testing graphs for the (radial) level planarity and, if possible, finding a corresponding drawing, were studied in a number of papers across many years, resulting in linear-time algorithms. However, these algorithms are notoriously complex to understand. Moreover, we believe that the algorithm that treats the radial setting is incomplete. In this chapter, we describe a self-contained linear-time algorithm that tests a graph for radial level planarity, and, if the result is positive, outputs a radial level planar embedding of it.

This chapter is based on unpublished joint work with Ignaz Rutter.

5.1 Introduction

Planarity is central topic in topological and algorithmic graph theory. Algorithmically, the problem has been settled by Hopcroft and Tarjan [HT74] who gave the first planarity testing algorithm that runs in optimal linear time. Nowadays several different planarity testing algorithms are known [Pat13], and various implementations are available in software packages such as OGDF [Chi+13]. By contrast, the concept of *(radial) level planarity* for directed graphs is much less well-understood. A level graph is a graph where each vertex v is assigned an integer level $\ell(v)$ and for each (directed) edge uv it is $\ell(u) < \ell(v)$. A *(radial) level-planar drawing* is a planar drawing where each vertex v lies on the horizontal line $y = \ell(v)$ (on the circle with radius $\ell(v)$ centered at the origin) and each edge uv is represented by a curve that is y-monotone

(radially monotone). Alternatively (but equivalently), radial level-planar drawings can be defined on the standing cylinder.

Jünger and Leipert [JLM98, JL02] give linear-time algorithms to test the existence of and compute level-planar drawings. Bachmaier, Brandenburg and Forster extend these algorithms to the radial case [BBF05]. The core of the approach lies in the fact that the possible orders of vertices on each level can be represented by (some variant of) a PQ-tree; a data structure that also features prominently in other planarity algorithms.

The idea of using PQ-trees for this task dates back to Di Battista and Nardelli who employed them to solve the case of single-source level graphs [DN88]. The generate the corresponding PQ-tree by an incremental algorithm that constructs the graph level by level and use induction to prove that in each step the possible vertex orders can be represented by a PQ-tree. It is straightforward to extend their approach to radial level-planar graphs with a single source. The difficulty starts when considering graphs with multiple sources. In this case, we grow one PQ-tree per source of the graph; however subgraphs that have been disjoint on the upper levels may join into one connected component in later levels, making it necessary to merge multiple PQ-trees into one. This approach was pioneered by Heath and Pemmaraju [HP95], who used it to design an $O(n^2)$-time algorithm for testing level planarity; and $O(n^3)$ time for computing a corresponding drawing. However, Jünger, Leipert and Mutzel [JLM97] point out several flaws in their algorithm that leads their algorithm to reject graphs even though they are level-planar. Following up on this, Jünger, Mutzel and Leipert [JLM98] correct these mistakes and describe a linear-time algorithm for testing level planarity. Jünger and Leipert [JL02] extend the algorithm to also produce a level-planar embedding in the same running time, if it exists. Bachmaier, Brandenburg and Forster then further extend this algorithm to the radial level-planar case [BBF05]. This is, however, not the end of the story. The level-planarity testing algorithm [JLM98] refers to a full version for several parts of the proof. However, such a full version has, to the best of our knowledge never appeared. A journal version of the linear-time embedding algorithm for level graphs is available [JL02], however the algorithm is long and complicated, and the correctness proof is on an entirely technical level along the steps of the algorithm that gives little structural insight. Even the overall proof strategy remains opaque. The extensions of Bachmaier and Brandenburg [BBF05] defer large parts of their correctness arguments as being analogous to the level-planar case. Altogether this makes for more than 150 pages of literature towards such linear-time algorithms, however a convincing proof of correctness is, in our opinion, currently not available. Though Leipert [Lei98] and Bachmaier [Bac04] describe prototypical implementations of their algorithms in their dissertations, none of these has been experimentally evaluated in any way. We are also not aware of any publicly available implementations.

For these reasons, simpler but slower algorithms have been proposed even long after the publication of the linear-time algorithm. Randerath et al. [Ran+01] show how to model level planarity as a 2-SAT formula of quadratic size, Fulek et al. [FPSŠ13] give a Hanani-Tutte characterization, which also leads to an $O(n^2)$-time algorithm. Both of these approaches have also been generalized to the radial case [FPS17, FPS16, BRS18], again implying running time $O(n^2)$.

Contribution. In this chapter, we give a new algorithm for radial level planarity testing that runs in linear time. Similar to the previous approaches, it is based on based (augmented) PQ-trees. Our algorithm combines a preprocessing that reduces the number of special cases to consider with a small number of simple key steps. In contrast to Jünger and Leipert [JL02] and Bachmaier et al. [BBF05] extending our algorithm to also compute a corresponding (radial) level-planar embedding is relatively simple and does not require a large extra effort. The main reason for this is that, instead of eliminating sinks by adding additional edges, we focus on the elimination of sources.

5.2 Preliminaries

We define a couple of basic operations for level-planar drawings. Let G be a level graph and let Γ be a level-planar drawing of G. Let u and w denote (not necessarily distinct) vertices on the same level of G and let $\varphi(u)$ and $\varphi(v)$ denote their angular coordinates in Γ, respectively. Define the *(counter-clockwise) angular distance* $\Delta(u,v)$ from u to v as the smallest value in $(0, 2\pi]$ so that $\varphi(u)+\Delta(u,v) = \varphi(v) \mod 2\pi$. Note that $u = v$ gives $\Delta(u,v) = 2\pi$ and that for $u \neq v$ it is generally $\Delta(u,v) \neq \Delta(v,u)$. A new vertex v can be *interspersed* between u and w by assigning the radial coordinate $\ell(u) = \ell(v)$ and the angular coordinate $(\varphi(u) + \Delta(u,v)/2) \bmod 2\pi$ to it.

Similarly, an edge can be interspersed. Let $e = (u,u')$ and $g = (w,w')$ denote (not necessarily distinct) edges of G between levels $r_1 := \max(\ell(u), \ell(w))$ and $r_2 := \min(\ell(u'), \ell(w'))$. Let φ_e and φ_g denote the drawings of e and g in Γ, respectively. For $r \in (r_1, r_2)$ define the *(counter-clockwise) angular distance* $\Delta(e,g,r)$ from e to g as the smallest value in $(0, 2\pi]$ so that $\varphi_e(r) + \Delta(e,g,r) = \varphi_g(r) \mod 2\pi$. Further, define $\Delta(e,g,r_1) = \lim_{r \searrow r_1} \Delta(e,g,r)$ and $\Delta(e,g,r_2) = \lim_{r \nearrow r_2} \Delta(e,g,r)$. These limits are well-defined because φ_e and φ_f are continuous on the interval $[r_1, r_2]$. Then $\Delta(e,g) : r \mapsto \Delta(e,g,r)$ is continuous on the interval $[r_1, r_2]$, too. Choose $\lambda(r_1)$ from the interval $[0,1]$ with $\varphi(u) + \lambda(r_1)\Delta(e,g,r_1) = \varphi(v) \mod 2\pi$. This choice is unique unless $\varphi(u) = \varphi(v) = \varphi(w)$, i.e., $u = v = w$. In that case let $\lambda(r_1) := 0$. Similarly, choose $\lambda(r_2) \in [0,1]$ with $\varphi(u') + \lambda(r_2)\Delta(e,g,r_2) = \varphi(v') \mod 2\pi$. Again, that choice is unique unless $u' = v' = w'$, in which case we choose $\lambda(r_2) := 0$.

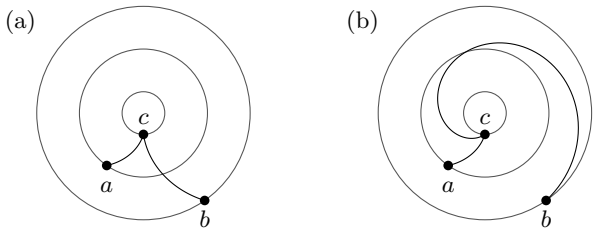


Figure 5.1: The drawings in (a) and (b) induce the same combinatorial embedding and the same cyclic vertex order on both levels, but their level-planar embedding differs: the linearized order of the edges around c is $(c, a), (c, b)$ in (a), and $(c, b), (c, a)$ in (b). We will require our PC-trees to have a C-node as their root, which helps us distinguish these embeddings.

For $r \in (r_1, r_2)$, let $\lambda(r)$ be the linear interpolation between $\lambda(r_1)$ and $\lambda(r_2)$. Hence, λ is continuous on the interval $[r_1, r_2]$. *Intersperse* $f = (v, v')$ between e and g by drawing f as $\varphi_f(r) \coloneqq \varphi_u(r) + \lambda(r)\Delta(e, g, r) \bmod 2\pi$, which is radially continuous. Consider a vertex v of G. There exists a cyclic order $\sigma = (e_1, e_2, \ldots, e_n, e_{n+1} = e_1)$ of the n child edges of v so that in Γ, each circle with radius $r \in (\ell(v), \ell(v)+1)$ intersects the child edges of v in (counter-clockwise) order σ. It is $\sum_{i=1}^{n} \Delta(e_i, e_{i+1}, r) = 2\pi$ and because Jordan arcs are (radially) continuous, there exists for each $\varepsilon > 0$ a value $\delta > 0$ such that for all r with $r < \ell(v) + \delta$ it is $\varphi(v) - \varepsilon < \varphi(e_i) < \varphi(v) + \varepsilon \mod 2\pi$ for each child edge e_i of v. This means that there exists one pair e_i, e_{i+1} of edges so that $\Delta(e_i, e_{i+1}, \ell(v)) = 2\pi$ whereas for every other pair of edges e_j, e_{j+1} with $j \neq i$ it is $\Delta(e_j, e_{j+1}, \ell(v)) = 0$. The *linearized order* of child edges of v in Γ is obtained by splitting σ between e_i and e_{i+1} so that e_{i+1} is the first edge and e_i is the last edge. The linearized order of parent edges of v in Γ is defined similarly.

Recall that a *level-planar (combinatorial) embedding* of G associates with each vertex of G a linearized order of its child edges and a linearized order of its parent edges, and that this notion is different from that of planar combinatorial embeddings. For an example, see Figure 5.1, where the two level-planar drawings induce different level-planar (combinatorial) embeddings, but the same planar combinatorial embedding. Also, recall that a level-planar embedding $\mathcal{E}$ of a single-source k-level graph induces counter-clockwise cyclic orders of the vertices on each level. Call the counter-clockwise order of level-k vertices the *frontier* of $\mathcal{E}$.

5.3 Regularization

In this section, we preprocess the input graph G and simplify its structure. This helps to reduce the number of case distinctions that we will have to make in Section 5.11. We start out by defining a so-called star form $G^\star$ of G, which has the property that it only changes in small deltas when processing it level by level. We then go on to define $G^\times$, which simplifies how components that are disjoint on previous levels grow together as the graph is processed level by level.

5.3.1 Star Form

Let $G = (V, E)$ together with ℓ be a k-level graph on n vertices. Assume without loss of generality that there exists exactly one vertex $\varsigma \in V$ with $\ell(\varsigma) = 1$. Define a $4n$-level graph $G^\star$ on $2n$ vertices, called the *star form* of G, as follows. Consider each level i of G with $1 \leq i \leq k$ in this order and let $v_1, v_2, \ldots, v_n$ denote the vertices on level i. In $G^\star$, append $4n$ levels $i_1, i_2, \ldots, i_{4n}$. For $1 \leq j \leq n$ create two vertices v'_j, v''_j with $\ell(v'_j) = i_{2j-1}$ and $\ell(v''_j) = i_{2(j+n)-1}$ connected by an edge (v'_j, v''_j). Refer to (v'_j, v''_j) as the *stretch edge* of v_j. For each edge (u, v) of G add the edge (u'', v') to $G^\star$. Define G^+ as the graph obtained from $G^\star$ by subdividing each edge until all edges are proper and refer to it as the *plus form*. The size of $G^\star$ is linear in the size of G, the size of G^+ may be quadratic in the size of G. We will use the plus form in our proofs, but our algorithm operates on the star form.

Now let G be a k-level graph in plus form. For $1 \leq i \leq k$ let $G[i]$ denote the restriction of G to the levels $\{1, \ldots, i\}$. Consider $G[i]$. Call a vertex on level $i-1$ *unpaired* if it does not have exactly one neighbor on level i. Likewise, call a vertex on level i *unpaired* if it does not have exactly one neighbor on level $i-1$. Because G is in plus form $G[i]$ has at most one unpaired vertex. If there is an unpaired vertex there are four cases of how $G[i]$ arises from $G[i-1]$.

1. The unpaired vertex has no neighbor.
   a) The unpaired vertex u lies on level $i-1$. Then $G[i]$ arises from $G[i-1]$ by *pruning* u.
   b) The unpaired vertex v lies on level i. Then $G[i]$ arises from $G[i-1]$ by *introducing* v.
2. Otherwise the unpaired vertex has at least two neighbors.
   a) The unpaired vertex u lies on level $i-1$. Then $G[i]$ arises from $G[i-1]$ by *growing* u with its children $v_1, v_2, \ldots, v_n$ on level i.
   b) The unpaired vertex v lies on level i. Then $G[i]$ arises from $G[i-1]$ by *bundling* the ancestors $u_1, u_2, \ldots, u_n$ of v.

Call the unpaired vertex and its neighbors on levels $i-1$ or i *pertinent*. In the following we restrict ourselves to graphs obtained by a sequence of these four basic operations. If G is a single-source level graph each $G[i]$ is connected and hence, when $G[i]$ is obtained from $G[i-1]$ by bundling, the ancestors belong to the same connected component.

5.3.2 Redrawing

In this section we further restrict the star form $G^\star$. For convenience we rather work with the plus-form G^+. Let v be a vertex of G^+ so that $G^+[\ell(v)-1]$ has more connected components than $G^+[\ell(v)]$. Let $H_1,\dots,H_k$ be the connected components of $G^+[\ell(v)-1]$ that have v as a neighbor. We say that v *joins* $H_1,\dots,H_k$. The component H_i is *v-singular*, if v is the only vertex adjacent to H_i that lies on level i; otherwise H_i is called *non-singular*.

We aim to further modify $G^\star$ in such a way that for each vertex v joining components $H_1,\dots,H_k$ we have that if $k \geq 3$, then they are all v-singular. That is, each vertex either joins only two components, or all the joined components are v-singular. The key observation is the following technical lemma, which allows us to make further assumptions about the linearized edge orders of the level-planar embeddings of $G^\star$.

Lemma 21. *Let v be a vertex of G^+ that joins more than two connected components and let H be the non-singular connected component joined by v that maximizes $\ell(H)$. Let I denote the incoming edges of v and let $I' \subseteq I$ be the subset of edges that are incident to H. If G^+ admits a level-planar embedding $\mathcal{E}$, then there also exists a level-planar embedding $\mathcal{E}'$ of G^+ that differs from $\mathcal{E}$ at most in the linearized order of the incoming edges of v so that $I \setminus I'$ is consecutive in $\mathcal{E}'$.*

Proof. Note that in this situation, v is a cut-vertex of $G^+[\ell(v)]$, and each split component of v corresponds to a component that is joined by v. Let C be such a component. Consider the level-planar embedding of C induced by $\mathcal{E}$. Let D be the disk with radius i centered at the origin. Removing the drawing of C from D yields several connected regions, which we call the *faces* of C. A face of C is a *boundary face* if it contains a point of the boundary of D; the other faces are called *internal*. The unique face that contains the origin is called *central face*. The component C is a *ring* if its central face is not a boundary face, and it is a *non-ring* otherwise.

The level-planarity of $\mathcal{E}$ gives the following properties.

1. If C, C' are both rings and $\ell(C) < \ell(C')$, then C is embedded in the central face of C'.
2. If C is contained in a non-boundary face of C', then C is v-singular.

3. If C is a non-singular component and C' is not contained in the central face of C, then C' is a non-ring.

4. For each component C there is a vertex p of C incident to the central face with $\ell(p) = \ell(C)$.

5. If C' is embedded inside a non-central face f of a component C, then there exists a vertex p of C incident to f with $\ell(p) < \ell(C')$; in particular $\ell(C) < \ell(C')$.

To modify the embedding so that $I \setminus I'$ becomes consecutive, we apply the following *redrawing operation*. Let S be a v-singular component incident joined by v that is not a ring. Since S is v-singular, it has a single boundary face, which is also central. By Property (iv) the unique vertex $\min S$ is incident to this face. Since S is v-singular, v is the only vertex of S on its level. Therefore, we can draw a left curve γ_ℓ and a right curve γ_r from $\min V$ to v so that each of them visits the levels of v and $\ell(S)$ only at its endpoints. Using Lemma 6 from Chapter 3, this implies that there exists a drawing of G^+ with the same embedding where we can additionally insert edges e_ℓ and e_r from $\min S$ to v that are embedded to the left and right of the leftmost and rightmost outgoing edge of $\min S$, respectively. This allows us to redraw S and everything embedded in its inner faces inside an arbitrary face f incident to v, whenever it is possibly to embed a single edge (p, v) with level $\ell(p) < \ell(S)$ in f. Again using the result of Da Lozzo et al., to embed such an edge, it suffices to have a vertex p with $\ell(p) < \ell(S)$ incident to f.

We are now ready to modify embedding $\mathcal{E}$. Recall that H is the non-singular component that maximizes $\ell(H)$. We distinguish cases based on whether H is a ring or not. If it is not, we further dinstinguish cases based on whether v is incident to the central face of H or not.

First consider the case that H is a ring. Let C be another component joined by v. If C is non-singular, Property (ii) gives that it is embedded in a boundary face of H, and hence not in the central face of H. Then Property (v) gives $\ell(C) > \ell(H)$, contradicting the choice of H. Thus all other components joined by v must be v-singular. If none of these v-singular components is a ring, let S be the v-singular component that minimizes $\ell(V)$ and let f be the face of H that contains $\ell(V)$. Using the above redrawing technique, we can then redraw all v-singular components embedded in other faces next to S. Hence assume that there exists a v-singular ring and let R be v-singular ring that maximizes $\ell(R)$. By Property (i), R is embedded inside the central face f of H. Moreover all other rings are embedded in the central face of R, and thus also inside f. Let S be a v-singular connected component that is not embedded inside f, which hence cannot be a ring. Then S lies in the (unique) boundary face of R, and thus not in its central face. By Property (v) R has a vertex p incident to its boundary with $\ell(p) < \ell(S)$. This allows us to redraw S inside inside f. By treating

all v-singular components embedded outside f in this way, we eventually arrive at an embedding where all other components joined by v are embedded inside f.

Second, consider the case that H is not a ring and that v is not incident to the central face of H. If C is another component joined by v, then it cannot be embedded inside the central face of H (as v is not incident to it), and Property (v) gives that $\ell(C) < \ell(H)$. By the choice of H, it follows that C is v-singular. Moreover, Property (iii) gives that C is not a ring. Thus all remaining components are v-singular non-rings. Let S be the one that minimizes $\ell(S)$ and let f be the face of H that contains S. Using the above redrawing technique, we can redraw all v-singular non-rings embedded outside f inside f.

Third, consider the case that H is not a ring but v is incident to the central face of H. Observe that v is incident to two boundary faces f and f', one of which, say f is also the central face of H. Let C be another component joined by v that is not embedded inside f. Property (v) gives $\ell(H) < \ell(C)$. By the choice of H, this implies that C is v-singular. Property (iii) further gives that C is not a ring. Finally, Property (iv) gives that H contains a vertex p incident to f with $\ell(p) = \ell(H) < \ell(C)$. We can thus redraw C inside f.

In all cases, we manage to redraw the graph changing only the order of the incoming edges of v so that all components joined by v lie in the same face of H, i.e., $I \setminus I'$ is consecutive. □

Observe that embeddings of $G^\star$ correspond bijectively to embeddings of G^+, and therefore Lemma 21 also applies to $G^\star$. Consider now a vertex v of $G^\star$ that joins $k \geq 3$ components of which at least one is not v-singular and let H be the non-singular component joined by v that maximizes $\ell(H)$. Consider the graph obtained by introducing an additional level just below $\ell(v)$, introducing a vertex v' on that level along with the edge (v, v'), and finally reconnecting all edges of H incident to v to v'. Lemma 21 gives that the modified graph is level-planar if and only if $G^\star$ is level-planar. Observe that by construction v' joins only two connected components, namely H and one v'-singular component. If $H_1, \ldots, H_k$ are the non-singular components joined by v ordered such that $\ell(H_i) < \ell(H_j)$ whenever $i < j$, then repeatedly applying this construction to v, we split v into a path $v, v_1, \ldots, v_k$ such that the original vertex v joins only v-singular components, and v_i joins the non-v-singular component H_i with a v-singular component. Lemma 21 guarantees that the resulting graph G' is level-planar if and only if $G^\star$ is level-planar. Moreover, Observe that the resulting graph still satisfies the property that each level contains exactly one vertex, which has either in- or outdegree 1. We can thus apply the same construction independently to all vertices of $G^\star$ that join several components. We call the resulting graph $G^\times$.

Theorem 7. *Given a graph level graph G we can compute in linear time a level graph $G^\times$ that is level-planar if and only if G is level-planar such that $G^\times$ has the property that it*

is both in star-form and each vertex v that joins more than two connected components joins only v-singular connected components.

Proof. We have already shown the existence of $G^\times$ and the fact that is level-planar if and only if G is. It remains to show that $G^\times$ can be constructed in linear time. First, compute the star form of G, which takes linear time. For brevity, let G denote this star form from now on. Then, process G level by level from the bottom up. At each level i for $1 \leq i \leq k$, where k is the number of levels of G, we want to know for each vertex on level i and its incoming edges to which connected component of $G[i]$ it belongs. To this end, we can use a union find data structure. When going from level $i-1$ to level i, consider each vertex v on level i and unify the sets belonging to all of its ancestors in $G[i]$. Of course, all incoming edges of v belong to the same connected component as v itself. The *source* of an edge (u, v) is the source of the component of $G[\ell(v)]$ on the smallest level (which is unique because G is in star form). Because the structure of union and find operations is completely determined by the fixed structure of G, we can use the linear-time algorithm of Gabow and Tarjan [GT85] to efficiently compute the source for every edge of G. In this way it is also easy to determine for each component whether it is v-singular or non-singular. Next, sort the edges of G with respect to the level of their sources. Because $|E|, k \in O(|V|)$, this can be achieved in linear time using counting sort. Sweep through the sorted list once and append each edge (u, v) that is encountered to a per-vertex list maintained at v. This produces at v a list of its incoming edges that is sorted with respect to the level of their sources. At this point, we have all the information we need to efficiently execute the procedure that splits vertices into paths described above to compute $G^\times$. □

5.4 PC-Trees

We start by altering our definition of PC-trees from Chapter 2 in requiring that the root of a PC-tree must be a C-node. Let $G = (V, E)$ be a k-level graph. Next, we annotate PC-trees with vertices of G as follows. Each P-node μ of T is associated with some vertex v of G. For each pair of arcs $\alpha \triangleright \beta$ around μ call v the *apex* between α and β, and call $\ell(v)$ the *space* between α and β. If μ is a C-node, annotate each pair of child arcs $\alpha \triangleright \beta$ around μ with a vertex of G or with ς, called the *apex* between α and β, and refer to $\ell(v)$ as the *space* between α and β. Expand the notion of apices and spaces to pairs of leaves $x \triangleright y$ of T. If the root of T is a C-node μ with exactly one child ν and x and y are the rightmost and leftmost leaves of the subtree of T rooted at ν, respectively, define the *apex* (*space*) between x and y as the apex (space) between (μ, ν) and (μ, ν) around μ. Otherwise, the lowest common ancestor μ of $x \neq y$ in T has child nodes $\nu \neq \nu'$ so that x is the rightmost leaf in the subtree of T rooted

at ν, y is the leftmost leaf in the subtree of T rooted at ν' and $(\mu,\nu) \triangleright (\mu,\nu')$ around μ. Define the *apex* (*space*) between x and y as the attachment (space) between (μ,ν) and (μ,ν') around μ.

Reference Network. At this point, the only pairs of arcs $\alpha \triangleright \beta$ around a node μ between which we have not defined an apex and a space yet, are those where μ is a C-node and α or β is its parent edge. For our algorithm it will be important to have access to this information, so it needs to be kept up to date across manipulations of the PC-tree performed by our algorithm. To this end, we introduce references that resolve to the apices we need and maintain them in what we call a *reference network*, which is a rooted forest P_T. Each vertex p of P_T is called a *reference*. It may optionally be annotated with a vertex v of G. Moreover, p may have a parent q in P_T, i.e., P_T contains the directed edge (p,q). Each edge of P_T has a *timestamp* $t(e) \in \{1,2,\dots,k\}$. The reference p can be resolved at time $i \in \{1,2,\dots,k\}$ as follows. First, walk up from p in P_T as far as possible, only using edges whose timestamp is at most i. Let r denote the reference of P_T that is reached in this way. We will guarantee that r is annotated with some vertex u of G. We then say that p *resolves* to u at time i and write $A(P_T,i,p) = u$.

Now let μ be a C-node of T and let $\alpha \triangleright \beta$ be arcs around μ such that α or β is the parent edge of μ. Our construction ensures that there is a reference p of P_T between α and β. Let u denote the vertex of G that p resolves to at time k. Call u the *apex* between α and β, and call $\ell(u)$ the *space* between α and β.

Later, we will augment a multi-source graph to a single-source graph by inserting for each source v except for one an edge from an apex a to v. We use the reference network to determine these apices. To do so efficiently, we use a reference p as a placeholder for a. Each vertex p of the reference network is annotated with a set $Z(p)$ of elements of the form $\langle v,t\rangle$ where v is a vertex of G and t is a level of G, i.e., $1 \le t \le k$. For each $\langle v,t\rangle \in Z(p)$ define the edge

$$e(P_T,\langle v,t\rangle) = \begin{cases} (A(P_T,t,p),v) & \text{if } A(P_T,t,p) \neq \varsigma \\ (\min V(G),v) & \text{if } A(P_T,t,p) = \varsigma \end{cases}$$

Define $E_T = \{e(P_T,\langle v,t\rangle) \mid \langle v,t\rangle \in Z(p) \text{ for all } p \in V(P_T)\}$ and observe $E_T \subset V \times V$. This is the set of edges that we use to augment the multi-source graph to a single-source graph.

Equivalence Transformations. There are two equivalence transformations for PC-trees. Namely, the order of arcs around P-nodes may be arbitrarily permuted and the order of arcs around C-nodes may be reversed. In the former case nothing further needs to be done. In the latter case, we need to update the reference network.

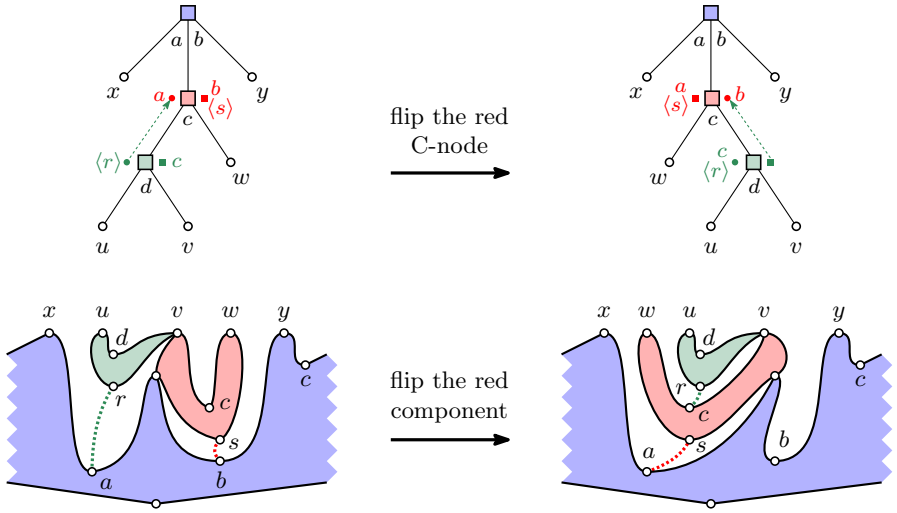


Figure 5.2: Flipping the red C-node in the PC-tree shown on top corresponds to flipping the red component in the graph shown below. This affects which augmentation edges need to be inserted in order to remove the sources s, r of the red and green component, respectively. The reference network, shown as colored annotations in the PC-tree, keeps track of the apices from which these augmentation edges originate.

Let μ be a C-node that has a parent λ, and let p and q denote the reference before and after (λ, μ) around μ. A *reference flip* of μ exchanges the parent edges and annotated vertices (including no parent edge and no annotated vertex) of p and q in P_T. Suppose that T arises from T' by reversing a C-node μ. The reference network P_T is obtained from $P_{T'}$ by performing a reference flip on μ and all of its C-node children.

A PC-tree S obtained from T by a sequence of such equivalence transformations is *equivalent* to T, we write $S \equiv T$. Such a PC-tree S also defines a reference network P_S. See Figure 5.2 for an example of a PC-tree and the reference network.

5.5 Invariant Properties

To prove the correctness of our algorithm, we define two properties and show that if they hold, then they also hold after performing one more basic step of our algorithm. In order to formulate these properties we need some further definitions. Let (μ, ν) be an arc of T where ν is an inner node of T. Define $H_T((\mu, \nu))$ as the spaces incident to (μ, ν) around μ and ν, and define $H_T(\nu)$ as the set of spaces between pairs of

leaves $x \triangleright y$ in the subtree of T rooted at ν. For each $S \equiv T$, define $G_S = G + E_S$.

Property 1. *For each $S \equiv T$ the following statements are true.*

1. *The graph G_S has the single source* $\min V(G)$.
2. *There is at most one pair of child arcs $\alpha \triangleright \beta$ in S so that the apex between α and β is ς.*
3. *For each arc (μ, ν) of S where ν is an inner node, it is* $\max H_S((\mu,\nu)) \leq \min H_S(\nu)$.
4. *For each arc (μ, ν) of S where ν is a C-node let o, r denote the references before and after (μ, ν) around ν, respectively.*
   a) *If μ is a P-node annotated with a vertex v of G, then o and r are annotated with v and have no outgoing edge in P_S.*
   b) *If μ is a C-node let p, q denote the references or apices before and after (μ, ν) around μ, respectively. If p is a reference, then there exists a directed path from r to p in P_S. Otherwise, r is annotated with p and has no outgoing edge in P_S. Symmetrically, if q is a reference, then there exists a directed path from o to q in P_S. Otherwise, o is annotated with q and has no outgoing edge in P_S.*

Let $S \equiv T$. Obtain the graph $\bar{G}_S$ from G_S as follows. If ς occurs as an apex in S, add the vertex ς and the edge $(\varsigma, \min V(G))$ to $\bar{G}_S$. Consider each pair of leaves $x \triangleright y$ in S. Let $x = \mu_1, \mu_2, \ldots, \mu_n = y$ denote the nodes on the unique path from x to y in S. For $1 < i < n$ it is $\{\mu_{i-1}, \mu_i\} \triangleright \{\mu_i, \mu_{i+1}\}$ around μ_i. Let a_i denote the apex between these two arcs around μ_i. Define $\mathcal{A}_S(x, y)$ as the sequence obtained from $(a_2, a_3, \ldots, a_{n-1})$ by removing consecutive duplicates of the same apex. For each a in $\mathcal{A}_S(x, y)$ create a level-k vertex $\bar{a}$ and add it to $\bar{G}_S$ together with the edge $(a, \bar{a})$. Subdivide $(a, \bar{a})$ so that $\bar{G}_S$ becomes proper, creating an *attachment path* originating from a and ending in $\bar{a}$. Observe that if G_S is a single-source graph (cf. Property 1 Statement 1), then so is $\bar{G}_S$.

Let G be a connected k-level graph and let T be a PC-tree so that the level-k vertices of G are the leaves of T. Let $S \equiv T$, and let $\mathcal{E}$ be a level-planar embedding of G. We say that S *matches* $\mathcal{E}$ and vice versa if the frontier of S equals the frontier of $\mathcal{E}$. Let U be some subset of the level-k vertices of G, that is, a subset of the leaves of S.

Property 2. *For each $S \equiv T$ there exists a level-planar embedding $\bar{\mathcal{G}}_S$ of $\bar{G}_S$ so that the following statements hold true, where $\mathcal{G}_S$ denotes the restriction of $\bar{\mathcal{G}}_S$ to G_S.*

1. *The level-planar embedding $\mathcal{G}_S$ matches S.*
2. *The vertices in U are consecutive in $\mathcal{G}_S$.*

3. *For leaves $x \triangleright y$ in S the endpoints of the attachment paths originating from the apices in $\mathcal{A}_S(x, y)$ appear between x and y in $\bar{\mathcal{G}}_S$ in this order.*

4. *For vertices $x \triangleright y$ in $\mathcal{G}_S$ the graph $I(j, k)$ can be merged into $\mathcal{G}_S$ between x and y if and only if the space h between x and y in S satisfies $h < j$.*

5. *For every level-planar embedding $\mathcal{E}$ of G in which the vertices in U are consecutive there exists an $R \equiv T$ such that R matches $\mathcal{E}$ and for vertices $x \triangleright y$ in $\mathcal{E}$ if $I(j, k)$ can be merged into $\mathcal{E}$ between x and y, then the space h between x and y in R satisfies $h < j$.*

We say that T *U-represents* G if Properties 1 and 2 hold true for it. Note that if T U-represents G, then each $S \equiv T$ U-represents G. We say that T *represents* G if T $\emptyset$-represents G. Our approach is to show that if G is level planar, then we our algorithm computes a non-null tree T that represents G. To this end we consider each basic operation one by one, showing Properties 1 and 2 by induction. Therefore, our algorithm tests for level planarity. Moreover, using our final PC-tree T we find that augmenting G by the edge set E_T gives a single-source level-planar graph G_T. We can then either use the algorithm due to Di Battista and Nardelli to compute a level-planar embedding of G_T [], or vertically flip G_T and run our algorithm a second-time, producing a single-source single-sink level-planar graph. For these graphs, level planarity and planarity coincide, so we can use any planar embedding algorithm to compute a level planar embedding. In either case, omitting the augmentation edges gives a level-planar embedding of G. The latter idea was proposed by Jünger and Leipert, although they eliminate sinks rather than sources [].

We start with the introduce operation, which serves as the base case for our inductive correctness proof. In this case G consists of a single vertex v. Define T as the PC-tree that consists of a root C-node μ and a single leaf v connected by the arc (μ, v). Define the apex between (μ, v) and (μ, v) around μ as ς. The reference network P_T consists of the vertex ς with $Z(\varsigma) = \emptyset$ (we do not need any augmentation edges yet). The following serves as the base case for our inductive correctness proof.

Corollary 4. *Let G be a k-level graph that consists of a single level-k vertex, and let T be defined as above. Then T represents G.*

5.6 Grow

Let G be a unary k-level graph that arises from $G' = G[k-1]$ by growing u with $v_1, v_2, \ldots, v_n$. Let T' be a PC-tree that represents G'. Obtain T from T' as follows. First, replace all non-pertinent leaves with their unique child in G. Replace u with a newly created P-node μ whose children are the leaves $v_1, v_2, \ldots, v_n$. Annotate μ

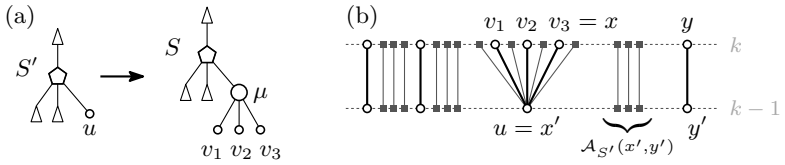


Figure 5.3: Growing a PC-tree (a) and the corresponding embedding (b). Edges of the graph are black and thick, whereas gray thin edges belong only to $\bar{\mathcal{G}}_S$.

with the vertex u. Refer to the resulting tree as T and say that T arises from T' by *growing* u with $v_1, \ldots, v_n$.

Lemma 22. *Let G be a unary k-level graph that arises from $G' = G[k-1]$ by growing a vertex u with $v_1, v_2, \ldots, v_n$. Let T' be a PC-tree that represents G' and let T be the PC-tree obtained from T' by growing u with $v_1, \ldots, v_n$. Then T represents G.*

Proof. Let $S \equiv T$. There exists a unique $S' \equiv T'$ so that S arises from S' without reordering.

We first show that Property 1 holds. For Statement 1, let $S \equiv T$. Observe that there exists a unique $S' \equiv T'$ so that S arises from S' without reordering. Then $E_S = E_{S'}$. By assumption for S', the graph $G'_{S'}$ has the single source $\min V(G)$. Every level-k vertex of G_S has a parent in $G_S[k-1]$ and it is $G_S[k-1] = G'_{S'}$. So G_S is a level graph with the single source $\min V(G)$, i.e., Statement 1 holds. For Statement 2, observe that the only apex in S that is not an apex in S' is $u \neq \varsigma$, so Statement 2 follows by assumption for S'. For Statement 3, consider an arc (ι, κ) of S such that κ is an inner node. If $(\iota, \kappa) \neq (\lambda, \mu)$, then $H_S((\iota, \kappa)) = H_{S'}((\iota, \kappa))$ and $H_S(\kappa) \subset (H_{S'}(\kappa) \cup \{\ell(u) = k-1\})$. All spaces in $H_{S'}(\kappa)$ correspond to vertices of $G[k-2]$. In particular, the only apex that does not correspond to a vertex of $G[k-2]$ is ς, which cannot occur in the subtree of S' rooted at κ by assumption. Therefore $\min H_S(\kappa) = \min H_{S'}(\kappa)$. It is further $\max H_{S'}((\iota, \kappa)) \leq \min H_{S'}(\kappa)$ by assumption for S', and then $\max H_S((\iota, \kappa)) \leq \min H_S(\kappa)$ follows. Otherwise $(\iota, \kappa) = (\lambda, \mu)$. It is $\max H_S((\lambda, \mu)) \leq k-2$ and $H_S(\mu) = \{\ell(u) = k-1\}$. This shows Statement 3. Statement 4 holds because each arc (μ, ν) of S where ν is a C-node is also an arc of S', where the statement holds by assumption.

We now turn to Property 2. Let π be a permutation so that $v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)}$ is the linearized order of children of μ in S. Obtain $\bar{\mathcal{G}}_S$ from $\bar{\mathcal{G}}'_{S'}$ by extending an edge from each non-pertinent vertex and from the endpoint of each attachment path, and by adding $\bar{u}_0, v_{\pi(1)}, \bar{u}_1, v_{\pi(2)}, \ldots, v_{\pi(n)}, \bar{u}_n$ as children of u so that they appear in that linearized order around u in $\bar{\mathcal{G}}_S$. See Figure 5.3.

Statement 1 holds by construction of $\bar{\mathcal{G}}_S$. Statement 2 holds vacuously for $U = \emptyset$.

For Statement 3, consider leaves $x \triangleright y$ around S. If $x = v_{\pi(i)}$ and $y = v_{\pi(i+1)}$, then $\bar{u}_i$ appears between $v_{\pi(i)}$ and $v_{\pi(i+1)}$ in $\bar{\mathcal{G}}_S$ by construction. Otherwise, let x' and y' denote the unique parents of x and y in G_S. Then $\mathcal{A}_S(x, y)$ arises from $\mathcal{A}_{S'}(x', y')$ by renaming non-pertinent vertices, plus, if $x' = u$, prepending u and, if $y' = u$, appending u. Then Statement 3 holds by construction of $\bar{\mathcal{G}}_S$.

For Statement 4, consider leaves $x \triangleright y$ around S. If $x = v_{\pi(i)}$ and $y = v_{\pi(i+1)}$, then the apex between x and y in S is u and the space between x and y in S is $\ell(u) = k-1$. Then for $j \leq k-1$ the graph $I(j,k)$ cannot be merged into $\bar{\mathcal{G}}_S$ between x and y, whereas the graph $I(k,k)$ can be trivially merged into $\bar{\mathcal{G}}_S$ between x and y by interspersing a vertex after x. Otherwise, let x' and y' denote the unique parents of x and y in G_S. The graph $I(k,k)$ can be merged into $\bar{\mathcal{G}}_S$ by interspersing a vertex after x. For $j < k$ the graph $I(j,k)$ can be merged into $\bar{\mathcal{G}}_S$ between x and y if and only if $I(j,k-1)$ can be merged into $\bar{\mathcal{G}}'_{S'}$ between x' and y'. Because S arises from S' without reordering, the space between x and y in S is the space between x' and y' in S'. Then Statement 4 holds for S because it holds for S'.

For Statement 5 consider a level-planar embedding $\mathcal{E}$ of G. Let ρ be the permutation so that $v_{\rho(1)}, v_{\rho(2)}, \ldots, v_{\rho(n)}$ is the linearized order of children of u in $\mathcal{E}$. Let $\mathcal{E}'$ denote the restriction of $\mathcal{E}$ to G'. By assumption there exists an $R' \equiv T'$ such that R' matches $\mathcal{E}'$ and for each pair x', y' of consecutive vertices in $\mathcal{E}'$ if $I(j,k-1)$ can be merged into $\mathcal{E}'$ between x' and y', then the space h' between x' and y' in R' satisfies $h' < j$. Obtain R from R' by growing u with $v_1, \ldots, v_n$ and then permuting the children of the newly-created P-node μ so that their linearized order is $v_{\rho(1)}, \ldots, v_{\rho(n)}$. Consider two vertices $x \triangleright y$ in $\mathcal{E}$. If $x = v_{\rho(i)}$ and $y = v_{\rho(i+1)}$ use the same argument as in the previous paragraph to argue that if $I(j,k)$ can be merged into $\mathcal{E}$ between x and y, then $j = k$ and note that the space between x and y in R is $\ell(u) = k-1$ by construction of R. Otherwise, if $I(j,k)$ can be merged into $\mathcal{E}$ between x and y, then $I(j,k-1)$ can be merged into $\mathcal{E}'$ between x' and y', i.e., the space h' between x' and y' in R' satisfies $h' < j$. By construction of R, the space h between x and y in R equals the space h' between x' and y' in R'. So $h = h' < j$ and Statement 5 holds. □

5.7 Tree Operation Contract

Inner nodes with exactly one child do not offer any additional freedom in choosing the frontier of a PC-tree, so they are candidates for contracting. However, our definition of PC-trees requires the root to be a C-node. Therefore, we disallow contracting a C-node when it is the root and its unique child is a P-node. Hence, a node μ of T is *contractible* if it has exactly one child ν and

(A) it is the root node and ν is a C-node, or

(B) it has a parent node.

In case (A) let a denote the apex between (μ,ν) and (μ,ν) around μ. Let α and β denote the arcs that precede and succeed (μ,ν) around ν, respectively. Let p denote the reference between α and (μ,ν) around ν, and let q denote the reference between (μ,ν) and β around ν. Delete the node μ and the arc (μ,ν). Define the apex between α and β around ν as a. If the endpoint of α is a C-node, change the reference before α to p. Symmetrically, if the endpoint of β is a C-node, change the reference after β to q. In case (B) let λ denote the parent node of μ. There are four (sub)cases; see Figure 5.4.

1. ν is a P-node. Replace the arc (λ,μ) with the arc (λ,ν) around λ, and replace the arc (μ,ν) with the arc (λ,ν) around ν. Finally, delete μ. The reference network remains unchanged.

2. ν is a C-node.
   a) μ is a C-node. Let p denote the reference between (μ,ν) and (λ,μ), and let q denote the reference between (λ,μ) and (μ,ν). Contract μ as in the previous case. Then, replace the references before and after (λ,ν) around ν with p and q, respectively. The reference network remains unchanged.
   b) μ is a P-node.
      i. λ is a C-node. Contract μ as in the first case. Insert an edge with timestamp k from the reference before (λ,ν) around ν to the reference after (λ,ν) around λ. Symmetrically, insert an edge with timestamp k from the reference after (λ,ν) around ν to the reference before (λ,ν) around λ.
      ii. λ is a P-node. Let p,q denote the references before and after (μ,ν) around ν. Contract μ as in the first case. Create new references p',q' and annotate them with the vertex with which λ is annotated. Add edges (p,p'), (q,q') with timestamp k to the reference network. Finally, set the references before and after (λ,ν) around ν as p' and q', respectively.

Lemma 23. *Let G be a level graph and let T' be a PC-tree that U-represents G. If T arises from T' by contracting a node, then T U-represents G.*

Proof. Let $S \equiv T$. There exists an $S' \equiv T'$ so that S arises from S' by contracting μ such that $E_S = E_{S'}$. This immediately follows in the cases where the reference network remains unchanged. In the remaining cases, i.e., when ν is a C-node and μ is a P-node,

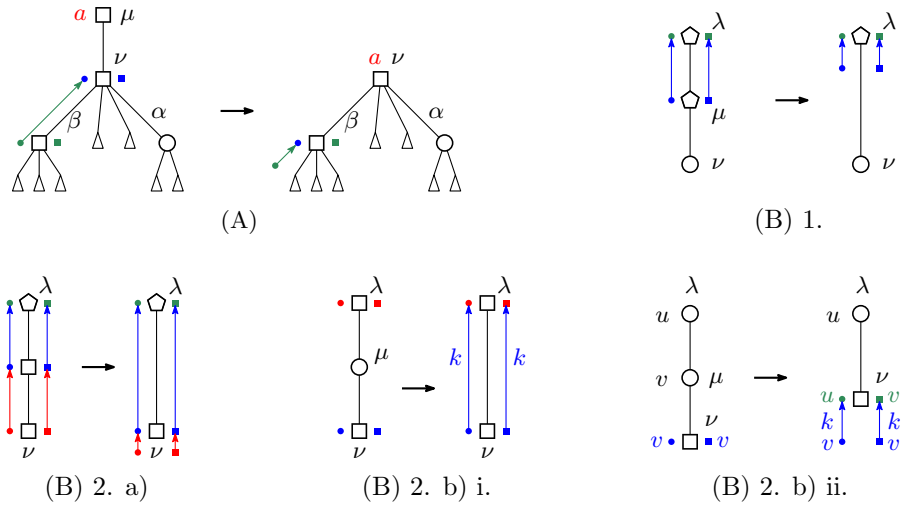


Figure 5.4: The different cases of the contract operation.

the only changes are that we add edges with timestamp k to the reference network, so all previous references are resolved to the same vertices of G.

We first show that Property 1 holds. For Statement 1, let $S \equiv T$. There exists an $S' \equiv T'$ so that S arises from S' by contracting μ and $E_S = E_{S'}$. Then $G_S = G_{S'}$ has the single source $\min V(G)$ by assumption for S'. For Statement 2, observe that the apices in S are a sub(multi)set of the apices in S'. For Statement 3, consider an arc (ι, κ) of S. If (ι, κ) also exists in S', then $\max H_S((\iota,\kappa)) \leq \max H_{S'}((\iota,\kappa))$. and $H_S(\kappa) = H_{S'}(\kappa)$ by construction of T. It is $\max H_{S'}((\iota,\kappa)) \leq \min H_{S'}(\kappa)$ by assumption for S', so it follows that $\max H_S((\iota,\kappa)) \leq \min H_S(\kappa)$. Otherwise, if (ι,κ) does not exist in S', it is $(\iota,\kappa) = (\lambda,\nu)$. Then $\max H_S((\lambda,\nu)) = \max H_{S'}((\mu,\nu))$ and $\min H_S(\nu) = \min H_{S'}(\nu)$ by construction of T. By assumption for S', it holds that $\max H_{S'}((\mu,\nu)) \leq \min H_{S'}(\nu)$. So, $\max H_S((\lambda,\nu)) \leq \min H_S(\nu)$. This shows Statement 3. The contract operation is designed so that Statement 4 holds for (λ,ν). All remaining arcs of S are also arcs in S', where the statement holds by assumption.

We now turn to Property 2. If μ is a C-node, define $\bar{\mathcal{G}}_S = \bar{\mathcal{G}}_{S'}$. Otherwise, if μ is a P-node, let v denote the vertex associated with μ. Obtain $\bar{\mathcal{G}}_S$ from $\bar{\mathcal{G}}_{S'}$ by removing the two attachment paths starting at v that originate from μ. Statement 1, Statement 2 and Statement 3 then immediately follow. For Statement 4 and Statement 5 observe that for any leaves $x \triangleright y$ of S the space between x and y in S equals the space between x and y in S'. □

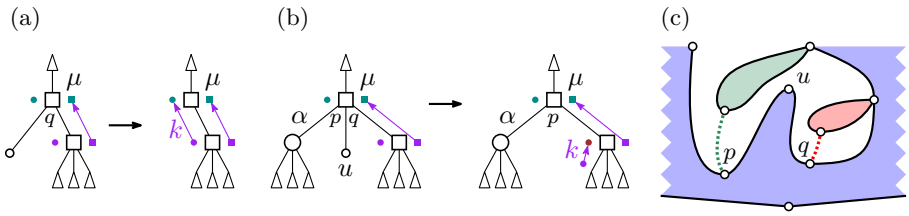


Figure 5.5: Two cases of pruning a vertex u (a,b). The reference network keeps track of the correct apices over time. Case (b) roughly corresponds to the situation shown in (c). The information that the source of the red component is removed by augmenting with an edge originating from q must not be overwritten by the information that the source of the green component is removed by an edge starting in p. Timestamped edges prevent such overwrites.

5.8 Prune

Let G be a unary k-level graph that arises from $G' = G[k-1]$ by pruning u. Let T' be a PC-tree that represents G'. Note that T' has at least two leaves. Let μ denote the parent of u in T'. Lemma 23 lets us assume without loss of generality that T' does not contain any contractible nodes, i.e., μ has at least two children.

See Figure 5.5. If μ is a P-node, delete u and its parent arc. If μ is a C-node, let α and β denote the arcs before and after (μ, u) around μ. Let p denote the apex or reference between α and (μ, u) and let q denote the apex or reference between (μ, u) and β. Delete u and its parent arc. Consider the case that α is the parent arc; see Figure 5.5 (a). Then p is a reference and we define the reference between α and β as p. Moreover, if the endpoint of $\beta = (\mu, \nu)$ is a C-node, let r denote the reference after (μ, ν) around ν. Then insert the edge (r, q) with timestamp k into the reference network. If β is the parent arc, then q is a reference and we define the reference between α and β as q. Moreover, if the endpoint of $\alpha = (\mu, \nu)$ is a C-node, let r denote the reference before (μ, ν) around ν. Then insert the edge (r, p) with timestamp k into the reference network. Finally, consider the remaining case, i.e., α and β are child arcs; see Figure 5.5 (b). Then p and q are apices. Then define the apex between α and β as the vertex $a \in \{p, q\}$ with $\ell(a) = \min\{\ell(p), \ell(q)\}$. Assume $a = p$, the case $a = q$ is symmetric. If the endpoint of $\beta = (\mu, \nu)$ is a C-node, let r denote the reference after (μ, ν) around ν. Replace r with a new reference r' annotated with p in the PC-tree, and add the edge (r, r') with timestamp k into the reference network. Denote the result by T.

Lemma 24. *Let G be k-level graph that arises from $G' = G[k-1]$ by pruning u. Let T'*

be an incontractible PC-tree that represents G' and let T be the PC-tree obtained from T' by pruning u. Then T represents G.

Proof. Let $S \equiv T$. Then there exists an $S' \equiv T'$ so that S arises from S' by pruning u without reordering. Note that $P_S \neq P_{S'}$, but because all edges present in P_S but not present in $P_{S'}$ have timestamp k and hence it is $E_S = E_{S'}$.

We first show that Property 1 holds. For Statement 1 note that because $E_S = E_{S'}$ it follows that $G_S[k-1] = G'_{S'}$. Further, every level-k vertex of G_S has a parent in $G_S[k-1] = G'_{S'}$, so G_S is a level graph with the single source $\min V(G)$. For Statement 2 observe that the apices in S are a sub(multi)set of the apices in S'. For Statement 3 consider an arc (ι, κ) of S. Then (ι, κ) is an arc in S' as well. By assumption it is $\max H_{S'}((\iota, \kappa)) \leq \min H_{S'}(\kappa)$. If (ι, κ) is not adjacent to (μ, u) in S', then $H_S((\iota, \kappa)) = H_{S'}((\iota, \kappa))$. Otherwise it is $\max H_S((\iota, \kappa)) \leq \max H_{S'}((\iota, \kappa))$ by construction of T. In particular, if μ has a parent λ and (μ, u) is either the leftmost or rightmost child of μ in S, then the inequality holds by assumption for (λ, μ). Otherwise, the inequality holds explicitely by choice of r. Moreover, it is $H_S(\kappa) \subseteq H_{S'}(\kappa)$, so $\min H_S(\kappa) \geq H_{S'}(\kappa)$. Together, this gives $\max H_S((\iota, \kappa)) \leq \min H_S(\kappa)$. This shows Statement 3. The prune operation is specifically designed so that Statement 4 holds for the arcs α and β that were incident to (μ, u) in S'. For all remaining arcs the statement holds by assumption for S'.

We now turn to Property 2. Observe that at least one of p, q is an apex and exactly one apex a is deleted from the PC-tree during the construction of T. Obtain $\bar{\mathcal{G}}_S$ from $\bar{\mathcal{G}}'_{S'}$ by extending an edge from each non-pertinent vertex, deleting the attachment path from a to $\bar{a}$, and extending an edge from each remaining attachment vertex. Statement 1 and Statement 2 hold trivially. Statement 3 holds by construction of T.

For Statement 4 consider two vertices $x \triangleright y$ in $\mathcal{G}_S$ and let x', y' denote their unique parents in G. If $x' \triangleright y'$ in $\mathcal{G}'_{S'}$, then $I(j,k)$ can be merged into $\mathcal{G}_S$ between x and y if and only if $I(j,k-1)$ can be merged into $\mathcal{G}'_{S'}$ between x' and y', i.e., if and only the space h' between x' and y' in S' satisfies $h' < j$. By construction of S the space between x and y in S is h', so Statement 4 holds. Otherwise x' and y' precede and succeed u in $\mathcal{G}'_{S'}$, respectively. Then $I(j,k)$ can be merged into $\mathcal{G}_S$ between x and y if and only if $I(j,k-1)$ can be merged into $\mathcal{G}'_{S'}$ between x' and u, or between u and y'. By assumption, this is possible if and only if $\min(h'_x, h'_y) < j$, where h'_x is the space between x' and u in S' and h'_y is the space between u and y' in S'. By construction of S, and because of Property 1 Statement 3, the space h between x and y in S is $\min(h'_x, h'_y)$. So, Statement 4 holds.

For Statement 5 consider a level-planar embedding $\mathcal{E}$ of G. Let $\mathcal{E}'$ denote the restriction of $\mathcal{E}$ to G'. By assumption there exists a PC-tree $R' \equiv T'$ such that R' matches $\mathcal{E}'$ and for each pair of vertices $x' \triangleright y'$ in the frontier of $\mathcal{E}'$, if $I(j,k-1)$ can be merged into $\mathcal{E}'$ between x' and y', then the space h' between x' and y' in R'

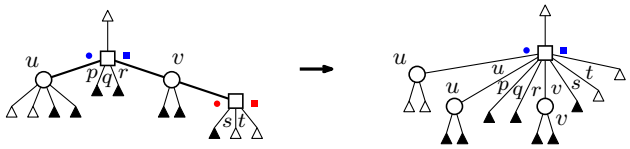


Figure 5.6: Handling of apices and references during an update.

satisfies $h' < j$. Obtain R from R' by pruning u. Clearly R matches $\mathcal{E}$. Consider a pair of vertices $x \triangleright y$ in $\mathcal{E}$ and let x', y' denote their unique parents in G. First, assume $x' \triangleright y'$ in $\mathcal{E}'$. If $I(j,k)$ can be merged into $\mathcal{E}$, then $I(j,k-1)$ can also be merged into $\mathcal{E}'$ between x' and y'. By assumption, the space h' between x' and y' in R' satisfies $h' < j$. By construction, the space between x and y in R is h'. Otherwise, $x' \triangleright u \triangleright y'$ around $\mathcal{E}'$. If $I(j,k)$ can be merged into $\mathcal{E}$ between x and y, then $I(j,k-1)$ can be merged into $\mathcal{E}'$ between x' and u, or between u and y'. By assumption, then $\min(h'_x, h'_y) < j$, where h'_x is the space between x' and u in R' and h'_y is the space between u and y' in R'. By construction of R, the space h between x and y in R is $\min(h'_x, h'_y)$, so Statement 5 holds. □

5.9 Tree Operation Update

Let T' be a PC-tree and let T be the PC-tree obtained from T' by updating with a subset $U = \{u_1, u_2, \ldots, u_n\}$ of its leaves. Recall that the update operation for PC-trees creates a new C-node χ. Annotate χ with apices as follows; see Figure 5.6.

Let $\alpha \triangleright \beta$ denote child arcs around χ. Further, let u (let v) denote the rightmost leaf (the leftmost leaf) in the subtree of T' rooted at the endpoint of α (at the endpoint of β). Define the apex between α and β in T as the apex between u and v in T'. As a side note, observe that this apex is incident to the terminal path, so we will be able to find these apices in running time proportional to the length of the terminal path. If the root of T' does not lie on the terminal path, let (μ,ν) denote the unique arc of T' whose endpoint lies on the terminal path. If ν is a C-node, move the references before and after (μ,ν) around ν in T' before and after (μ,χ) around χ in T. If ν is a P-node, create new references before and after (μ,χ) around χ in T and annotate them with the vertex with which ν is annotated. Now consider a child edge (χ,ν) of χ in T such that ν is a C-node. Note that ν is a child of a C-node λ on the terminal path in T'. Any apex incident to (λ,ν) in T' is also incident to (χ,ν) in T by construction. Now suppose that there is a reference r before (after) (λ,ν) around λ in T', and there is an apex a before (after) (χ,ν) around χ in T. Let $\hat{r}$ denote the root of the tree of $P_{T'}$ to which r belongs. Note that $\hat{r}$ is annotated with a. In T, set the reference after

(before) (χ, ν) around ν to $\hat{r}$. Finally, if the highest node on the terminal path in T' is a P-node, we may need to add incoming edges to the newly created references around the parent arc of χ in T. In particular, suppose that the leftmost child δ of χ in T is a C-node. Let r denote the reference after (χ, δ) around δ. Add an edge with timestamp k to the reference network. Proceed symmetrically for the rightmost child of χ.

Lemma 25. *Let G be a level graph and let T' be a PC-tree that represents G. If T arises from T' by U-updating, then T U-represents G.*

Proof. We first show that Property 1 holds. Let $S \equiv T$. There exists a unique $S' \equiv T'$ so that S arises from S' by U-updating without reordering. In particular, S' and S have the same frontier. Further, if the highest node on the terminal path is a C-node, then $P_S = P_{S'}$ and hence $E_S = E_{S'}$. Otherwise, if the highest node on the terminal path is a P-node, then P_S arises from $P_{S'}$ by introducing two new references p, q with $Z(p) = Z(q) = \emptyset$ and all new edges have timestamp k. Then it is $E_S = E_{S'}$.

Statement 1 holds because $E_S = E_{S'}$. For Statement 2 observe that each P-node of S is annotated with a vertex v of G, i.e., $v \neq \varsigma$. Each C-node $\mu \neq \chi$ of S is a C-node in S' that has the exact same apices and does not lie on the terminal path. Each apex around χ in S is a distinct apex around a node on the terminal path in S'. This shows Statement 2.

For Statement 3, first consider the parent arc (λ, χ) of χ. Let (μ, ν) denote the unique arc of S' so that μ does not lie on the terminal path and ν does lie on the terminal path. It is $H_S((\lambda, \chi)) = H_{S'}((\mu, \nu))$ and $H_S(\chi) \subseteq H_{S'}(\nu)$ by construction of T. Then $\max H_S((\lambda, \chi)) \leq \min H_S(\chi)$ follows from $\max H_{S'}((\mu, \nu)) \leq \min H_{S'}(\nu)$, which holds by assumption for S'. Now consider a child arc (χ, ν) of χ. If ν is a split P-node, let ν' denote the original P-node in S', otherwise let $\nu' = \nu$. Let μ' denote the parent node of ν' in S'. Note that μ' lies on the terminal path. By definition of the apices around χ and using Statement 3 on S' we obtain that it is $\max H_S((\chi, \nu)) \leq \max H_{S'}((\mu', \nu'))$ and $H_S(\nu) \subseteq H_{S'}(\nu')$. Then Statement 3 holds for (χ, ν). Finally, consider an arc (μ, ν) that is not incident to χ. If μ is a split P-node, let μ' denote the original P-node in S', otherwise let $\mu' = \mu$. Likewise, if ν is a split P-node, let ν' denote the original P-node in S', otherwise let $\nu' = \nu$. Then (μ', ν') is an arc in S' and it is $H_S((\mu, \nu)) = H_{S'}((\mu', \nu'))$ and $H_S(\nu) \subseteq H_{S'}(\nu')$. This shows Statement 3. Our handling of the references in the construction of T is explicitly designed to ensure that Statement 4 holds for all arcs incident to χ. For an arc (μ, ν) where μ is a split P-node and ν is a C-node the statement holds because μ is annotated with the same vertex as its original P-node in T. All remaining arcs in S exist unchanged in S', where the statement holds by assumption.

We now turn to Property 2. So, define $\bar{\mathcal{G}}_S$ as $\bar{\mathcal{G}}_{S'}$. Let $x \triangleright y$ be two leaves of S and S'. Then $A_S(x, y) = A_{S'}(x, y)$. Together with $E_S = E_{S'}$ it is $G_S = G_{S'}$, so Statement 1

and Statement 2 hold trivially. Statements 3 and 4 hold by construction of T. For Statement 5 consider a level-planar embedding $\mathcal{E}$ of G in which the vertices in U are consecutive. There exists an $R' \equiv T'$ such that R' that matches $\mathcal{E}$ and for each pair of vertices $x \triangleright y$ in $\mathcal{E}$ if $I(j,k)$ can be merged into $\mathcal{E}$ between x and y, then the space h' between x and y in R' satisfies $h' < j$. Obtain R from R' by U-updating (without reordering, because the vertices in U are consecutive in $\mathcal{E}$ and R'). As with the previous statement, for each $x \triangleright y$ around R the space h between x and y in R equals the space h' between x and y in R'. This shows Statement 5. □

5.10 Unary Bundle

Let G be a k-level graph that arises from $G' = G[k-1]$, which is a connected graph, by bundling $U = \{u_1, u_2, \ldots, u_n\}$ into v. Let T' be a PC-tree that U-represents G'. We obtain T from T' by *replacing* $u_1, \ldots, u_n$ with v as follows.

There are three cases. First, consider the case that $u_1, \ldots, u_n$ comprise all the leaves of T'. We seek to preserve an apex on the smallest level. Let P denote the set of apices stored in T' and let $p = \min P$. Define T as the PC-tree that consists of the root C-node μ and a single leaf v connected by an arc (μ, v). Define the apex between (μ, v) and (μ, v) as p. Define $P_T = P_{T'}$.

Lemma 26. *Let G be a k-level graph that arises from $G' = G[k-1]$ by unary bundling $U = \{u_1, u_2, \ldots, u_n\}$ into v. Let T' be a PC-tree that U-represents G' such that U denotes the set of leaves of T', and let T be the PC-tree obtained from T' by replacing $u_1, \ldots, u_n$ with v. Then T represents G.*

Proof. We first show that Property 1 holds. Let $S \equiv T$, i.e., $S = T$. Because $P_T = P_{T'}$ it is $E_S = E_{T'}$. For Statement 1, observe that $G'_{T'}$ has a single source $\min V(G')$ and v has parents $u_1, \ldots, u_n$ in G'. Therefore, G_S has the single source $\min V(G') = \min V(G)$. Because S contains a single arc Statements 2, 3 and 4 trivially hold true.

We now turn to Property 2. Let $S \equiv T$, i.e., $S = T$. Let π be a permutation so that $u_{\pi(1)}, u_{\pi(2)}, \ldots, u_{\pi(n)}$ is the frontier around T' and the apex p between $u_{\pi(n)}$ and $u_{\pi(1)}$ in T' equals the apex between v and v in S. Obtain $\bar{\mathcal{G}}_S$ from $\bar{\mathcal{G}}'_{T'}$ as follows. Remove all attachment paths except for the one ending at $\bar{p}$. Extend an edge $e_{\pi(1)}$ from $u_{\pi(1)}$ to v. Then, for each i with $1 < i \leq n$ intersperse the edge $e_{\pi(i)} = (u_{\pi(i)}, v)$ after $e_{\pi(i-1)}$. The linearized order of parents of v in $\bar{\mathcal{G}}_S$ then is $u_{\pi(1)}, u_{\pi(2)}, \ldots, u_{\pi(n)}$. Finally, extend an edge from the attachment vertex $\bar{p}$. Because G has a single vertex on level k, namely v, Statement 1 and Statement 2 trivially hold true. Statement 3 holds by construction of $\bar{\mathcal{G}}_S$. For Statement 4, observe that $I(j,k)$ can be merged into $\mathcal{G}_S$ between v and v if and only if $I(j,k-1)$ can be merged into $\mathcal{G}'_{T'}$ between $u_{\pi(n)}$ and $u_{\pi(1)}$. Using Statement 5 on T' this is the case when $\ell(p) < j$. By construction, $\ell(p)$ is

also the space between v and v in S. So, Statement 4 holds. For Statement 5, consider a level-planar embedding $\mathcal{E}$ of G. Because G has a single vertex on level k, namely v, T matches $\mathcal{E}$. Let ρ be a permutation such that the linearized order of parent vertices of v in $\mathcal{E}$ is $u_{\rho(1)}, u_{\rho(2)}, \ldots, u_{\rho(n)}$. If $I(j,k)$ can be merged between v and v in $\mathcal{E}$, then $I(j,k-1)$ can be merged between $u_{\rho(n)}$ and $u_{\rho(1)}$ in the restriction $\mathcal{E}'$ of $\mathcal{E}$ to G'. Let $R' \equiv T'$ denote a PC-tree as per Statement 5. This gives $h < j$ for the space h between $u_{\rho(n)}$ and $u_{\rho(1)}$ in R'. Our choice of p in the construction of S guarantees $\ell(p) \leq h$, so $\ell(p) < j$. This shows Statement 5. □

In the remaining two cases, there exists a white leaf in T'.

Because T' U-represents G' it was obtained by an update operation. Let χ denote the C-node that is newly created during this update. Assume without loss of generality that no contractible node except for, possibly, χ, applying Lemma 23 as needed. Then the counter-clockwise order of arcs around χ is $\beta, \ldots, \gamma, \delta, \ldots, \alpha$, where the endpoints of $\beta, \ldots, \gamma$ are black and the endpoints of $\delta, \ldots, \alpha$ are white. There are two cases, namely that

(a) χ is the root or the parent of χ is white, or

(b) the parent of χ is black.

See Figure 5.7 (a) and (b) for case (a) and (b), respectively. These two cases differ significantly. Indeed, the treatment of the second case requires quite a bit of extra work. This is the case that we believe was not handled by Bachmaier et al. To understand its importance, see Figure 5.8. In (a), we replace the left-maximal tree, which corresponds to the embedding shown below. In particular, any component H with $\ell(a) < \ell(H)$ can then be merged between $\tilde{v}$ and $\tilde{u}$. If we simply replaced in an arbitrary tree such as the one shown in (b), we would assume that only components H with $\ell(b) < \ell(H)$ can be merged between $\tilde{v}$ and $\tilde{u}$, which is incorrect because $\ell(a) < \ell(b)$.

Consider case (a). Then $\beta, \ldots, \gamma$ are child arcs of χ in T'. Let p denote the reference or apex between α and β, and let q denote the reference or apex between γ and δ. Obtain T from T' as follows. Delete all arcs $\beta, \ldots, \gamma$ and the subtrees rooted at their endpoints. Create a new leaf v and insert the arc (χ, v) between α and δ around χ. Define the reference or apex between α and (χ, v) as p, and define the reference or apex between (χ, v) and β as q. Define $P_T = P_{T'}$.

Lemma 27. *Let G be a k-level graph that arises from $G' = G[k-1]$ by unary bundling $U = \{u_1, u_2, \ldots, u_n\}$ into v. Let T' be a PC-tree that U-represents G' such that χ is the root of T' or the parent of χ is white. Let T be the PC-tree obtained from T' by replacing $u_1, \ldots, u_n$ with v. Then T represents G.*

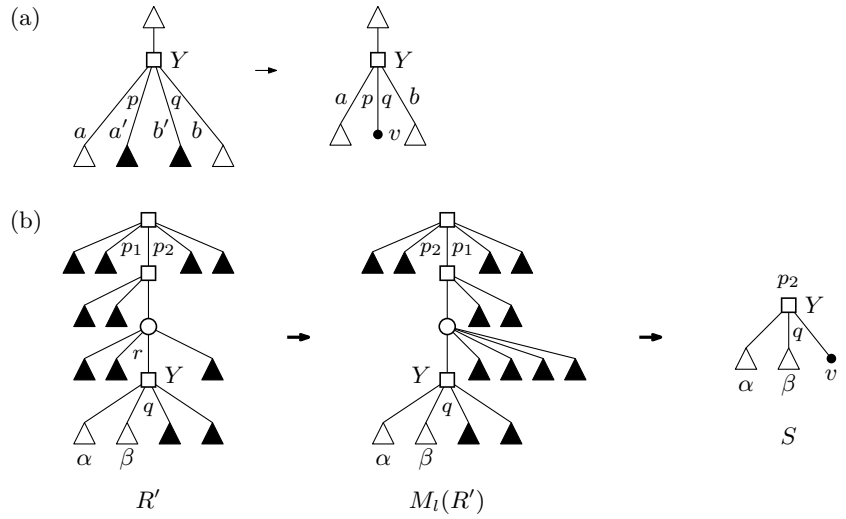


Figure 5.7: Two cases of the replace operation.

Proof. Let $S \equiv T$. Let $S' \equiv T'$ be a PC-tree such that S is obtained from S' by replacing $u_1, \ldots, u_n$ with v. In general this choice of S' is not unique. To handle these degrees of freedom we define a canonical tree $M(S')$ as follows. Let τ be the permutation so that the linearized order of black leaves around T' is $u_{\tau(1)}, u_{\tau(2)}, \ldots, u_{\tau(n)}$ (this linearized order exists because T' has at least one white leaf). If the orientation of χ is the same in S' and T', obtain $M(S')$ from S' by performing equivalence transformations only on black nodes so that the linearized order of black leaves around $M(S')$ is $u_{\tau(1)}, u_{\tau(2)}, \ldots, u_{\tau(n)}$. Otherwise, if the orientation of χ is opposite in S' and T', obtain $M(S')$ from S' by performing equivalence transformations only on black nodes so that the linearized order of black leaves around $M(S')$ is $u_{\tau(n)}, u_{\tau(n-1)}, \ldots, u_{\tau(1)}$. Observe that $M(S') \equiv T'$ and that S is obtained from $M(S')$ by replacing $u_1, \ldots, u_n$ with v. From now assume $S' = M(S')$ without loss of generality. Then $E_S = E_{S'}$.

We first show that Property 1 holds. For Statement 1 observe that $G'_{S'}$ has the single source $\min V(G')$ and, because $E_S = E_{S'}$, that G_S arises from $G'_{S'}$ by attaching v as a child to $u_1, \ldots, u_n$. Therefore, G_S has the single source $\min V(G') = \min V(G)$ and Statement 1 holds. For Statement 2, observe that the apices in S are a sub(multi)set of the apices in S'. For Statement 3, note that each inner node ν of S' is also an inner node of S and it is $H_S(\nu) \subseteq H_{S'}(\nu)$. Each arc (μ, ν) of S' where ν is an inner node is also an arc of S' and it is $H_S((\mu, \nu)) = H_{S'}((\mu, \nu))$ by construction of T. This shows Statement 3. For Statement 4, observe that each arc (μ, ν) of S where ν is a C-node

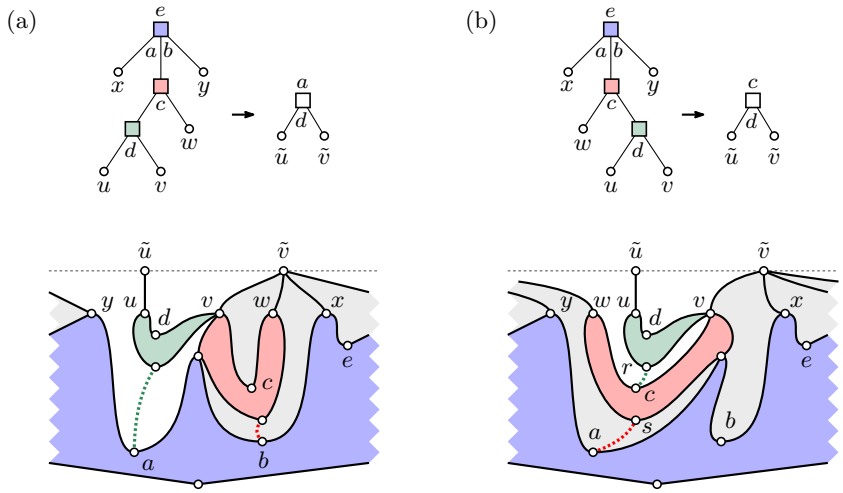


Figure 5.8: The importance of bundling correctly in the radial setting. In (a), we correctly replace, i.e., in the left-maximal tree, which corresponds to the embedding shown below. In particular, any component H with $\ell(a) < \ell(H)$ can then be merged between $\tilde{v}$ and $\tilde{u}$. If we simply replaced in an arbitrary tree such as the one shown in (b), we would assume that only components H with $\ell(b) < \ell(H)$ can be merged between $\tilde{v}$ and $\tilde{u}$, which is incorrect because $\ell(a) < \ell(b)$.

is also an arc in S'. Moreover, the references around (μ, ν) are the same in S and S'. Hence, Statement 4 holds for S by assumption for S'.

We now turn to Property 2. Obtain $\bar{\mathcal{G}}_S$ from $\bar{\mathcal{G}}'_{S'}$ as follows. Let π be a permutation so that the black leaves $u_1, \ldots, u_n$ appear in the order $u_{\pi(1)}, u_{\pi(2)}, \ldots, u_{\pi(n)}$ around the frontier of S'. For each $1 \leq i < n$ remove all attachment paths ending between the vertices $u_{\pi(i)}$ and $u_{\pi(i+1)}$. Moreover, remove all attachment paths before $u_{\pi(1)}$ and after the attachment path from p. Symmetrically, remove all attachment paths after $u_{\pi(n)}$ and before the attachment path from q. Extend an edge $e_{\pi(1)}$ from $u_{\pi(1)}$ to v. Then, for each i with $1 < i \leq n$ intersperse the edge $(u_{\pi(i)}, v)$ after $e_{\pi(i-1)}$. The linearized order of parents of v in $\bar{\mathcal{G}}_S$ then is $u_{\pi(1)}, u_{\pi(2)}, \ldots, u_{\pi(n)}$. Extend an edge from each remaining attachment path and from each non-pertinent vertex and restrict this embedding to $\bar{G}_S$.

For Statement 1 observe that $u_1, \ldots, u_n$ are consecutive around the frontiers of $\mathcal{G}'_{S'}$ and S'. The frontier of $\mathcal{G}_S$ is obtained from the frontier of $\mathcal{G}'_{S'}$ by replacing the consecutive subsequence $u_{\pi(1)}, u_{\pi(2)}, \ldots, u_{\pi(n)}$ with v and renaming all non-pertinent vertices. The frontier of S is obtained from the frontier of S' in the exact same way. There-

fore, $\mathcal{G}_S$ matches S and Statement 1 holds. Statement 2 holds vacuously for $U = \emptyset$. Consider Statement 3. Let $w \triangleright v \triangleright z$ around S. Consider leaves $x \triangleright y$ in S. If $x = w$ and $y = v$ let x' denote the unique parent of x in G and let $y' = u_{\pi(1)}$. Then $A_S(x, y)$ is obtained from $A_{S'}(x', y')$ by removing the suffix after p. If $x = v$ and $y = z$ let $x' = u_{\pi(n)}$ and let y' denote the unique parent of y in G. Then $A_S(x, y)$ is obtained from $A_{S'}(x', y')$ by removing the prefix before q. Otherwise it is $x, y \notin \{w, v, z\}$. Then let x', y' denote the unique parents of x, y in G, respectively. Then $A_S(x, y) = A_{S'}(x', y')$. In all three cases Statement 3 follows by construction of T and by assumption for S'.

Consider Statement 4. Let $w \triangleright v \triangleright z$ around S. Consider leaves $x \triangleright y$ in S. If $x = w$ and $y = v$ let x' denote the unique parent of x in G. By construction of $\bar{\mathcal{G}}_S$ the graph $I(j, k)$ can be merged into $\mathcal{G}_S$ between x and y if and only if $I(j, k-1)$ can be merged into $\mathcal{G}'_{S'}$ between x' and $u_{\pi(1)}$. Using Statement 4 on S', $I(j, k-1)$ can be merged into $\mathcal{G}'_{S'}$ between x' and $u_{\pi(1)}$ if and only if the space h' between x' and $u_{\pi(1)}$ in S' satisfies $h' < j$. By construction of S, the space between x and y is h' and Statement 4 holds. If $x = v$ and $y = z$ let y' denote the unique parent of y in G. By construction of $\bar{\mathcal{G}}_S$ the graph $I(j, k)$ can be merged into $\mathcal{G}_S$ between x and y if and only if $I(j, k-1)$ can be merged into $\mathcal{G}'_{S'}$ between $u_{\pi(n)}$ and y'. Using Statement 4 on S', $I(j, k-1)$ can be merged into $\mathcal{G}'_{S'}$ between $u_{\pi(n)}$ and y' if and only if the space h' between $u_{\pi(n)}$ and y' in S' satisfies $h' < j$. By construction of S, the space between x and y is h' and Statement 4 holds. Otherwise it is $x, y \notin \{w, v, z\}$. Let x', y' denote the unique parents of x, y in G. By construction of $\bar{\mathcal{G}}_S$ the graph $I(j, k)$ can be merged into $\mathcal{G}_S$ between x and y if and only if $I(j, k-1)$ can be merged into $\mathcal{G}'_{S'}$ between x' and y'. Using Statement 4 on S', $I(j, k-1)$ can be merged into $\mathcal{G}'_{S'}$ between x' and y' if and only if the space h' between x' and y' in S' satisfies $h' < j$. By construction of S, the space between x and y is h' and Statement 4 holds.

For Statement 5 consider a level-planar embedding $\mathcal{E}$ of G. Let $\mathcal{E}'$ denote the restriction of $\mathcal{E}$ to $G' = G[k-1]$. By Statement 5 there exists an $R' \equiv T'$ such that R' matches $\mathcal{E}'$ and for each pair of vertices $x' \triangleright y'$ in $\mathcal{E}'$ if $I(j, k-1)$ can be merged into $\mathcal{E}'$ between x' and y', then the space h' between x' and y' in R' satisfies $h' < j$. Obtain R from $M(R')$ by replacing $u_1, \ldots, u_n$ with v without reordering. Consider vertices $x \triangleright y$ in $\mathcal{E}$. If $x = v$ let x' denote the parent of v that is the last black leaf around $M(R')$, i.e., either $u_{\tau(n)}$ or $u_{\tau(1)}$. Otherwise, let x' denote the unique parent of x in G. If $y = v$ let y' denote the parent of v that is the first black leaf around $M(R')$, i.e., either $u_{\tau(1)}$ or $u_{\tau(n)}$. Otherwise, let y' denote the unique parent of y in G. In the proof of Statement 4 we have shown that the space between x and y in R equals the space between x' and y' in $M(R')$. Note that by construction of $M(R')$, the space between the last white leaf and the first black leaf around R' equals the space between the last white leaf and the first black leaf around $M(R')$. Likewise, the space between the last black leaf and the first white leaf around R' equals the space between the last black leaf and the first white leaf around $M(R')$. Finally, for each pair of white

leaves $a \triangleright b$ in R' it is also $a \triangleright b$ in $M(R')$, and the space between them is the same in R' and $M(R')$. So the space h between x and y in R equals the space between x' and y' in R'. If $I(j,k)$ can be merged into $\mathcal{E}$ between x and y then $I(j,k-1)$ can be merged into $\mathcal{E}'$ between x' and y'. Then the space h' between x' and y' in R' satisfies $h' < j$. This equals the space between x and y in R, which shows Statement 5. □

Now consider the remaining case (b). Recall that the counter-clockwise order of arcs around χ is $\beta, \ldots, \gamma, \delta, \ldots, \alpha$, where the endpoints of $\beta, \ldots, \gamma$ are black and the endpoints of $\delta, \ldots, \alpha$ are white. In this case,the parent of χ is black, i.e., the sequence $\beta, \ldots, \gamma$ includes the parent arc of χ.

Obtain the *left-maximal tree* $M_l(T')$ of T' from T' as follows. Consider each black node μ on the path from χ to the root of T' (that is, each node on the path from χ to the root of T', except for χ itself). Let (μ,ν) denote the child arc of μ that lies on the path from χ to the root of T'. If μ is a P-node, reorder the child arcs of μ so that (μ,ν) becomes the first child arc in the linearized order of child arcs of μ. If μ is a C-node, let α and β denote the predecessor and successor of (μ,ν) around μ. If α is the parent arc of μ, leave μ unchanged. If β is the parent arc of μ, reverse μ. If both α and β are child arcs of μ, let p denote the apex between α and (μ,ν), and let q denote the apex between (μ,ν) and β. If $\ell(q) < \ell(p)$, reverse μ. Otherwise, leave μ unchanged. Obtain the *right-maximal tree* $M_r(T')$ of T' from $M_l(T')$ by reversing the order of arcs around each node on the path from χ to the root of $M_l(T')$, except for χ itself.

Let p_l denote the apex before the first black leaf in $M_l(T')$, and let q_l denote the apex after the last black leaf in $M_l(T')$. Likewise, let p_r denote the apex before the first black leaf in $M_r(T')$, and let q_r denote the apex after the last black leaf in $M_r(T')$. Finally, let p denote the apex before the first black leaf in T', and let q denote the apex after the last black leaf in T'. By construction of $M_l(T')$ it is $\ell(p_l) \le \ell(p)$. By construction of $M_r(T')$ it is $\ell(q_r) \le \ell(q)$. Moreover, it is $\ell(p_l) \le \ell(p)$ and $\ell(q_l) \le \ell(q)$, or $\ell(p_r) \le \ell(p)$ and $\ell(q_r) \le \ell(q)$. In the former case, define $M(T') = M_l(T')$, otherwise define $M(T') = M_r(T')$.

Obtain T from $M(T')$ as follows. Let p denote the apex between the unique vertices $a \triangleright b$ around $M(T')$ where a is white and b is black, and let q denote the apex between the unique vertices $c \triangleright d$ around $M(T')$ where c is black and d is white. Delete the arcs $\beta, \ldots, \gamma$ and, from the resulting forest, every tree that does not contain χ. Create a new leaf v and insert the arc (χ, v) between α and δ around χ. Define the apex between α and (χ, v) as p, and define the apex between (χ, v) and β as q. If β is the parent arc of χ let r denote the reference between $\alpha = (\chi,\nu)$ and β around χ in T'. Let $\hat{r}$ denote the root of the tree of $P_{T'}$ to which r belongs. Observe that $\hat{r}$ is annotated with p. If ν is a C-node, move $\hat{r}$ to be the reference before $\alpha = (\chi,\nu)$ around ν in T. Symmetrically, if γ is the parent arc of χ let o denote the reference between γ and $\delta = (\chi, \xi)$ around χ in T'. Let $\hat{o}$ denote the root of the tree of $P_{T'}$ to

which o belongs. Observe that $\hat{o}$ is annotated with q. If ξ is a C-node, move $\hat{o}$ to be the reference after $\delta = (\chi, \xi)$ around ξ in T. Finally, define $P_T = P_{T'}$.

Lemma 28. *Let G be a k-level graph that arises from $G' = G[k-1]$ by unary bundling $U = \{u_1, u_2, \ldots, u_n\}$ into v. Let T' be a PC-tree that U-represents G' such that the parent of χ is black. Let T be the PC-tree obtained from T' by replacing $u_1, \ldots, u_n$ with v. Then T represents G.*

Proof. Let $S \equiv T$. Then there exists an $S' \equiv T'$ such that S is obtained from S' by replacing $u_1, \ldots, u_n$ with v. In particular, the linearized order of white leaves around S is obtained from the linearized order of white leaves around S' by renaming the non-pertinent vertices. However, the choice of S' is not unique in general because the linearized order of black leaves around S' is not fixed by S. To handle this degree of freedom we define a canonical tree $M(S')$ as follows. To this end, we first define $M_l(S')$ and $M_r(S')$ as follows. Consider the case that the orientation of χ is the same in S' and T'. Obtain $M_l(S')$ from S' by performing equivalence transformations only on black nodes so that the linearized order of black leaves around $M_l(S')$ equals the linearized order of black leaves around $M_l(T')$. Obtain $M_r(S')$ from S' by performing equivalence transformations only on black nodes so that the linearized order of black leaves around $M_r(S')$ equals the linearized order of black leaves around $M_r(T')$. Now consider the other case, namely that the orientation of χ is opposite in S' and T'. Obtain $M_l(S')$ from S' by performing equivalence transformations only on black nodes so that the linearized order of black leaves around $M_l(S')$ equals the reverse linearized order of black leaves around $M_r(T')$. Obtain $M_r(S')$ from S' by performing equivalence transformations only on black nodes so that the linearized order of black leaves around $M_r(S')$ equals the reverse linearized order of black leaves around $M_l(T')$.

Let p_l denote the apex before the first black leaf in $M_l(S')$, and let q_l denote the apex after the last black leaf in $M_l(S')$. Likewise, let p_r denote the apex before the first black leaf in $M_r(S')$, and let q_r denote the apex after the last black leaf in $M_r(S')$. Finally, let p denote the apex before the first black leaf in S', and let q denote the apex after the last black leaf in S'. It is $\ell(p_l) \leq \ell(p)$ and $\ell(q_l) \leq \ell(q)$, or $\ell(p_r) \leq \ell(p)$ and $\ell(q_r) \leq \ell(q)$. In the former case, define $M(S') = M_l(S')$. Otherwise, define $M(S') = M_r(S')$. From now on assume $S' = M(S')$ without loss of generality. Importantly, this implies $E_S = E_{S'}$.

We first show that Property 1 holds. For Statement 1 observe that $G'_{S'}$ has the single source $\min V(G')$, it is $E_S = E_{S'}$ and every level-k vertex of G has a parent in G'. Therefore, G_S has the single source $\min V(G') = \min V(G)$ and Statement 1 holds. For Statement 2, observe that the apices in S are a sub(multi)set of the apices in S'. For Statement 3, note that each arc (μ, ν) of S where ν is an inner node is also an arc of S' and it is $H_S(\nu) = H_{S'}(\nu)$. Furthermore, unless (μ, ν) is adjacent

to (χ, v) in S, it is $H_S((\mu,\nu)) = H_{S'}((\mu,\nu))$. Finally, if (μ,ν) is adjacent to (χ, v), then $\max H_S((\mu,\nu)) \leq \max H_{S'}((\mu,\nu))$ by construction of S and by Statement 3 for S'. This shows Statement 3. We explicitly made sure that Statement 4 holds for the arcs α and δ, which are adjacent to the arc (χ, v) during the construction of T. All remaining arcs of S are also arcs of S' and the exact same references and apices are incident to these arcs in S and S'. Statement 4 holds for these arcs by assumption for S'.

We now turn to Property 2. Construct $\bar{\mathcal{G}}_S$ as in the proof of Lemma 27. Show Statement 1, 2, 3 and 4 in the exact same way. For Statement 5, consider a level-planar embedding $\mathcal{E}$ of G. Let $\mathcal{E}'$ denote the restriction of $\mathcal{E}$ to $G' = G[k-1]$. By Statement 5 there exists an $R' \equiv T'$ such that R' matches $\mathcal{E}'$ and for each pair of vertices $x' \triangleright y'$ in $\mathcal{E}'$ if $I(j, k-1)$ can be merged into $\mathcal{E}'$ between x' and y', then the space h' between x' and y' in R' satisfies $h' < j$. Obtain R from $M(R')$ by replacing $u_1, \ldots, u_n$ with v. If $I(j,k)$ can be merged into $\mathcal{E}$ between x and y then $I(j, k-1)$ can be merged into $\mathcal{E}'$ between the corresponding x' and y'. By Statement 5, then the space h' between x' and y' in R' satisfies $h' < j$. By construction of $M(R')$ the space h'_M between x' and y' in $M(R')$ satisfies $h'_M \leq h'$. By construction of R from $M(R')$ the space h between x and y in R is h'_M. So, $h = h'_M \leq h' < j$, and Statement 5 holds. □

Together, Lemmas 26, 27 and 28 give the following corollary.

Corollary 5. *Let G be a k-level graph that arises from $G' = G[k-1]$ by unary bundling $u_1, u_2, \ldots, u_n$ into v. Let T' be a PC-tree that represents G' and let T be the PC-tree obtained from T' by bundling $u_1, \ldots, u_n$ into v. Then T represents G.*

5.11 General Bundle

The remaining operation is the general bundle, i.e., G arises from $G' = G[k-1]$ by bundling the ancestors $u_1, u_2, \ldots, u_n$ of v, where $u_1, \ldots, u_n$ belong to two or more distinct connected components of G'. Recall Theorem 7, which states that either all connected components of G' are v-singular, or the ancestors of v belong to exactly two connected components. We treat the former case in Section 5.11.1. The latter case is treated in Sections 5.11.2 and 5.11.3. As a preliminary step, we investigate how embeddings of two distinct connected components can be merged into a single embedding. Let $G = G_1 \cup G_2$ be a k-level graph such that G_1 and G_2 are connected. Recalling that G is regularized, assume $\ell(G_1) < \ell(G_2)$ without loss of generality. Consider a level-planar embedding $\mathcal{E}$ of G. For $i = 1, 2$, the embedding $\mathcal{E}$ induces a level-planar embedding $\mathcal{E}_i$ of G_i. Assume that Property 1 and 2 hold for G_i. Because G_i is connected this is the case if and only if these properties hold for $G_i[k-1]$. Let T_i denote a PC-tree that represents G_i and let $R_i \equiv T_i$ as in Property 2 Statement 5.

Because $\mathcal{E}$ is level planar, the level-k vertices of G_i are consecutive in the frontier of $\mathcal{E}$. Let $a, y, \ldots, x, b, \ldots, a$ denote the frontier of $\mathcal{E}$ where $b, \ldots, a$ are the vertices of G_1 and $y, \ldots, x$ are the vertices of G_2.

Lemma 29. *The space h between a and b in R_1 satisfies $h < \ell(G_2)$, and the apex between x and y in R_2 is ς.*

Proof. The existence of $\mathcal{E}$ implies that $I(\ell(G_2), k)$ can be merged into $\mathcal{E}_1$ between a and b (Lemma 6 from Chapter 3). From Property 2 Statement 5 it follows that $h < \ell(G_2)$. Symmetrically, we obtain that the space h' between x and y in R_2 satisfies $h' < \ell(G_1)$. With $\ell(G_1) < \ell(G_2)$ and the fact that every apex in R_2 other than ς is a vertex of G_2 this implies that the apex between x and y in R_2 is ς. □

Consider PC-trees $S_1 \equiv T_1, S_2 \equiv T_2$. Let $a \triangleright b$ denote leaves of S_1 and let p denote the apex between a and b in S_1. Define $p' = \min V(G_1)$ if $p = \varsigma$, or $p' = p$ if $p \neq \varsigma$. Further, let $x \triangleright y$ denote leaves of S_2 and let q denote the apex between x and y in S_2. We say that *the merge candidate a, b, x, y is feasible for S_1, S_2* if there exists a level-planar embedding of the graph $G_{S_1} + G_{S_2} + (p', \min V(G_2))$ whose frontier is $a, y, \ldots, x, b, \ldots, a$, where $b, \ldots, a$ are the vertices of G_1 and $y, \ldots, x$ are the vertices of G_2, and whose restriction to G_{S_1}, G_{S_2} is $\mathcal{G}_{S_1}, \mathcal{G}_{S_2}$, respectively.

Lemma 30. *If $\ell(p) < \ell(G_2)$ and $q = \varsigma$, then the merge candidate a, b, x, y is feasible for $\mathcal{G}_{S_1}$ and $\mathcal{G}_{S_2}$.*

Proof. Justified by Property 2 Statement 4 and $\ell(p) < \ell(G_2)$, intersperse $\mathcal{G}_{S_2}$ (split between x and y) into $\bar{\mathcal{G}}_{S_1}$ between a and b. Symmetrically, Property 2 Statement 4 and $\ell(\varsigma) = 0 < \ell(G_1)$ justifies interspersing $\bar{\mathcal{G}}_{S_1}$ (split between a and b) into $\mathcal{G}_{S_2}$ between x and y. Next, intersperse the edge $(p', \min V(G_2))$ after the augmentation path from p to $\bar{p}$. Finally, remove all augmentation paths. We have obtained a level-planar embedding of the graph $G_{S_1} + G_{S_2} + (p', \min V(G_2))$. By construction, its frontier is $a, y, \ldots, x, b, \ldots, a$, and the restriction to G_{S_1}, G_{S_2} is $\mathcal{G}_{S_1}, \mathcal{G}_{S_2}$, respectively. □

By Property 1 Statement 2 ς occurs at most once in R_2. So the choice of x and y is implied as the unique pair of leaves $x \triangleright y$ of R_2 such that the apex between x and y is ς. We then simply say that *the merge candidate a, b is feasible for S_1, S_2*.

5.11.1 Many v-Singular Components

In this section we treat the case that G is v-singular, i.e., G is a connected k-level graph with a single vertex v on level k. Let $G_1, G_2, \ldots, G_n$ denote the components of G that are connected by v such that $\ell(G_i) < \ell(G_{i+1})$ for $1 \leq i < n$. Moreover, for

each i with $1 \leq i \leq n$ let T_i denote a PC-tree that represents G_i, let $G_i' = G_i[k-1]$ and let T_i' denote a PC-tree that represents G_i'.

Obtain T as follows. Let p denote the unique apex stored in T_1. If $\ell(p) \geq \ell(G_2)$, or if any $T_2, \ldots, T_n$ does not contain ς as an apex, define T as the null tree. Otherwise, define T as the PC-tree that consists of the root C-node μ and a single leaf v connected by an arc (μ, v). Define the apex between (μ, v) and (μ, v) as p. Define P_T as follows. First, take the union of each P_{T_i} for $1 \leq i \leq n$. Then, create a new reference r and annotate it with p. Next, add the edge (v, r) with timestamp k for each root v of P_{T_i} annotated with ς, where $1 < i \leq n$. Note that there are at most two such roots per P_{T_i}. This means that any augmentation edge of G_{T_i} previously beginning at ς will from now on begin at p instead. Finally, add $\langle \min V(G_i), k \rangle$ to $Z(r)$ for each i with $1 < i \leq n$.

Lemma 31. *Let G be a v-singular k-level graph that arises from $G' = G[k-1]$ by bundling. Let T be the PC-tree obtained from $T_1, \ldots, T_n$ by merging. Then T represents G.*

Proof. If $\ell(p) \geq \ell(G_2)$, or if any $T_2, \ldots, T_n$ does not contain ς as an attachment, then Lemma 29 gives that G is not level planar. We defined T as the null tree in this case, so T represents G.

Otherwise, T is not the null tree. Let $S \equiv T$, i.e., $S = T$. By construction of S it is $E_S = \{e(P_S, z) \mid z \in Z(r)\} \cup \left(\bigcup_{1 \leq i \leq n} E_{T_i}\right)$, where r denotes the newly created reference. In particular, $E_{T_i} \subset E_S$.

We first show that Property 1 holds. For Statement 1, observe that for each $1 \leq i \leq n$ the graph G_{T_i} has the single source $\min V(G_i)$. The graph G_T is the union of all G_{T_i} together with edges that connect the single source of G_{T_i} for $1 < i \leq n$ to the vertex p of G_{T_1}. So, G_T is has the single source $\min V(G_1) = \min V(G)$ and Statement 1 holds. Statement 2 holds because T contains a single apex. Statement 3 and Statement 4 hold because T does not contain an arc whose endpoint is an inner node.

We now turn to Property 2. Let $S \equiv T$, i.e., $S = T$. Define $\bar{\mathcal{G}}_T$ as follows. Start with $\bar{\mathcal{G}}_{T_1}$. Intersperse $\mathcal{G}_{T_2}$ into $\bar{\mathcal{G}}_{T_1}$ after the augmentation path from p to $\bar{p}$. Then intersperse the edge $(p', \min V(G_2))$ after that augmentation path as well. This creates two separate instances of the vertex v, one from $\bar{\mathcal{G}}_{T_1}$ (call it v_1) and one from $\mathcal{G}_{T_2}$ (call it v_2). Reroute the edges with endpoint v_2 towards v_1 as follows. Let $u_1, u_2, \ldots, u_m$ denote the parent vertices of v in the linearized order in which they occur around v in $\mathcal{G}_{T_2}$. For $j = m, m-1, \ldots, 1$ intersperse the edge (u_j, v_1) after the edge (u_j, v_2), then delete (u_j, v_2). Finally, delete v_2. For $i = 3, \ldots, n$ intersperse $\mathcal{G}_{T_i}$ into this embedding after the augmentation path from p to $\bar{p}$ in a similar fashion and denote the result by $\bar{\mathcal{G}}_T$. Statement 1 holds because G is v-singular. Statement 2 holds vacuously for $U = \emptyset$. Statement 3 holds by construction of $\bar{\mathcal{G}}_T$. For Statement 4, observe that if $I(j, k)$ can be merged into $\mathcal{G}_T$ between v and v, then $I(j, k)$ can be

merged into $\mathcal{G}_{T_1}$, and then the space h_1 between v and v in T_1 satisfies $h_1 < j$. By construction the space h between v and v in T equals h_1. So if $h \geq j$, then $I(j,k)$ cannot be merged into $\mathcal{G}_T$ between v and v. On the other hand, the existence of the attachment path from p to $\bar{p}$ in $\bar{\mathcal{G}}_T$ shows that if $h < j$ then $I(j,k)$ can be merged into $\mathcal{G}_T$ between v and v. For Statement 5, consider a level-planar embedding $\mathcal{E}$ of G. Then T trivially matches $\mathcal{E}$. Let $\mathcal{E}_1$ denote the restriction of $\mathcal{E}$ to G_1. If $I(j,k)$ can be merged into $\mathcal{E}$ between v and v, then $I(j,k)$ can be merged into $\mathcal{E}_1$ between v and v. Applying Statement 4, this means that the only space h_1 stored in T_1 satisfies $h_1 < j$. By construction the space h between v and v in T equals h_1. So $h = h_1 < j$ and Statement 5 holds. □

5.11.2 Independent Merging

In the remaining cases G' consists of just two connected components. Let T_1, T_2 denote PC-trees that represent G_1, G_2, respectively. Consider a level-planar embedding $\mathcal{E}$ of G (if one exists). It induces level-planar embeddings $\mathcal{E}_1, \mathcal{E}_2$ of G_1, G_2. Let $R_2 \equiv T_2$ be as in Property 2 Statement 5. Lemma 29 gives that R_2 contains ς as an apex, and, together with the fact that $\mathcal{E}$ is level-planar, that v is incident to the apex ς. We reflect this necessary condition by updating T_2 so that v is incident to the apex ς as follows. If T_2 does not contain ς as an apex, then G is not level planar, and, correspondingly, we define T as the null tree. Otherwise, by Property 1 Statement 2 T_2 contains a unique pair of consecutive child arcs α, β so that the apex between them is ς. By Property 1 Statement 3 α, β are child arcs of the root of T_2. If $\alpha = \beta$, let L denote the set of leaves of T_2. Obtain T_2^+ from T_2 by $(L \setminus \{v\})$-updating. Note that during this update the root of T_2 behaves like a white leaf. If $\alpha \neq \beta$, let $\gamma \in \{\alpha, \beta\}$ denote the arc so that v is a leaf in the subtree of T_2 rooted at the endpoint of γ, and let $\delta \in \{\alpha, \beta\}$ such that $\delta \neq \gamma$. Let L denote the set of leaves of T_2 that appear in the subtree of T_2 rooted at the endpoint of δ. Obtain T_2^+ from T_2 by $(L \cup \{v\})$-updating. If T_2^+ is the null tree, then G is not level planar and, accordingly, we define T to be the null tree. Otherwise, if T_2^+ is not the null tree, then v is a child of the root of T_2^+. Assume without loss of generality that the apex before v in T_2^+ is ς, reversing the root of T_2^+ if necessary.

Consider the case that G_1 is v-singular. Let p denote the one apex stored in T_1. If $\ell(p) \geq \ell(G_2)$, define T as the null tree. Otherwise, it is $\ell(p) < \ell(G_2)$ and T_2^+ contains ς as an apex. Then obtain T from T_2^+ as follows. Replace the apex ς with p. Start defining P_T as the union of P_{T_1} and P_{T_2}. Create a new reference o, annotate o with p and set $Z(o) = \{\langle \min V(G_2), k\rangle\}$. Let (μ, ν) denote the arc that precedes (μ, v) around the root μ of T_2^+. If v is a C-node, let r denote the reference before (μ, ν) around v. Note that r is annotated with ς. Replace r with o and add the edge (r, o) with timestamp k to P_T.

Lemma 32. *Let G be a binary k-level graph so that G_1 is v-singular. Further, let T be the PC-tree obtained from T_1, T_2^+ by merging. Then T represents G.*

Proof. If T is the null tree, then the correctness follows from Lemma 29. Consider the other case, namely that T is not the null tree and let $S \equiv T$. Let $S_2 \equiv T_2^+$ so that S is obtained by merging T_1 and S_2. Then $E_S = E_{T_1} \cup E_{S_2} \cup \{e(P_S, \langle \min V(G_2), k \rangle)\}$.

We first show that Property 1 holds. For Statement 1 observe that G_{T_1} has the single source $\min V(G_1)$ and G_{S_2} has the single source $\min V(G_2)$. The graph G_S is the union of G_{T_1} and G_{S_2} together with, because $(\min V(G_2), k) \in Z(o)$, an edge that connects a vertex in G_1 to $\min V(G_2)$. So, G_S has the single source $\min V(G_1)$, which, because $\ell(G_1) < \ell(G_2)$ is $\min V(G)$. For Statement 2, observe that T_2^+ contains ς as an apex once and in the construction of T it is replaced with p. Depending on whether $p = \varsigma$ or $p \neq \varsigma$, the tree S contains ς as an apex once or not at all. For Statement 3, observe that with $\ell(p) < \ell(G_2)$ the claim follows because Statement 3 holds for S_2. Statement 4 holds by construction of P_T.

We now turn to Property 2. Obtain $\bar{\mathcal{G}}_S$ as follows, starting with $\bar{\mathcal{G}}_{T_1}$. First, duplicate the (unique) augmentation path ending in $\bar{p}$ in $\bar{\mathcal{G}}_{T_1}$. Merge $\bar{\mathcal{G}}_{S_2} - \varsigma$ into $\bar{\mathcal{G}}_{T_1}$ between these two duplicates. Note that this creates two separate instances of the vertex v, one from $\bar{\mathcal{G}}_{T_1}$ (call it v_1), and one from $\bar{\mathcal{G}}_{S_2}$ (call it v_2). Depending on whether ς is the apex before or after v in S_2, either $v_1 \triangleright \bar{p} \triangleright v_2$ or $v_2 \triangleright \bar{p} \triangleright v_1$ in this intermediate embedding. In both cases, delete the middle attachment path with endpoint $\bar{p}$, the other attachment path remains in the embedding. Then, in the first case, reroute all edges with endpoint v_1 to v_2. In the second case, reroute all edges with endpoint v_2 to v_1. See the proof of Lemma 31 for a more detailed description of how this rerouting can be achieved by interspersing.

Statement 1 follows because Statement 1 holds for $\bar{\mathcal{G}}_{T_1}$ and $\bar{\mathcal{G}}_{S_2}$, and by explicit construction of $\bar{\mathcal{G}}_S$ from those embeddings. Statement 2 holds vacuously for $U = \emptyset$. Statement 3 holds by construction. For Statement 4, consider vertices $x \triangleright y$ in $\mathcal{G}_S$. If the apex between x and y in S is not p, then Statement 4 holds for $\mathcal{G}_S$ because it holds for $\mathcal{G}_{S_2}$. If the apex between x and y in S is p, then $I(j,k)$ can be merged into $\mathcal{G}_S$ between x and y if and only if $I(j,k)$ can be merged into $\mathcal{G}_{T_1}$ between v and v (i.e., if the space h_1 between v and v in T_1 satisfies $h_1 < j$) *and* into $\mathcal{G}_{S_2}$ between x and y (i.e., the space h_2 between x and y in S_2 satisfies $h_2 < j$). It is $h_1 = \ell(p)$ and $h_2 = 0$, so merging is possible if $\ell(p) < j$. Note that $\ell(p)$ is the space between x and y in S. Then Statement 4 holds.

For Statement 5, consider a level-planar embedding $\mathcal{E}$ of G. Let $\mathcal{E}_2$ denote the restriction of $\mathcal{E}$ to G_2. By construction of T_2^+ and by Statement 5 there exists an $R_2 \equiv T_2^+$ such that R_2 matches $\mathcal{E}_2$ and for each pair of vertices $x \triangleright y$ in $\mathcal{E}_2$ if $I(j,k)$ can be merged into $\mathcal{E}_2$ between x and y, then the space h_2 between x and y in R_2 satisfies $h_2 < j$. Obtain $R \equiv T$ by merging T_1 and R_2. The frontiers of R, R_2 and $\mathcal{E}, \mathcal{E}_2$ are identical, so R

matches $\mathcal{E}$. Consider vertices $x \triangleright y$ in $\mathcal{E}$. If $I(j,k)$ can be merged into $\mathcal{E}$ between x and y, then $I(j,k)$ can be merged into $\mathcal{E}_1$ between v and v, and into $\mathcal{E}_2$ between x and y. Then the space h_1 between v and v in T_1 satisfies $h_1 < j$, and the space h_2 between x and y in R_2 satisfies $h_2 < j$. If the apex between x and y in R is p, it is $h_1 = \ell(p)$ and $h_2 = 0$. Otherwise it is $h_1 = \ell(p)$ and $h_2 \geq \ell(G_2) > \ell(p)$ equals the space between x and y in R. So Statement 5 holds. □

For the remainder of this section, assume that G_1 is not v-singular. Let T_1' denote the PC-tree that U_1-represents G_1', where U_1 is the set of ancestors of v in G_1. That is, T_1 is obtained from T_1' by replacing U_1 with v. Consider the case that G_2 is v-singular and that T_1' contains two consecutive black leaves so that the apex p between them satisfies $\ell(p) < \ell(G_2)$. Note that because G_2 is v-singular it is $T_2^+ = T_2$. Take T as T_1. Start defining P_T as the union of P_{T_1} and $P_{T_2^+}$. Next, create a new reference o, annotate o with p and set $Z(o) = \{\langle \min V(G_2), k\rangle\}$. Then, add the edge (v,o) with timestamp k for each root v of $P_{T_2^+}$ annotated with ς.

Lemma 33. *Let G be a binary k-level graph so that G_2 is v-singular and that T_1' contains two consecutive black leaves so that the apex p between them satisfies $\ell(p) < \ell(G_2)$. Further, let T be the PC-tree obtained from T_1, T_2^+ by merging. Then T represents G.*

Proof. Let $S \equiv T$. Let $S_1 \equiv T_1$ denote the unique PC-tree so that S is obtained by merging S_1 and T_2^+. Then $E_S = E_{S_1} \cup E_{T_2^+} \cup \{e(P_S, (\min V(G_2), k))\}$.

We first show Property 1. Statement 1 holds as G_{S_1} has the single source $\min V(G_1)$, the graph G_{T_2} has the single source $\min V(G_2)$, and the edge $e(P_S, (\min V(G_2), k))$ connects a vertex of G_{S_1} to the source of $G_{T_2^+}$. Statement 2, Statement 3 and Statement 4 hold for S because they hold for S_1.

We now turn to Property 2. Define $\bar{\mathcal{G}}_S$ as follows. Let $S_1' \equiv T_1'$ denote a PC-tree so that S_1 arises from S_1' by replacing U_1 with v without reordering. By assumption T_1' contains two black leaves $x \triangleright y$ so that the apex p between them satisfies $\ell(p) < \ell(G_2)$. Then S_1' also contains the apex p and two black leaves $x' \triangleright y'$ so that the apex between them is p, which satisfies $\ell(p) < \ell(G_2)$. Let T_2' denote the PC-tree that represents G_2'. Obtain $\bar{\mathcal{G}}'$ by merging $\bar{\mathcal{G}}_{T_2'}$ into $\bar{\mathcal{G}}_{S_1'}$ between x' and y'. Obtain $\bar{\mathcal{G}}_S$ from $\bar{\mathcal{G}}'$ as in the proofs of Lemma 27 and Lemma 28. Observe that the restriction of $\bar{\mathcal{G}}_S$ to G_{S_2} is $\bar{\mathcal{G}}_{S_2}$. Moreover, observe that $\bar{\mathcal{G}}_S$ is obtained by inserting $\mathcal{G}_{T_2^+}$ into an internal face of $\bar{\mathcal{G}}_{S_1}$. All statements of Property 2 then translate directly from G_1 and S_1 to G and S. □

5.11.3 Interdependent Merging

The remaining case is that G_1 is not v-singular and either G_2 is not v-singular, or G_2 is v-singular and in T_1' there do not exist two consecutive black leaves so that the

apex p between them satisfies $\ell(p) < \ell(G_2)$. We then say that G_1 and G_2 are *interdependent*. Consider the linearized order of parent edges of v in a level-planar embedding $\mathcal{E}$ of G. The edges that belong to G_2 appear consecutively in this linearized order because G_1 is not v-singular. The edges that belong to G_1 appear consecutively in this linearized order because either G_2 is not v-singular, or G_2 is v-singular but in T_1' there do not exist two black leaves so that the apex p between them satisfies $\ell(p) < \ell(G_2)$. Therefore, $\mathcal{E}$ can be obtained by merging $\mathcal{E}_2$ into $\mathcal{E}_1$ between two appropriate vertices, and then merging the two instances of v.

Let $R_1 \equiv T_1$ be as in Property 2 Statement 5 for $\mathcal{E}_1$. Lemma 29 gives that the space h before or after v in R_1 satisfies $h < \ell(G_2)$. We update T_1 to reflect this necessary condition as follows. Let (φ, ψ) denote the first arc in T_1 on the path from v to the root of T_1, so that

1. both $\ell(p) < \ell(G_2)$ and $\ell(q) < \ell(G_2)$, or
2. the arcs preceding and succeeding (φ, ψ) around φ are both child arcs of φ, and either $\ell(p) < \ell(G_2)$ or $\ell(q) < \ell(G_2)$ (but not both),

where p and q are the apices before and after (φ, ψ) around φ, respectively (resolving references if necessary). Note that if no such arc exists, then G is not level planar and, correspondingly, we define T as the null tree.

Lemma 34. *Let $\mathcal{E}$ be a level-planar embedding of G and let $\mathcal{E}_1$ denote its restriction to G_1. Let $R_1 \equiv T_1$ be as in Property 2 Statement 5 for $\mathcal{E}_1$. Then v is the leftmost or rightmost leaf in the subtree of R_1 rooted at ψ.*

Proof. If $\psi = v$ the claim is trivially true. Otherwise, ψ is an inner node of R_1. Consider case (1). Let (ψ, ω) denote the arc that precedes (φ, ψ) on the path from v to the root of R_1. By choice of (φ, ψ) it is $\max H_{R_1}((\psi, \omega)) \geq \ell(G_2)$. Together with Property 1 Statement 3 this gives $\ell(G_2) \leq \max H_{R_1}((\psi, \omega)) \leq \min H_{R_1}(\omega)$. Therefore, v must be either the leftmost or the rightmost leaf of the subtree of R_1 rooted at ω. Consider that the arcs that precede and succeed (ψ, ω) around ψ are both child arcs. Because we are not in case (2) we have that the apices p', q' incident to (ψ, ω) around ψ satisfy $\ell(p'), \ell(q') \geq \ell(G_2)$. Then the spaces h_1, h_2 before and after v in R_1 satisfy $h_1, h_2 \geq \ell(G_2)$, a contradiction to Lemma 29. Therefore, (φ, ψ) precedes or succeeds (ψ, ω) around ψ. Consider the case that (φ, ψ) precedes (ψ, ω) around ψ, i.e., (ψ, ω) is the leftmost child arc of ψ. Let p' denote the apex between (φ, ψ) and (ψ, ω) (resolving a reference if necessary), and let q' denote the apex between (ψ, ω) and its successor around ψ. It is $\ell(p') \in H_{R_1}((\varphi, \psi))$ and $\ell(q') \in H_{R_1}(\psi)$. So, Property 1 Statement 3 gives that $\ell(p') \leq \ell(q')$. By choice of (φ, ψ) it is $\ell(q') \geq \ell(G_2)$. Recall that v must be either the leftmost or the rightmost leaf of the subtree of T_1 rooted at ω. Together, this means that v must be the

leftmost leaf of the subtree of T_1 rooted at ψ. Because (ψ, ω) is the leftmost child arc of ψ this means that v is also the leftmost leaf of the subtree of R_1 rooted at ψ. If (φ, ψ) succeeds (ψ, ω) around ψ, a symmetric argument argument shows that v is the rightmost leaf in the subtree of R_1 rooted at ψ.

In case (2) we know $\ell(G_2) \leq \max H_{T_1}((\varphi, \psi)) \leq \min H_{T_1}(\psi)$, which suffices to show the claim. □

Motivated by Lemma 34, let L denote the set of leaves in the subtree of T_1 rooted at ψ. Obtain T_1^+ from T_1 by $(L \setminus \{v\})$-updating. Let χ denote the C-node that is newly created during this update. Observe that v is either the first or the last child of χ. Assume without loss of generality that the latter case holds true, reversing the orientation of χ if necessary. Consider T_2^+. Recall that by construction, v is a child of the root ρ of T_2^+. Let $(\rho, v), \alpha_1, \alpha_2, \ldots, \alpha_n$ denote the counter-clockwise order of arcs around ρ in T_2. The apex between α_n and (ρ, v) is ς. Let r_2 denote the apex between (ρ, v) and α_1.

Obtain T by merging T_2^+ into T_1^+ by inserting the arcs $\alpha_1, \ldots, \alpha_n$ after (χ, v) around χ. Take P_T as the union of $P_{T_1^+}$ and $P_{T_2^+}$. Define the apex between (χ, v) and α_1 as r_2. If χ is not the root, then let r_1 denote the reference between (χ, v) and the parent arc of χ in T_1^+. Then define the reference between α_n and the parent arc of χ in T as r_1. Add $\langle \min V(G_2), k \rangle$ to $Z(r_1)$. If the endpoint of α_n is a C-node ν, let o denote the reference before α_n around ν. Add the edge (o, r_1) with timestamp k to the reference network. If χ is the root, let c denote the apex after (χ, v) around χ. In T, define the apex after α_n as c. Create a new reference o', annotate it with c, and set $Z(o') = \{\langle \min V(G_2), k \rangle\}$. If the endpoint of α_n is a C-node ν, let o denote the reference before α_n around ν. Replace o with o' and add the edge (o, o') with timestamp k to the reference network. This completes the treatment of case (1).

In case (2) we continue with one final step. Observe that $p \neq q$ implies that φ is a C-node, which means that the parent of χ in T_1^+ and T is φ. Contract χ into φ as follows. Let $\beta_1 = (\varphi, \chi), \beta_2, \ldots, \beta_b$ denote the counter-clockwise order of arcs around φ. Recall that β_b, β_1 and β_2 are all child arcs of φ, and that p is the apex between β_b and β_1, and q is the apex between β_1 and β_2. Further, let $\gamma_1 = (\varphi, \chi), \gamma_2, \ldots, \gamma_g$ denote the counter-clockwise order of arcs around χ. Contract χ into φ by setting the order of arcs around φ as $\beta_2, \beta_3, \ldots, \beta_b, \gamma_2, \gamma_3, \ldots, \gamma_g$. Define the apex between β_b and γ_2 as p, and the apex between γ_g and β_2 as q. If the endpoint of γ_2 is a C-node, move the reference between γ_1 and γ_2 around χ to be the reference after γ_2 around its endpoint. Symmetrically, if the endpoint of γ_g is a C-node, move the reference between γ_g and γ_1 around χ to be the reference before γ_g around its endpoint.

Lemma 35. *Let G be a binary k-level graph so that G_1 and G_2 are interdependent. Further, let T be the PC-tree obtained from T_1^+, T_2^+ by merging. Then T represents G.*

Proof. We have already argued the correctness in the case when T is the null tree. Consider the other case, namely that T is not the null tree. For now, consider case (1). Let $S \equiv T$. Let $S_1 \equiv T_1^+, S_2 \equiv T_2^+$ denote PC-trees so that S arises from merging S_1 and S_2 without reordering. As always, $E_S = E_{S_1} \cup E_{S_2} \cup \{e(P_S, \langle \min V(G_2), k \rangle)\}$.

We first show Property 1. For Statement 1, observe $E_S = E_{S_1} \cup E_{S_2} \cup \{x, \min V(G_2)\}$, where x is a vertex of G_1. Because $G_1 + E_{S_1}$ and $G_2 + E_{S_2}$ are single-source graphs, this means that $G + E_S$ has the single source $\min V(G_1) = \min V(G)$. So, Statement 1 holds. For Statement 2, observe that ς is the apex between α_n and (ρ, ν) in T_2^+, so it is the one apex in T_2^+ that is not copied to T. Statement 2 then holds for T because it holds for T_1^+ by Lemma 25. For Statement 3, first consider the arc (φ, χ) of T. By construction of T it is $H_T((\varphi, \chi)) = H_{T_1^+}((\varphi, \chi))$ and $H_T(\chi) = H_{T_1^+}(\chi) \cup \left(H_{T_2^+}(\rho) \setminus \{\varsigma\}\right)$. By assumption for T_1^+ it is $\max H_{T_1^+}((\varphi, \chi)) \leq \min H_{T_1^+}(\chi)$. By choice of (φ, χ) during the construction of T it is $\max H_{T_1^+}((\varphi, \chi)) < \ell(G_2)$. Because every apex in T_2^+ except for ς is a vertex of G_2, it is $\ell(G_2) \leq \min \left(H_{T_2^+}(\rho) \setminus \{\varsigma\}\right)$. Hence, $\max H_T((\varphi, \chi)) \leq \min H_T(\chi)$. Next, consider the arc α_n in T, which is the rightmost child arc of χ. Let μ denote its endpoint, which is an inner node in T_2^+. It is $H_T(\mu) = H_{T_2^+}(\mu)$. Let a denote the apex between the predecessor of α_n and α_n around ρ in T_2^+. It is $H_{T_2^+}(\alpha_n) = \{\ell(a), \ell(\varsigma)\}$, so $\max H_{T_2^+}(\alpha_n) = \ell(a)$. Property 2 guarantees $a \neq \varsigma$, then $\ell(G_2) \leq \ell(a)$. By construction of T it is $H_T(\alpha_n) = \{\ell(a), \ell(x)\}$, where x is the apex after (χ, ν) in T_1^+, resolving a reference if necessary. By choice of (φ, χ) it is $\ell(x) < \ell(G_2)$. Hence, $\max H_T(\alpha_n) = \ell(a) = \max H_{T_2^+}(\alpha_n)$. By assumption for T_2^+ we have that it is $\max H_{T_2^+}(\alpha_n) \leq \min H_{T_2^+}(\mu)$, and with the previous arguments we conclude that it is $\max H_T(\alpha_n) \leq \min H_T(\mu)$. For all other arcs of T whose existence can be traced back to T_2^+ no sets change and the claim follows by assumption for T_2^+. Finally, consider an arc (μ, ν) of T whose existence can be traced back to T_1^+. It is $H_T((\mu, \nu)) = H_{T_1^+}((\mu, \nu))$. If the subtree of T rooted at T does not contain χ, then $H_T(\nu) = H_{T_1^+}(\nu)$ and the claim follows by assumption for T_1^+. If the subtree of T rooted at T contains χ, then $H_T(\nu) = H_{T_1^+}(\nu) \cup \left(H_{T_2^+}(\rho) \setminus \{\varsigma\}\right)$. By assumption for T_1^+ we have $\max H_{T_1^+}((\mu, \nu)) \leq \min H_{T_1^+}(\nu)$. By choice of (φ, χ) we have $\max H_{T_1^+}((\mu, \nu)) < \ell(G_2)$. Because every apex in T_2^+ other than ς is an apex of G_2, it is $\ell(G_2) \leq \min \left(H_{T_2^+}(\rho) \setminus \{\varsigma\}\right)$. This shows $\max H_T((\mu, \nu)) \leq \min H_T(\nu)$. This shows Statement 3 for T. Statement 4 holds by construction of P_T.

We now turn to Property 2. Obtain $\bar{\mathcal{G}}_S$ from $\bar{\mathcal{G}}_{S_1}$ and $\bar{\mathcal{G}}_{S_2}$ as follows. Define a, b as follows. If χ has the same orientation in S_1 as in T_1^+, let $a = \nu$ and let b denote the successor of ν around S_1, and define $r = q$. Otherwise, if χ has the opposite orientation in S_1 as in T_1^+, let $b = \nu$ and let a denote the predecessor of ν around S_1, and define $r = p$. In the former case, ρ also has the same orientation in S_2 as in T_2^+,

define x as the predecessor of v around S_2 and define $y = v$. In the latter case, ρ has the opposite orientation in S_2 as in T_2^+, define $x = v$ and define y as the successor of v around S_2. The apex between x and y in S_2 is then ς. Merge $\bar{\mathcal{G}}_{S_2}$ (split at the augmentation path from ς to $\bar{\varsigma}$ between x and y) into $\bar{\mathcal{G}}_{S_1}$ between a and b, as justified by Lemma 30. More precisely, if χ has the same orientation in S_1 as in T_1^+, merge before the augmentation path between a and b, which begins at the apex $r = q$. Otherwise, if χ has the opposite orientation in S_1 as in T_1^+, merge after the augmentation path between a and b, which begins at the apex $r = p$.

Statement 1 holds by construction of $\bar{\mathcal{G}}_S$. Statement 2 holds vacuously for $U = \emptyset$. Statement 3 holds by construction of $\bar{\mathcal{G}}_S$. For Statement 4, consider vertices $x \triangleright y$ in $\mathcal{G}_S$. If both x and y are vertices of G_1, then $I(j,k)$ can be merged into $\mathcal{G}_S$ between x and y if and only if $I(j,k)$ can be merged into $\mathcal{G}_{S_1}$ between x and y. By construction of $\bar{\mathcal{G}}_S$, the space between x and y in S equals the space between x and y in S_1. A similar argument holds if both x and y are vertices of G_2. This leaves exactly one pair of vertices in $\mathcal{G}_S$. Namely, if the orientation of χ is the same in S as in T, let x denote the rightmost leaf of the subtree of S rooted at α_n (or $x = v$ if G_2 is v-singular) and let y denote the successor of v around $\mathcal{G}_S$, which is a vertex of G_1. Then $I(j,k)$ can be merged into $\mathcal{G}_S$ between x and y if it can be merged into $\mathcal{G}_{S_1}$ between v and y *and* it can be merged into $\mathcal{G}_{S_2}$ between x and v. The space between v and y in S_1 is $\ell(r_1)$ and the space between x and v in S_2 is $\ell(\varsigma)$. By construction of T, the space between x and y in S is $\ell(r_1)$. A similar argument holds if the orientation of χ is opposite in S and T. This shows Statement 4.

For Statement 5, consider a level-planar embedding $\mathcal{E}$ of G. Let $\mathcal{E}_1, \mathcal{E}_2$ denote the restriction of $\mathcal{E}$ to G_1, G_2, respectively. By assumption for T_1, T_2 and by construction of T_1^+, T_2^+ there exists for $i = 1, 2$ a PC-tree $R_i \equiv T_i^+$ such that R_i matches $\mathcal{E}_i$ and for each pair of vertices $x \triangleright y$ in $\mathcal{E}_i$ if $I(j,k)$ can be merged into $\mathcal{E}_i$ between x and y, then the space h_i between x and y in R_i satisfies $h_i < j$. Obtain R by merging R_1 and R_2. Clearly $R \equiv T$, and R matches $\mathcal{E}$.

The proof of case 2 is a special case of the proof of case 1, namely for a fixed orientation of χ with respect to its parent node in T. Note that the contraction of χ into φ enforces precisely this fixed relative orientation. □

5.12 Disjoint Connected Components

Finally, we treat disjoint connected components. In the non-radial case, level planarity of the entire graph trivially reduces to level planarity of each of its connected components. This is not true in the radial setting; see Figure 5.9 for an example. Consider a level graph G and let $G_1, G_2, \ldots, G_n$ denote its disjoint connected components that contain a vertex on level k. Without loss of generality,

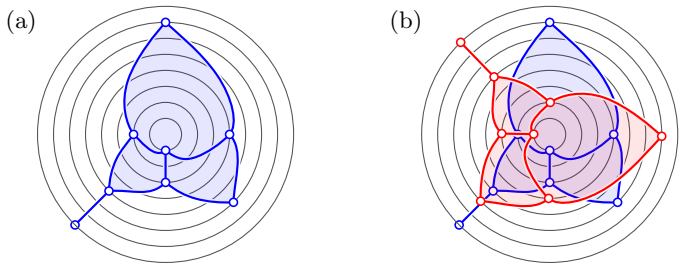


Figure 5.9: The graph shown in (a) is radial level planar, but two disjoint copies of it are not (b).

assume $\ell(G_1) < \ell(G_2) < \cdots < \ell(G_n)$. Let $T_1, T_2, \ldots, T_n$ be PC-trees such that T_i represents G_i for $1 \leq i \leq n$. Using Lemmas 29 and 30, we find that G is level planar if and only if T_1 contains an apex p with $\ell(p) < \ell(G_2)$ (from which $\ell(p) < \ell(G_i)$ follows for $2 \leq i \leq n$) and for $2 \leq i \leq n$ the PC-tree T_i contains ς as an apex. We can run this check once for every level.

5.13 Implementation in Linear Time

Our algorithm as described in the previous sections clearly runs in polynomial time. We now argue that it can even be implemented in linear time. First, recall from Theorem 7 that given a level graph G the graph $G^\times$ (on which our algorithm operates) can be computed in linear time. From now on, we refer to $G^\times$ simply as G.

To obtain the linear-time bound for PC-tree updates, Hsu and McConnell use a *potential* φ as follows [HM03, Section 4.4]. Let T be a PC-tree and let C and P denote the set of C-nodes and P-nodes in T, respectively. Moreover, let u denote the number of leaves that are to be made consecutive during all upcoming updates. Define the potential of T as $\varphi(T) = u + |C| + \sum_{x \in P}(\deg(x) - 1)$. The idea is that φ provides a budget from which the cost of updates can be paid. Hsu and McConnell show that updates (including subsequent contracts) are within the budget. We have to show that this extends to the adaptions that we made to the update operation, and also to the other operations that we defined.

First, consider growing a vertex u with children $v_1, v_2, \ldots, v_n$. This requires depositing another n credits. We can bill this cost to the edges $(u, v_1), (u, v_2), \ldots, (u, v_n)$. To cover the costs of all grow operations a number of credits linear in the size of the input graph need to be deposited. Contracting and pruning are constant-time operations. To implement our adapted update operation, observe that all references

and apices that need to be handled are stored incident to the terminal path. Therefore, the overhead incurred by the treatment of references and apices is linear in the length p of the terminal path. Recall from Lemma 1 that updating requires $O(|U|+p)$ time. So the overhead is linear in the cost of the update itself.

For the unary bundle, we need to inspect the black neighbors of χ to find out which one of the three cases applies. There are at most $|U|$ black neighbors, so again this is bounded by the cost of the update that creates these black neighbors. The first two cases of the unary bundle, namely those treated by Lemmas 26 and 27, are constant-time operations. The third case, namely the one treated by Lemma 28, requires computing the left-maximal and right-maximal trees. The running time of this computation is proportional to the size of subtree induced by the black nodes. Because this entire subtree is then replaced by a single leaf v we can pay for this running time using the PC-tree budget.

Now consider the general bundle. Merging v-singular components $G_1, G_2, \ldots, G_n$ is clearly feasible in $O(n)$ time. In the remaining cases, the updates involved in the computation of T_1^+ and T_2^+ are covered by the PC-tree budgets. Independent merging as in Lemma 32 is a constant-time operation. Independent merging as in Lemma 33 requires us to inspect the black neighbors of χ in T_1', which we can bill to the preceding update as we did before. During an interdependent merge as described by Lemma 35, the cost of walking up in T_1 to compute T_1^+ can be forward-billed to the subsequent update. Once T_1^+ and T_2^+ are computed, interdependent merging is a constant-time operation. The budget of the merged tree T is obtained by combining the budgets of T_1^+ and T_2^+ plus, possibly, a constant number of credits.

To handle disjoint connected components, we maintain for each PC-tree T a list of apices stored in T that is sorted with respect to the level of each apex. Recall that new apices are only created during a grow, at which point the apex can simply be appended to the list. We store for each apex a pointer into the list so that when an apex is removed from the tree it can be removed from the list in constant time. Furthermore, the value $\ell(G_i)$ can easily be maintained for each connected component G_i. The check described in Section 5.12 can then be implemented in constant time, so running it once for each level incurs at most linear additional running time.

Let T denote the PC-tree that represents G. To compute a level-planar embedding of G in linear time, we need to compute E_T in linear time. First, recall that reversing C-nodes involves performing a reference flip of that node and all of its C-node children. For our algorithm to run in linear time, reversing a C-node needs to be a constant-time operation. To achieve this, we store the references incident to an arc (λ, μ) around μ not in the circular list of arcs around μ, but in the one around λ. When asking for the references incident to (λ, μ) around μ, we then have to query the circular list of arcs around λ instead. This only takes a constant amount of additional running time. The benefit of this is that reversing a C-node, which reverses the

circular list of arcs around it, automatically performs a reference flip of all of its children. We then only have perform a reference flip of the node itself, which also takes a constant amount of additional running time. Next, recall that the reference network is a forest of rooted trees. Resolving references works by walking towards the root of the tree that contains the reference and then returning the vertex with which this root is annotated. Over time, previously disjoint trees of that forest are merged together. Resolving references in overall linear time can be achieved by an application of the offline union-find algorithm due to Gabow and Tarjan [GT85]. This augments G into a single-source supergraph of G. We can then use the linear-time level-planar embedding algorithm for single-source level graphs by Di Battista and Nardelli [DN88]. Alternatively, we can vertically flip the single-source supergraph of G and run our algorithm again to obtain an st-planar augmentation of G. Using the fact that for st-graphs planarity coincides with level planarity [Lei98, p. 117, Theorem 5.1] we can then use the linear-time planar embedding algorithm by Chiba et al. [CNAO85] to obtain a level-planar embedding of G. We conclude the following.

Theorem 8. *A level graph can be tested for (radial) level planarity in linear time. If the result is positive, it can be embedded in linear time as well.*

5.14 Conclusion

We have presented a linear-time algorithm that tests a level graph for (radial) level planarity and, if the result is positive, outputs a (radial-)level-planar embedding of it. Although our algorithm is not the first linear-time algorithm for (non-radial) level planarity testing, it is, in our opinion, the first one to come with a convincing proof of correctness. For radial level planarity testing, we have argued that our algorithm covers an important case that seems to have not been handled by the existing literature. This makes our algorithm the first complete linear-time algorithm for radial level planarity testing and embedding.

Part II

Constrained Embeddings

6 Partial and Constrained Level Planarity

Let $G = (V, E)$ be a k-level graph with level assignment ℓ. In the problem Constrained Level Planarity (CLP for short), we are given a partial ordering $\prec_i$ of $V_i := \ell^{-1}(i)$ for each $i \in [k]$, and we seek a level-planar drawing where the order of the vertices on ℓ_i is a linear extension of $\prec_i$. A special case of this is the problem Partial Level Planarity (PLP for short), where we are asked to extend a given level-planar drawing $\mathcal{H}$ of a subgraph $H \subseteq G$ to a complete drawing $\mathcal{G}$ of G without modifying the given drawing, i.e., the restriction of $\mathcal{G}$ to H must coincide with $\mathcal{H}$.

We give a simple polynomial-time algorithm with running time $O(n^5)$ for CLP of single-source graphs that is based on a simplified version of an existing level-planarity testing algorithm for single-source graphs. We introduce a modified type of PQ-tree data structure that is capable of efficiently handling the arising constraints to improve the running time to $O(n+ks)$, where s denotes the size of the constraints. We complement this result by showing that PLP is NP-complete even in very restricted cases. In particular, PLP remains NP-complete even when G has a constant number of levels, and when G is a subdivision of a triconnected planar graph with bounded degree.

This chapter extends work initiated as part of my master's thesis [Brü16] and is based on joint work with Ignaz Rutter [BR17]. Here, we present improved running times of $O(n + ks)$ for CLP and $O(kn)$ for PLP, compared to $O(n^3)$ and $O(n^2)$ in [Brü16], respectively. Moreover, we have restructured the correctness proof.

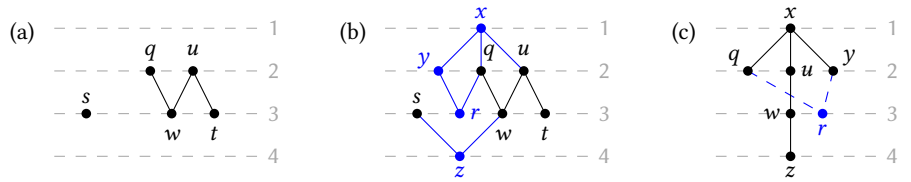


Figure 6.1: A partial level-planar drawing (a) and an extension of it (b). The partial level-planar drawing shown in (c) does not have an extension. For instance, the vertex r and its incident edges cannot be inserted into the partial drawing without creating edge crossings.

6.1 Introduction

In this chapter, we study a generalization of the level planarity testing problem where, in addition to the k-level graph G, we are given *constraints* on the order of the vertices in each level in the form of a partial order $\prec_i$ for each level $i \in [k]$. The task is to determine a level-planar drawing of G *compatible* with the constraints, i.e., such that the vertex order in each level i is a linear extension of $\prec_i$. We call this problem Constrained Level Planarity or CLP for short.

A strongly related problem is the problem of extending a given drawing $\mathcal{H}$ of a subgraph $H \subseteq G$, also called *partial drawing*, to a level-planar drawing $\mathcal{G}$ of G without changing the given drawing $\mathcal{H}$. That is, the restriction of $\mathcal{G}$ to H must coincide with $\mathcal{H}$. We call such a drawing $\mathcal{G}$ an *extension*. The problem of deciding whether a given partial drawing admits an extension is called Partial Level Planarity, or PLP for short. See Figure 6.1 that shows a partial drawing (a) and an extension of it (b), and also a partial drawing that does not have an extension (c). Of course, taking for each level i the order $\prec_i$ as the vertex order on level i in $\mathcal{H}$ yields a set of constraints for G. An extension is necessarily compatible with all these constraints. Later, we will see that for so-called proper level graphs, where all edges connect vertices of adjacent levels, these instances are actually equivalent. Since both problems can be reduced to the corresponding problem on proper level graphs it turns out that indeed PLP can be seen as a special case of CLP.

Related Work. Historically, level drawings were among the first drawing styles that were studied systematically with the introduction of the famous Sugiyama framework [STT81]. Level drawings are frequently used for visualizing hierarchical data and there is extensive research on every step of the framework; see [HN13] for a survey. Due to the importance of crossings on the visual clarity of drawings [PCJ97], the concept of level planarity has also received considerable attention. Di Battista

and Nardelli [DN88] gave the first efficient recognition algorithm for the subclass of single-source graphs. Jünger and Leipert [JLM98] solved the general case and gave a linear-time algorithm for testing whether a given level graph is level planar. Later they improved their algorithm to also output a corresponding level planar embedding, if it exists, in the same running time [JL02]. Their algorithm is, however, quite complicated. As a result, Randerath et al. [Ran+01] gave a much simpler recognition algorithm with running time $O(n^2)$, and Harrigan and Healy [HH07] formulated a quadratic-time recognition and embedding algorithm. There is also a quadratic-time algorithm based on a Hanani-Tutte characterization [FPSŠ13]. Moreover, radial variants, where the levels are represented by concentric circles rather than horizontal lines, have been considered [BBF05]. There also exists a Hanani-Tutte characterization for radial level planarity [FPS17]. Recently, a connection between the algorithm due to Randerath et al. [Ran+01] and the Hanani-Tutte characterization due to Fulek et al. [FPS17] was used to obtain a simple, efficient recognition algorithm for radial level-planar graphs [BRS18]. Further variants on cylinders [Bra14] and on the torus [Ang+15a] have also been considered.

Constrained versions of graph representations and planarity variants are another type of widely studied problem as, for visualization purposes, it is often desirable to find visualizations that satisfy additional properties. Some constrained versions of level planarity have been considered. Harrigan and Healy [HH07] and Angelini et al. [Ang+15b] both study level planarity where vertices of the same level must be ordered consistently with different kinds of *constraint trees*. Klemz and Rote show that deciding level planarity is NP-complete even in the severely constrained case where the orders of all vertices on all levels are fixed [KR19]. In real-world settings, sources of constraints include restrictions imposed by a user, e.g., by specifying the left-to-right order of some vertices or even fixing a part of the drawing, or as a consistency requirement stemming from the dynamic nature of a network, where one considers the restrictions imposed by stable parts of a network as constraints on the representation of the next snapshot. The latter leads to a *partial drawing extension* problem. More formally, a *partially drawn graph* is a triplet $(G, H, \mathcal{H})$, where G is a graph, $H \subseteq G$ is a subgraph and $\mathcal{H}$ is a drawing of H. The partial drawing extension problem with input $(G, H, \mathcal{H})$ asks to complete the drawing $\mathcal{H}$ to a drawing $\mathcal{G}$ of G without modifying the given subdrawing of $\mathcal{H}$, i.e., the restriction of $\mathcal{G}$ to H is identical to $\mathcal{H}$. The partial drawing extension problem, and the related simultaneous drawing problem have received considerable attention in the last years.

Angelini et al. [Ang+15c] give a linear-time algorithm for partial drawing extension of *topological* planar drawings, where edges are represented by arbitrary, non-crossing Jordan arcs between their endpoints. In this case, the positive instances have also been characterized via forbidden subgraphs [JKR13] and there exists a Hanani-Tutte characterization [Sch13], which also leads to a polynomial-time algorithm. In contrast,

for planar straight-line drawings, the partial drawing extension problem is NP-complete [Pat06], though it becomes polynomial-time solvable if H is biconnected and $\mathcal{H}$ is a convex drawing [MNR16]. Recently, partial upward planar drawings have been considered [DDF20, BHR19]. Moreover, the problem has also been studied under the name partial representation extension for graph representations that are not node-link representations, in particular for different types of intersection representations, namely for interval representations [Kla+17b, BR16], for proper and unit interval representations [Kla+17a], for chordal graphs (which are intersection graphs of subtrees of a tree) [KKOS15], for permutation and function graphs [KKKW12] and, recently, for circle graphs [CFK19] and trapezoid graphs [KW17].

We point out that the partial drawing extension problem is strongly related to the *simultaneous drawing problem*, whose input consists of two graphs G_1 and G_2 sharing a subgraph $G = G_1 \cap G_2$. The problem is to decide whether there exist drawings of G_1 and G_2 that coincide on G. This is equivalent to finding a drawing of G that extends to drawings of G_1 and G_2. It is known that the problem is NP-complete for planar straight-line drawings [Est+07], though some special cases have recently been shown to be polynomial-time solvable [GHKR14]. For topological planar drawings, this problem is called Simultaneous Embedding with Fixed Edges and despite significant progress in the last years [Ang+12, BR15, BKR18, BR16, FT20], the complexity status is still open; see [BKR13] for a survey. The simultaneous representation problem for intersection representations was introduced by Jampani and Lubiw, who studied permutation and chordal graphs [JL12] and interval graphs [JL10]. Bläsius and Rutter [BR16] improved the running time for simultaneous interval representations to linear. Recently, level planarity has been considered in this setting, too; simultaneous level planarity is NP-complete for two graphs on three levels, as well as for three graphs on two levels, an efficient algorithm exists only for two graphs on two levels [Ang+15a].

Contribution and Outline. We study the level planarity problem subject to constraints and in the context of partial drawings. We present preliminaries in Section 6.2. In particular, we show that, indeed, PLP reduces to CLP. In Section 6.3 we present a simplified version of the algorithm of Di Battista and Nardelli [DN88] for the single-source case. We then introduce the order graph, a data structure that allows us to efficiently intersect an implicit representation of the set of all level-planar drawings of a graph with an implicit representation of a set of (not necessarily planar) level drawings that satisfy the constraints. Since the latter set is guaranteed to contain all level-planar drawings that satisfy the constraints, this allows us to solve CLP for single-source graphs in time $O(n^5 + s)$, where s is the *size* of the constraints, i.e., the number of distinct vertex pairs that are related by some $\prec_i$. We then refine the algo-

rithm and develop a modified version of a PQ-tree, which we call *constrained PQ-tree*. It handles both the planarity and the constraints in the same data structure. With this modification the algorithm runs in time $O(n + ks)$ for proper k-level graphs. In Section 6.4 we complement these results by showing that PLP is NP-complete even in quite restricted cases, in particular, even when it has a constant number of levels, and when it is a subdivision of a triconnected planar graph (i.e., its combinatorial embedding is essentially fixed), the graph has fixed maximum degree and the graph has only a constant number of sources per level. This shows that our algorithmic results are likely tight.

Moreover, while a Hanani-Tutte-style characterization exists for level planarity, our hardness result rules out the existence of such a characterization for partial level planarity. This contrasts the situation for usual (topological) planarity, where Hanani-Tutte characterizations exist for both variants [Sch13].

6.2 Preliminaries

In this chapter, we restrict our treatment to proper level graphs. This is without loss of generality, though the resulting proper graph may have size in $\Theta(n^2)$, where $n = |V|$ is the number of vertices of the original graph.

Consider a level-planar embedding of a proper k-level graph $G = (V, E)$, which consists of linear orderings $\prec_i$ of the vertices in V_i along the horizontal line ℓ_i with y-coordinate i. By defining $u \prec v$ if $\ell(u) < \ell(v)$ or if $\ell(u) = \ell(v)$ and $u \prec_i v$, we can describe a level drawing by a single linear ordering $\prec$ of V, which lists the vertices in $V_1, \dots, V_k$ in their left-to-right orders. Di Battista and Nardelli give the following characterization of level-planar drawings.

Lemma 36 ([DN88, Lemma 1]). *Let $G = (V, E)$ be a proper k-level graph and let $\prec$ be a drawing of G. Then $\prec$ is planar if and only if for any distinct vertices $u, w \in V_j$ and $v, x \in V_{j+1}$, $j \in [k]$ with $(u, v), (w, x) \in E$ it is $u \prec w \Leftrightarrow v \prec x$.*

Two vertex pairs ss' with $s, s' \in V_i$ and tt' with $t, t' \in V_j$ are *equivalent* ($ss' \equiv tt'$ for short) if in every level-planar drawing $\prec$ of G it is either $s \prec s'$ and $t \prec t'$ or $s' \prec s$ and $t' \prec t$. Lemma 36 allows us to determine a first set of equivalent vertex pairs.

Of course, if $ss' \equiv tt'$ and $tt' \equiv uu'$, transitivity implies $ss' \equiv uu'$. More generally, we can obtain additional equivalent pairs by considering suitable paths in G as follows. Let $\text{lace}(a, b)$ denote the set of all paths whose first vertex is in level a, whose last vertex is in level b, and whose interior vertices v satisfy $\ell(v) \in [a, b]$.

Lemma 37 ([DN88, Lemma 2]). *Let G be a proper k-level graph and let $(s, \dots, t)$ and $(s', \dots, t')$ in be two vertex-disjoint paths in $\text{lace}(a, b)$ with $1 \le a < b \le k$. Then $ss' \equiv tt'$.*

Drawing constraints for a proper level graph $G = (V, E)$ are expressed as a partial ordering $\prec'$ of V. A drawing $\prec$ of G *satisfies* the constraints $\prec'$ if and only if for every $u, v \in V$ with $u \prec' v$ it is also $u \prec v$. The *size* s of $\prec'$ is the number of distinct vertex pairs that are related by $\prec'$. In general, s is quadratic in n.

Let $(G, H, \mathcal{H})$ be an instance of PLP. The proper subdivision of $\mathcal{H}$ induces a total left-to-right order of the vertices on each level in the proper subdivision of H, i.e., an instance of CLP. Note that if a partial drawing is stored transitively reduced, its size is linear in n. Moreover, Lemma 36 gives that in a straight-line level-planar drawing of a proper graph, the vertices can be moved horizontally without creating crossings as long as their order within the levels does not change. Thus, if a drawing satisfying the order constraints of an instance of PLP exists, the vertices of the drawing can always be moved so that the partial drawing is preserved. It follows that PLP is indeed a special case of CLP.

Lemma 38. *For proper level graphs, PLP reduces to CLP in linear time.*

Note that this uses the fact that $\mathcal{H}$ is a drawing. A combinatorial embedding of H would not be sufficient, because testing level planarity of non-proper level graphs where the orders of all vertices on all levels are fixed is NP-complete [KR19]. The complete edge-vertex order can therefore be assumed to be encoded in $\mathcal{H}$, which implies that PLP reduces to CLP in linear time even for non-proper graphs.

Conversely, drawing constraints are strictly more powerful than partial drawings, e.g., it is possible to express the constraints $u \prec' v, u \prec' w$ without deciding whether $v \prec' w$ or $w \prec' v$. This is not possible with partial drawings, because the totality property requires that v and w be related by the partial drawing as well.

6.3 Single-Source Graphs

Di Battista and Nardelli [DN88] presented a linear-time algorithm to test single-source proper level graphs for level planarity. In this section, we present a simplified variant of their algorithm with the same running time. We then extend this algorithm to a CLP algorithm with running time $O(n^5 + s)$, where s is the size of the constraints $\prec'$, by introducing the new concept of *order graphs*. Finally, we introduce *constrained PQ-trees*, which are modified PQ-trees that are capable of efficiently maintaining additional constraints. With the use of constrained PQ-trees, we improve the running time of our algorithm to $O(n + ks)$, where k is the number of levels.

6.3.1 A Simple Level Planarity Testing Algorithm for Single-Source Graphs

For a proper k-level planar graph, we denote by G_j the subgraph induced by the vertices on levels $1, \ldots, j$. Moreover, by $E_j = (V_j \times V_{j+1}) \cap E$, we denote the edges of G between levels j and $j+1$. We start out by making the following observation about level-planar drawings.

Lemma 39. *Let G be a proper k-level graph together with a planar drawing $\prec$. Let $j \in [k-1]$ and for some $\lambda \in (0,1)$ let the order of edges in E_j in which they intersect with the horizontal line $y = j + \lambda$ be $(u_1, v_1) \prec (u_2, v_2) \prec \ldots \prec (u_t, v_t)$. Then the edge endpoints are ordered consecutively, i.e., if $u_a = u_b$ with $1 \leq a < b \leq t$, it follows that $u_a = u_{a+1} = \ldots = u_b$, and if $v_a = v_b$ with $1 \leq a < b \leq t$, it follows that $v_a = v_{a+1} = \ldots = v_b$.*

Proof. Assume that $\prec$ is planar and that there exists $u_c \neq u_a = u_b$ with $a < c < b$. Then it is $(u_a, v_a) \prec (u_c, v_c) \prec (u_b = u_a, v_b)$. From $u_a \neq u_c \neq u_b$ it follows that either $u_a \prec u_c$ or $u_c \prec u_b = u_a$ holds true. If it is $u_a \prec u_c$, then (u_c, v_c) intersects with $(u_b = u_a, v_b)$. Conversely, if it is $u_c \prec u_a$, then (u_c, v_c) intersects with (u_a, v_a). With Lemma 36, this contradicts the planarity of $\prec$, so no such u_c can exist. The same idea can be used to show that there exists no $v_c \neq v_a$ with $a < c < b$. □

The idea of Di Battista and Nardelli is to perform a sweep of the single-source proper k-level graph, successively visiting each level $j \in [k]$ in increasing order. When going from level j to level $j+1$, they compute how level-planar drawings of G_j can be extended to level-planar drawings of G_{j+1}. To efficiently represent all possible planar drawings simultaneously, they use PQ-trees.

Our algorithm follows this approach, computing the same PQ-trees, albeit in a simpler way. Consider how a planar drawing $\prec_j$ of G_j can be extended to a drawing $\prec_{j+1}$ of G_{j+1}. Start by drawing non-intersecting edge stubs protruding from u for any edge $(u, v) \in E_j$. Lemma 39 states that in any planar drawing, these edge stubs have to be ordered so that identical endpoints are consecutive. The stubs can then be extended to complete edges meeting at their shared endpoints without causing any edge crossings.

The algorithm is as follows. Instead of considering individual drawings, for each level $j \in [k]$ a PQ-tree T_j is computed that represents the orders of level-j vertices across all level-planar drawings of G_j; see Figure 6.2 for an example. The tree T_1 consists of a single leaf, the source of G. Given a tree T_j, the tree T_{j+1} for the subsequent level is generated as follows. If T_j is the null-tree, so is T_{j+1}. If T_j is not the null-tree, consider each leaf u of T_j. If u has no child edges in E_j, mark u as obsolete. Otherwise, add each child edge as a child of u in T_j. Once all leaves have been

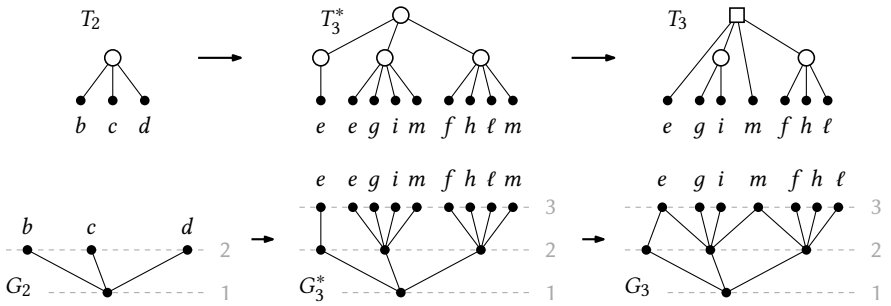


Figure 6.2: Illustration of Di Battista and Nardelli's algorithm applied to the graph $G = G_3$. Starting from the PQ-tree T_2, the middle PQ-tree is generated by adding all level-3 neighbors as children to b, c, d. Then, multiple occurrences are made consecutive and replaced by a single vertex, leading to T_3, as shown in the right.

considered, post-order traverse through T_j and mark any node as obsolete if all its children are obsolete. Then remove all obsolete nodes from T_j. For each vertex $v \in V_j$ the edge leafs with endpoint v are made consecutive in T_j using the update operation. Recalling the PQ-tree update operation, the edge leafs are now consecutive black subtrees attached to a Q-node. Finally, these black subtrees are replaced by a single leaf v. The following lemma establishes the correctness of this algorithm.

Lemma 40. *Let G be a single-source proper k-level graph and let T_j be the PQ-tree for some level $j \in [k]$. It is $(v_1, v_2, \ldots, v_t) \in \text{consistent}(T_j)$ if and only if there exists a level-planar drawing $\prec_j$ of G_j that satisfies $v_1 \prec_j v_2 \prec_j \ldots \prec_j v_t$.*

Proof. We prove the claim by induction over the number of levels. Since G has a single source, it is $|V_1| = 1$ and the claim is trivially true for $j = 1$. Assume that the claim is true for some $j \in [k-1]$. We show that it also holds for $j+1$.

By the inductive hypothesis, consistent(T_j) contains exactly the orders of level-j vertices across all level-planar drawings of G_j. We say that T_j *represents* G_j for short. Let $G^\star_{j+1}$ denote the graph obtained from G_{j+1} by splitting the vertices on level j such that they all have degree 1 and label each leaf with the original vertex on level $j+1$. By Lemma 42 the level-planar drawings of G_{j+1} correspond bijectively to the level-planar drawings of $G^\star_{j+1}$ where the vertices of level $j+1$ with the same label are consecutive.

Let $T^\star_{j+1}$ be the PQ-tree obtained from T_j by turning each level-j leaf u into a P-node with a leaf for every edge in E_j incident to u, or removing u if no edge in E_j is incident to u. Note that the edges in E_j correspond bijectively with the level-$(j+1)$ vertices of $G^\star_{j+1}$. By Lemma 39, $T^\star_{j+1}$ represents the orderings of the level-$(j+1)$ vertices

across all level-planar drawings of $G^{\star}_{j+1}$, and after applying the update operations, the resulting PQ-tree represents the level-$(j+1)$ vertex orders of all level-planar drawings of $G^{\star}_{j+1}$ where level-$(j+1)$-vertices with the same label are consecutive. Since these correspond bijectively to the orders of the level-planar drawings of G_{j+1}, the PQ-tree T_{j+1} where all leaves with the same label v are replaced by a single leaf v for each $v \in V_{j+1}$ indeed represents the orders of vertices on level $j+1$ across all planar drawings of G_{j+1}. □

From Lemma 40 it follows that if T_j is the null-tree for any $j \in [k]$, then G is not level planar. Otherwise, if T_k is not the null-tree, G is level planar. For a family of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$ all update(T, S_i) operations are feasible in a total running time of $O(|T| + |\mathcal{S}| + \sum_{i=1}^{t} |S_i|)$ [BL76]. Let n_j denote the number of level-j vertices. It is $|T_j| \in O(n_j + n_{j+1})$, $|\mathcal{S}| \in O(n_{j+1})$ and since $\mathcal{S}$ is a family of disjoint sets $\sum_{i=1}^{t} |S_i| \in O(n_{j+1})$. This gives linear running time for the entire algorithm. Di Battista and Nardelli also give an important link of inner nodes of the PQ-tree T_j to parts of the graph G_j.

Lemma 41 ([DN88]). *Let G be a single-source proper level graph and let T_j be the PQ-tree for some level $j \in [k]$. Every P-node Y in T_j corresponds to a cutvertex of G_j whose removal from G_j separates vertices that appear in subtrees rooted at distinct children of Y. Every Q-node Y in T_j corresponds to a maximal biconnected component H in G. Further, there exists a cycle C in H so that each child of Y corresponds to a cutvertex of G_j that lies on C whose removal separates the vertices that appear in the yield of the subtree rooted at that child from H.*

6.3.2 A Polynomial-Time CLP Algorithm for Single-Source Graphs

The algorithm from the previous section uses PQ-trees T_j to restrict all drawings of a level graph to just those that are level planar, represented by consistent(T_j). This section introduces order graphs P_j, which are used to restrict all drawings of a level graph to just those that are compatible with the constraints $\prec'$, represented by consistent(P_j). Together, these two data structures restrict all drawings of a level graph to those that are planar *and* satisfy the given constraints $\prec'$, represented by consistent$(T_j, P_j) :=$ consistent$(T_j) \cap$ consistent(P_j). We show that there exists a level-planar drawing of G compatible with $\prec'$ if and only if consistent(T_j, P_j) is non-empty for all $j \in [k]$. The remainder of this section describes (i) the order graph data structure, (ii) the generation of the order graphs P_j, (iii) the interoperation of PQ-trees and order graphs, (iv) how to determine whether consistent(T_j, P_j) is non-empty, and (v) the resulting CLP algorithm.

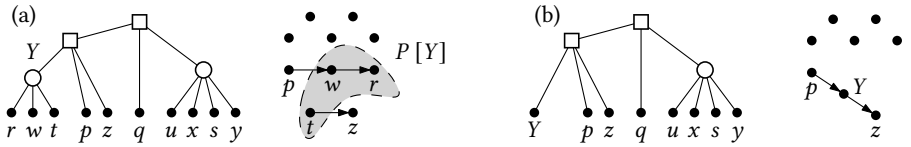


Figure 6.3: Illustration of Lemma 42. Part (a) shows T and P, and (b) shows $T \circ Y$ and $P \circ Y$.

Let $G = (V, E)$ be a proper level graph. The order graph $P_k = (V, F)$ for $G = G_k$ is initially the empty digraph on the same vertex set as G. Edges are then added step by step. First, for any vertex pair $u, v \in V_j$ on some level $j \in [k]$ with $u \prec' v$, the edge (u, v) is added to P_k. Next, according to Lemma 36, if $(u, w) \in F$ and $uw \equiv vx$, then (v, x) is added to P_k. To ensure transitivity, if $(u, v), (v, w) \in F$, then (u, w) is added to P_k as well. These last two steps are repeated until no more edges can be added to P_k. The level-j order graph P_j is the order graph for G_j. We define $\text{yield}(P_j) = V_j$ and $\text{consistent}(P_j)$ as the set of topological orderings of the subgraph of P_j induced by V_j.

Adding edges from the constraints takes time linear in the size of $\prec'$. Adding edges whose existence is implied by Lemma 36 is possible by considering all quadratically many edge pairs in E. The other edges can be added by running a transitive closure algorithm, e.g., the Floyd-Warshall algorithm [War62], which requires cubic time. These last two steps are repeated at most $O(n^2)$ times, because P_k can contain at most n^2 edges. This gives an overall running time of $O(n^5)$. Because we only add necessary constraints we do not have to recompute P_j for every level j, but can instead simply compute only P_k.

The next step is to determine whether a PQ-tree T and an order graph P are consistent, i.e., whether $\text{consistent}(T, P) \neq \emptyset$. To this end, we use a recursive procedure similar to the one by Klavík et al. [Kla+17b, Step 3]. First, it picks an inner node Y of T that has only leaves as children. If $P[Y] \coloneqq P[\text{yield}(\text{pertinent}(Y))]$ has no suitable topological ordering, T and P are inconsistent. Otherwise, the PQ-tree $T \circ Y$ is generated from T by removing all children of Y, making it a leaf, and the order graph $P \circ Y$ is generated from P by identifying the vertices in $\text{yield}(\text{pertinent}(Y))$ into a single vertex Y; see Figure 6.3. Then T and P are consistent if and only if $T \circ Y$ and $P \circ Y$ are consistent. This is formalized as follows.

Lemma 42. *Let T be a PQ-tree and P an order graph with identical yields. Let Y be an inner node of T so that all its children $y_1, y_2, \ldots, y_t$ are leaves. If Y is a P-node and $P[Y]$ has no topological ordering, or if Y is a Q-node and neither $(y_1, y_2, \ldots, y_t)$ nor $(y_t, y_{t-1}, \ldots, y_1)$ are topological orderings of $P[Y]$, then T and P are inconsistent.*

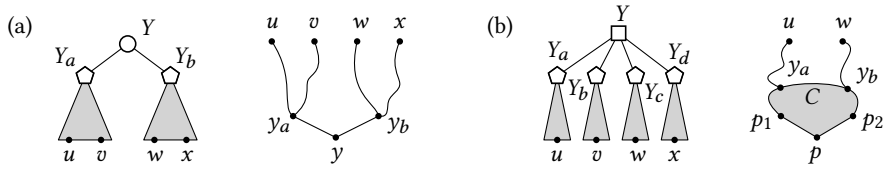


Figure 6.4: Proof of Lemma 43 for P-nodes and Q-nodes in (a) and (b), respectively.

Otherwise, T and P are consistent if and only if $T \circ Y$ and $P \circ Y$ are consistent.

Proof. If $P\,[Y]$ has no topological ordering, it is consistent$(P) = \emptyset$, so T and P are inconsistent. If Y is a Q-node and neither $(y_1, y_2, \ldots, y_t)$ nor $(y_t, y_{t-1}, \ldots, y_1)$ is a topological ordering of $P\,[Y]$, either ordering of the children of Y is inconsistent with P. Now assume that T and P are consistent, as witnessed by some permutation $\pi \in$ consistent(T, P). Since $\pi \in$ consistent(T), the children of Y appear consecutively in π. Replacing this sequence with Y yields a permutation π' that is in consistent$(T \circ Y, P \circ Y)$. Conversely, assume that $T \circ Y$ and $P \circ Y$ are consistent, as witnessed by some permutation π'. If Y is a P-node, $P\,[Y]$ has a topological ordering ρ, which is used to replace Y in π', yielding π. It is $\pi \in$ consistent(P). The leaves $y_1, y_2, \ldots, y_t$ can be attached to $T \circ Y$ in the order ρ. This implies that $\pi \in$ consistent(T, P). The same idea applies to Q-nodes with $(y_1, y_2, \ldots, y_t)$ and $(y_t, y_{t-1}, \ldots, y_1)$ as the only topological ordering candidates. □

The running time of the consistency procedure is linear in the size of the order graph P, or $O(n^2)$ across all order graphs. Note that instead of destroying T by calculating $T \circ Y$, one can simply treat Y as a leaf. The consistency procedure then applies equivalence transformations on T to make its frontier a topological ordering of P, i.e., reordering T according to P. To prove the correctness of the algorithm, the following relationship between inner nodes of the PQ-tree and order graphs is used.

Lemma 43. *Let G be a single-source planar proper level graph and let T be the corresponding PQ-tree for level j. Further, let Y be a node in T with ordered children $Y_1, Y_2, \ldots, Y_t$ and let*

$$u \in \text{yield}(\text{pertinent}(Y_a)), \qquad w \in \text{yield}(\text{pertinent}(Y_b)),$$
$$v \in \text{yield}(\text{pertinent}(Y_c)), \qquad x \in \text{yield}(\text{pertinent}(Y_d)),$$

where $1 \le a < b \le t$ and $1 \le c < d \le t$. If Y is a Q-node, or if Y is a P-node with $c = a$ and $d = b$, then $uw \equiv vx$.

Proof. First, assume that Y is a P-node, as shown in Figure 6.4 (a). Lemma 41 gives that Y corresponds to a cutvertex y in G. Since u and w belong to the yields of

pertinent subtrees of distinct children $Y_a \neq Y_b$ of Y, there are two paths $(u, \dots, y_a, y)$ and $(w, \dots, y_b, y)$ in G that have y as the only common vertex. For some subset of vertices, define a *peak* to be any vertex whose level is maximal among the vertices of that subset. As G is a single-source graph, y_a and y_b belong to the same level and are peaks of the respective paths. Lemma 37 gives $uw \equiv y_a y_b$. The same idea yields $vx \equiv y_a y_b$, which gives $uw \equiv vx$. Now assume that Y is a Q-node, as shown in Figure 6.4 (b). Lemma 41 gives that Y corresponds to a biconnected component H in G. Since G is planar, H is bounded by a cycle C, and since G is single-source, C has exactly one peak p. This peak has two children p_1, p_2 on C, regardless of the choice of a, b, c, d. As u and w belong to the yields of pertinent subtrees of distinct children of Y, there are disjoint paths $(u, \dots, y_a, p_1)$ and $(w, \dots, y_b, p_2)$ in G, where y_a and y_b are the first vertices on the paths that lie on C. Lemma 37 gives $uw \equiv p_1 p_2$. Applying the same idea yields $vx \equiv p_1 p_2$, and therefore $uw \equiv vx$. □

Lemma 44. *Let G be a single-source proper k-level graph together with constraints $\prec'$, let T_j be the PQ-tree for some level j and let P_j be the order graph for level j in step j. Then it is $(v_1, v_2, \dots, v_t) \in \text{consistent}(T_j, P_j)$ if and only if there exists a planar drawing $\prec_j$ of G_j that is compatible with $\prec'_j$ and satisfies $v_1 \prec_j v_2 \prec_j \dots \prec_j v_t$.*

Proof. Consider any level-planar drawing $\prec_j$ of G_j that extends $\prec'_j$ such that it is $v_1 \prec_j v_2 \prec_j \dots \prec_j v_t$. Since $\prec_j$ is level planar, $\pi := (v_1, v_2, \dots, v_t) \in \text{consistent}(T_j)$ by Lemma 40. Because $\prec_j$ extends $\prec'_j$, and only necessary edges have been added to the order graphs, it is $\pi \in \text{consistent}(P_j)$. Together, this gives $\pi \in \text{consistent}(T_j, P_j)$.

We prove the reverse direction by induction on the number of levels. The claim is trivially true for the base case $j = 1$ because V_j consists of the single source of G. Now let the claim hold true for level j and show that it is true for level $j + 1$. Let $(v_1, v_2, \dots, v_t) \in \text{consistent}(T_{j+1}, P_{j+1})$. Lemma 40 gives the existence of a planar drawing $\prec_{j+1}$ of G_{j+1} with $v_1 \prec_{j+1} v_2 \prec_{j+1} \dots \prec_{j+1} v_t$. Let $(u_1, u_2, \dots, u_{t'})$ be the order of the level-j vertices induced by $\prec_{j+1}$. Because $\prec_{j+1}$ is a level-planar drawing it is $(u_1, u_2, \dots, u_{t'}) \in \text{consistent}(T_j)$ by Lemma 40. A *conflict* is a pair u, w of reversed vertices, i.e., where $u \prec_{j+1} w$ and P_j contains the arc (w, u). We may assume without loss of generality that the choice of $\prec_{j+1}$ minimizes the number of conflicts in $(u_1, u_2, \dots, u_{t'})$. The idea is then to show that if there is a conflict, there exists a drawing $\prec'_{j+1}$ that does not have this conflict (i.e., the conflict is *resolved*) and introduces no new conflicts, a contradiction to the assumption that $\prec_{j+1}$ minimizes the number of conflicts. Therefore, $\prec_{j+1}$ has no conflicts, i.e., it is indeed in $\text{consistent}(T_j, P_j)$.

From $(u_1, u_2, \dots, u_{t'}) \notin \text{consistent}(P_j)$ it follows that there are distinct level-j vertices u, w with order $u \prec_{j+1} w$ for which the order graph P_j prescribes the reverse order. The relative order of u and w in $\text{frontier}(T_j)$ is determined at an internal

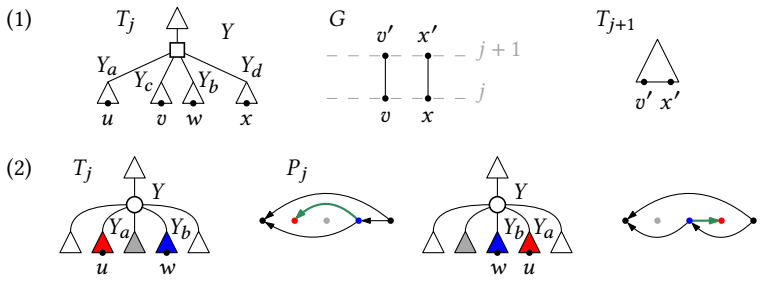


Figure 6.5: Case (1) and (2) of the proof of Lemma 44. In case (1), suppose that vertices u, w appear in that order in the frontier of T_j and that there are vertices v, x with $uw \equiv vx$ that have distinct children v', x' on level $j+1$. Then the frontier of T_{j+1} cannot lie in consistent(P_{j+1}). In case (2), find a conflict induced by a closest pair of reversed vertices, e.g., the green conflict between the children Y_a and Y_b (left). Then this conflict can be resolved without creating new conflicts (right).

node Y of T_j where u and w are leaves of subtrees rooted at distinct children of Y, say Y_a and Y_b; i.e., Y_a appears to the left of Y_b in the ordered sequence of children of Y. We distinguish two cases based on whether Y is a Q-node or a P-node.

1. Suppose that Y is a Q-node. See Figure 6.5 (1).
   a) Suppose that there are level-j vertices v, x that (i) are leaves of subtrees rooted at distinct children of Y, say Y_c and Y_d, where Y_c appears to the left of Y_d, and (ii) have distinct children v', x' on level $j+1$ in G_{j+1}. Lemma 43 gives $uw \equiv vx$. Lemma 36 gives $vx \equiv v'x'$, which means that the order graph P_{j+1} prescribes the order $x' \prec_{j+1} v'$. But planarity of $\prec_{j+1}$ gives $v' \prec_{j+1} x'$, a contradiction to $(v_1, v_2, \ldots, v_t) \in$ consistent(P_{j+1}).
   b) Otherwise, reverse the order of the children of Y. Let T'_j denote the new tree. By Lemma 40 there exists a planar drawing $\prec'$ where the level-j vertices appear in the order frontier(T'_j). Now extend this drawing to level $j+1$ with the fixed order $(v_1, v_2, \ldots, v_t)$. Suppose that this drawing is not level planar, i.e., there exists a crossing between two edges, say (v, v') and (x, x'). This implies $\ell(v) = \ell(x) = j$, $\ell(v') = \ell(x') = j+1$ and $v \neq x, v' \neq x'$. Because $\prec_{j+1}$ is planar we may assume $v \prec_{j+1} x$ and $v' \prec_{j+1} x'$. To cause the crossing, reversing the order of the children of Y must reverse the order of v and x. This means that v and x are leaves of subtrees rooted at distinct children of Y. Furthermore, v' and x' are distinct children of v and x in G_{j+1}, respectively. This means that the assumptions from case 1a are fulfilled, a contradiction. Therefore, reversing

the order of the children of Y does not cause a crossing. Hence, $\prec'$ has one conflict less than $\prec_{j+1}$, a contradiction.

2. Suppose that Y is a P-node.
   a) Suppose that there are level-j vertices v, x that (i) are leaves of the subtrees rooted at Y_a and Y_b, respectively, and (ii) have distinct children v', x' on level $j+1$ in G_{j+1}. Planarity gives $v' \prec_{j+1} x'$. Lemma 43 gives $uw \equiv vx$ and with $vx \equiv v'x'$ this means that if $u \prec_{j+1} w$ contradicts the order graph P_j, then $v' \prec_{j+1} x'$ contradicts the order graph P_{j+1}, a contradiction to $(v_1, v_2, \ldots, v_t) \in \text{consistent}(P_{j+1})$.
   b) Otherwise. We say that Y_a, Y_b *conflict* if it is $a < b$ but the order graph P_{j+1} contains the edge (w, u), where u and w are leaves of the subtrees rooted at Y_a and Y_b, respectively. We may assume without loss of generality that Y_a, Y_b are chosen so that u and w are a closest pair of reversed vertices. See Figure 6.5 (2). Then we can change the relative order of Y_a and Y_b without introducing new conflicts by either moving Y_a so that it becomes the right neighbor of Y_b, or by moving Y_b so that it becomes the left neighbor of Y_a. We now show how to do so without breaking planarity.
      i. There exist leaves u, w in the subtrees rooted at Y_a, Y_b, respectively that both have children in G_{j+1}. Because we're not in case 2a all leaves in the subtrees rooted at Y_a, Y_b have no child or a common child x. Because $\prec_{j+1}$ is planar, all leaves of subtrees rooted at Y_i with $i \in [a, b]$ also have either no child, or the child x. Observe that moving Y_a so that it becomes the right neighbor of Y_b only changes the relative order of vertices that are leaves of subtrees rooted at children Y_i with $i \in [a, b]$. Because all of these vertices have the same child x on level $j+1$, changing their order does not break planarity.
      ii. Otherwise, the leaves of one or both of the subtrees rooted at Y_a, Y_b have no children, say Y_a. Then Y_a can be moved so that it becomes the right neighbor of Y_b. This only changes the relative order of vertices that are leaves of subtrees rooted at Y_a and children Y_i with $i \in (a, b]$. Because Y_a has no leaf that has a child on level $j+1$, this cannot cause a crossing. Analogously, if no leaves of Y_b have children, it can be moved so that it becomes the left neighbor of Y_a.

   In both cases we have resolved a conflict without creating new conflicts while maintaining planarity.

This means that $(u_1, u_2, \ldots, u_{t'})$ is in consistent(T_j, P_j). When extending $\prec'_j$ to level j+1 with the fixed order $(v_1, v_2, \ldots, v_t)$, there are no crossings between levels j and $j+1$,

i.e., $\prec'_j$ is a level-planar drawing of G_{j+1} that extends $\prec'$. □

6.3.3 An Efficient CLP Algorithm for Single-Source Graphs

The running time of the CLP algorithm from the previous section can be improved to $O(n + ks)$, where s is the number of constraints in the partial drawing $\prec'$. To this end, order graphs and PQ-trees are merged into a single new data structure, the constrained PQ-tree.

A constrained PQ-tree is a PQ-tree T that stores additional edge constraints. An *edge constraint* is an ordered pair of edges (e, f) of the tree that have the same parent node Y with the meaning that e must occur before f in the counter-clockwise order of edges around Y starting with the parent edge. This information is stored in doubly linked constraint lists at both e and f; see Figure 6.6 for an example (constraints and parent edges are colored). Storing these lists requires one pointer to the head of the list for each edge and two list nodes of constant size for every constraint. Hence, T has a total size of $O(n + ks)$.

During the update procedure, a constraint (e, f) may need to be replaced by another equivalent constraint. See Figure 6.6. This is the case when their parent node is split so that the edges corresponding to e and f end up at different parent nodes after the update. In the following we distinguish cases based on whether e and f are terminal, black, or white.

Assume that e is terminal. Then (e, f) needs to be replaced by the equivalent constraint (e', f), where e' is determined as follows. Let Y be the parent node of e and f and let $e = (Y, Z)$. Then e' is the black child edge of Z that is a neighbor of e. This edge can be found in constant time. The same procedure can be used if f is terminal, e.g., the green constraint in Figure 6.6. At this point, no constraints involve terminal edges. Now, a constraint has to be moved if one edge is white, and the other edge is black. Assume that e is white and f is black. Then (e, f) is replaced by the constraint $((Y, x), (Y', x))$, where Y' is the split node of Y. Analogously, if e is black and f is white, (e, f) is replaced by $((Y', x), (Y, x))$. For an example, see the blue constraint in Figure 6.6. Note that all replacement operations have constant running time. All other constraints do not have to be replaced, e.g., the red constraint in Figure 6.6.

Let $T' = \text{update}(T, S)$ and $T'' = \text{update}(T', S')$ with $S \cap S' = \emptyset$. If some edge e of T is black or terminal during the $\text{update}(T, S)$ operation and it exists in T', it is white in T' during the $\text{update}(T', S')$ operation. Only constraints that involve a black or terminal edge are considered for replacement. Hence, any constraint is considered for replacement at most twice, leading to an additional running time in $O(s)$ across all update operations for one level. Walking through the tree to consider all relevant constraints for replacement requires an additional running time in $O(|S| + p)$ for

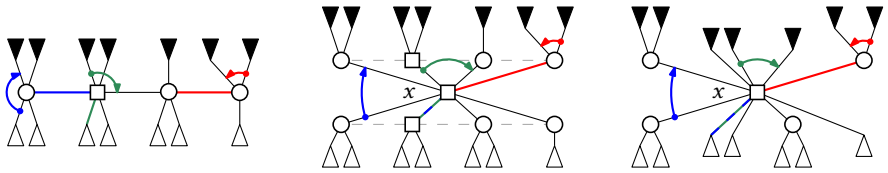


Figure 6.6: The update procedure for constrained PQ-trees. Constraints are drawn in colors, together with their respective parent edges. The blue constraint involves a white and a black edge, the green constraint involves a black edge and a terminal edge, and the red constraint involves two black edges.

a single update operation. The original PQ-tree update procedure is run as part of this new procedure. With Lemma 1, this gives a total running time of $O(n + ks)$ for all update operations of the constrained PQ-tree T over the k levels. We note that Q-nodes are not explicitly represented in the PQ-tree data structure and therefore references to their parents cannot necessarily be queried in constant time. However, such references *are* computed for Q-nodes on the terminal path as part of the update procedure, and only constraints around such nodes are candidates for replacement.

To check whether a constrained PQ-tree has a frontier that is consistent with the stored constraints, we use a similar procedure to the one from Section 6.3.2, i.e., we check for every P-node whether the constraints around it are acyclic and for every Q-node whether the constraints around it are non-conflicting. The running time of this procedure for level j is $O(n_j + s)$. To solve CLP, run the algorithm from Section 6.3.1 to compute the constrained PQ-tree T_j for each level j, then add all level-j constraints and run the consistency procedure. The partial drawing can be extended to a complete planar drawing of G if and only if T_j is consistent for each level. We conclude the following.

Theorem 9. *For single-source proper k-level graphs CLP can be solved in time $O(n + ks)$, where n is the number of vertices in the graph and s is the size of the constraints. In particular, PLP can be solved in $O(kn)$ time for single source proper k-level graphs.*

6.4 Complexity of the General Case

It is an interesting question whether the algorithm can be extended to graphs with multiple sources. Jünger et al. [JLM98] extend Di Battista and Nardelli's level planarity testing algorithm for multi-source graphs. To this end, they run an instance of Di Battista and Nardelli's algorithm for every source of the graph. As soon as the

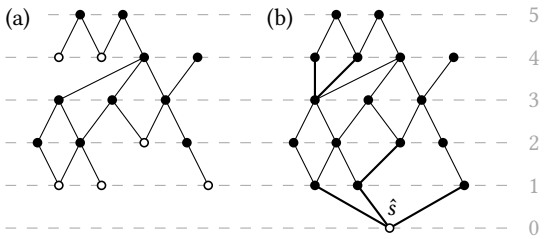


Figure 6.7: Removing sources from a multi-source level graph. Part (a) shows a level graph G with six sources. Sources are highlighted as white disks, all other vertices are drawn as black vertices. By applying the procedure outlined above Corollary 6, the supergraph G' shown in part (b) is generated. This new graph has only one source. Only a single new vertex has been added, namely $\hat{s}$, and six edges, one for every former source, have been added. These new edges are drawn in bold.

algorithm has progressed to a level j where multiple sources belong to the same connected component in G_j, the corresponding PQ-trees are merged. Unfortunately, the order graphs from the previous section cannot simply be reused in the multi-source case, because neither does Lemma 43 apply to multi-source graphs, nor is it obvious how constraints should be handled during the merge operation. One possible solution stems from the fact that any level graph with multiple sources can be transformed into a level graph with one source by adding one vertex and some edges, without changing its planarity. To see this, consider a planar drawing of a graph. Any level-j source can be removed by adding an edge to a suitable vertex on level $j-1$. For an example, see Figure 6.7. The difficulty of this procedure is to find out *which* edges to add. If the input graph G has a fixed number r of sources, one can simply try all possible combinations of edges. There are at most n choices per source, leading to a total of n^r choices. For each of them we employ the algorithm from Theorem 9.

Corollary 6. *For proper k-level graphs with r sources CLP can be solved in $O(n^r(n+ks))$ time.*

Indeed, it is unlikely that order graphs can be used for multi-source graphs as well, because in this section we show that PLP is NP-complete in the general case by reducing from the NP-hard problems 3-Partition and Planar Monotone 3-Sat [dK12]. The first reduction shows that PLP is NP-complete even for proper level graphs whose number of levels is bounded by a constant. The second reduction shows that PLP is NP-complete even for proper level graphs with fixed combinatorial embedding and high connectivity.

6.4.1 3-Partition Reduction

The 3-Partition problem asks whether a given multiset of integers can be partitioned into triples (referred to as *buckets*) so that the sums of elements of all triples are equal (i.e., all buckets have the same size). More formally, an instance of 3-Partition consists of a set A of $n = 3m$ elements, a bound $B \in \mathbb{Z}^+$, and a size $s(a) \in \mathbb{Z}^+$ for each $a \in A$ such that $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$. The question is then whether A can be partitioned into m disjoint sets $A_1, A_2, \dots A_m$ such that for $1 \leq i \leq m$ the equation $\sum_{a \in A_i} s(a) = B$ holds true. Note that each A_i must therefore contain exactly three elements from A. The 3-Partition problem is strongly NP-complete [GJ75].

The idea of the reduction is to construct a *bucket graph*, which largely consists of gadgets of two kinds. The first kind of gadget is the *socket*. There are a total of $\sum_{a \in A} s(a)$ sockets, which are organized into m *buckets*. The second kind of gadget is the *pin*. Every element $a \in A$ is represented by a *plug* that consists of $s(a)$ pins. As the naming suggests, sockets and plugs interact with each other. In any planar embedding of the bucket graph, every plug of size $s(a)$ is *coupled* to $s(a)$ sockets, and every socket is coupled to one pin of some plug. The embedding of the sockets is fixed by the partial drawing, whereas the order of the plugs remains variable. The bucket structure ensures the required property of the 3-Partition problem that every set A_i must contain exactly three elements. Furthermore, the sets $A_1, A_2, \dots, A_m$ can be easily determined by inspecting the i-th bucket of the bucket graph.

The socket, pin and plug gadgets are shown in Figure 6.8. A socket consists of a U-shaped path, together with two short y-monotone intertwined paths connected to it. The endpoints of these paths lie on the same level so that the endpoint of the path running downwards lies to the right of the endpoint of the path running upwards. Sockets are always fixed by the partial drawing. A pin is a simple serpentine-shaped path designed to fit into socket by "snaking" its way through the socket to reach the bottom. Pins are always left variable by the partial drawing. A pin is coupled with a socket when the end of plug p_e lies in between the vertices s_ℓ and s_r of the socket. To create adjacent sockets, take two sockets and identify the path $(\zeta, \dots, s_r)$ of the left socket and the path $(\alpha, \dots, s_\ell)$ of the right socket. To create a plug of size k, identify the roots of k pins. Figure 6.8 (c) shows a plug of size two coupled with two adjacent sockets. It is readily observed that a socket cannot be coupled with more than one pin or plug without causing edge crossings.

The bucket graph is constructed step-by-step, starting out with an empty level graph. Given an instance of the 3-Partition problem, the first step is to create a total of mB pins so that the roots of all pins are level-1 vertices. Then, for every element $a \in A$, take $s(a)$ pins to create one plug of size $s(a)$ and associate a with it. Given a plug p, the associated element is denoted by $a(p)$. At this point the strong

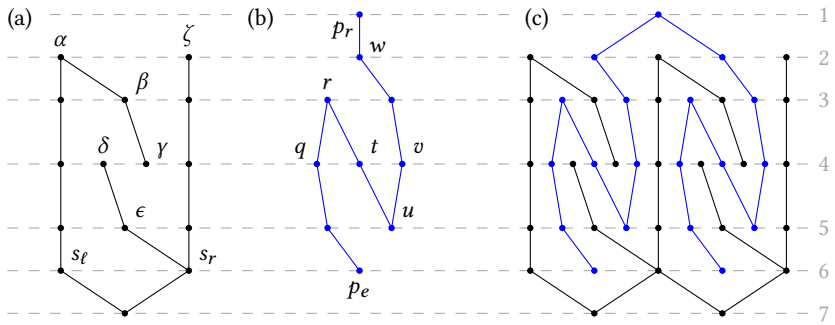


Figure 6.8: Part (a) shows a socket drawn in black and part (b) shows a pin drawn in blue. The pin has a root vertex p_r and an end vertex p_e. A pin is coupled with a socket when the end vertex of the pin, p_e, lies between the vertices s_ℓ and s_r of the socket. It is shown that a socket can be coupled with at most one pin. In part (c), a plug of size two is coupled with two adjacent sockets. The plug of size two is created by merging the roots of two pins. It is shown that a plug of size k is always coupled with k adjacent sockets.

NP-completeness is required because a plug of size $s(a)$ is created by merging the roots of $s(a)$ pins. Next, create mB adjacent sockets. Note that this equals the number of pins created. Then, constrain the end vertex p_e of every plug to lie within this socket structure, i.e., to lie to the right of the vertex s_ℓ of the leftmost socket and to the left of the vertex s_r of the rightmost socket. This means that in every level planar embedding extending the partial drawing, every plug has to be coupled with a socket. Conversely, every socket has to be coupled with a pin of a plug. Therefore, each plug is coupled with a socket, and each socket is coupled with a plug. Furthermore, a plug has to be coupled with consecutive sockets in order to maintain planarity. The last step of the construction is to organize the sockets into *buckets*. The bucket structure forces all sockets coupled with a given plug to be part of the same bucket, i.e., no plug may span across buckets. The buckets are constructed by adding $m + 1$ new level 1 vertices to the graph, called *bucket separators*. Then, the i-th bucket separator is connected to the vertex α of the i-th socket, and the $m + 1$-th bucket separator is connected to the vertex ζ of the m-th socket. Since the bucket separators lie on the same level as the roots of the plugs, any plug must clearly lie completely on either side of any bucket separator.

At this point, the equivalence of finding a solution for a 3-Partition instance $\mathcal{I}$ and extending the partial drawing $<'$ of the corresponding bucket graph G can be shown. Assume $\mathcal{A} = \{\{a_1, a_2, a_3\}, \ldots, \{a_{3m-2}, a_{3m-1}, a_{3m}\}\}$ is a solution of the given

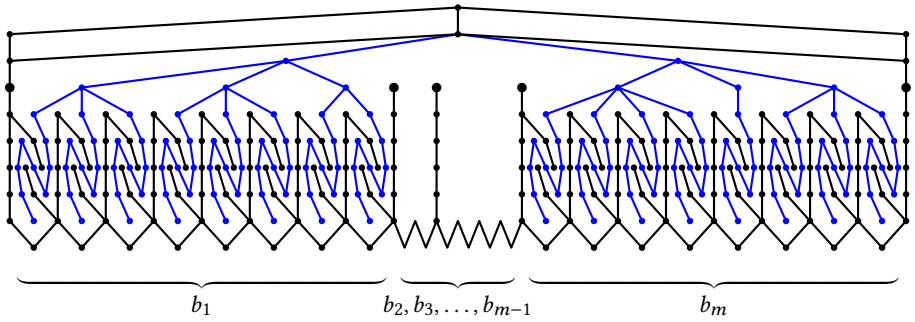


Figure 6.9: A bucket graph. There are a total of m buckets, each of which consists of eight adjacent sockets, drawn in black. The bucket separators are drawn as slightly larger black vertices. The plugs are drawn in blue. The sockets of the first bucket are coupled with three plugs. Two of these plugs have size two and the third plug has size two. This means that one of the triples in the corresponding 3-Partition problem is $(3, 3, 2)$. Similarly, the m-th bucket corresponds to the triple $(4, 1, 3)$.

3-Partition instance $\mathcal{I}$. Since $\mathcal{A}$ is a solution for $\mathcal{I}$, it is $s(a_1) + s(a_2) + s(a_3) = B$. Let p_i be the plug in the bucket graph G that is associated with a_i, i.e., $a(p_i) = a_i$. To complete the partial drawing $<'$ of G, take the first three plugs p_1, p_2, p_3 and couple them with the sockets of the first bucket. This same reasoning can be applied to all other triples in $\mathcal{A}$, resulting in a complete level-planar drawing $<$ of the bucket graph G. Now let G be the bucket graph corresponding to a 3-Partition instance $\mathcal{I}$ and assume that G has a level-planar drawing $<$. The drawing $<$ gives a strict total order on the roots of all plugs: $p_1 < p_2 < p_3 < \ldots < p_{3m-2} < p_{3m-1} < p_{3m}$. Consider the first bucket b_1 and recall the size restriction $B/4 < s(a) < B/2$ for each $a \in A$ of the instance $\mathcal{I}$. This means that p_1 and p_2 alone cannot fill the bucket b_1. Likewise, p_1, p_2, p_3 and p_4 cannot fit in the bucket b_1. So, the only plug candidates to fill the first bucket are the three first plugs p_1, p_2 and p_3. Each socket is coupled with a plug and because each bucket is made up of B sockets, it follows that $s(a(p_1))+s(a(p_2))+s(a(p_3)) \leq B$. Because no plug spans across multiple buckets, this means that $s(a(p_1)) + s(a(p_2)) + s(a(p_3)) \geq B$. Together with the previous statement, this gives $s(a(p_1)) + s(a(p_2)) + s(a(p_3)) = B$. This same reasoning can then be applied to the next three plugs p_4, p_5 and p_6, and so on, up to the last three plugs p_{3m-2}, p_{3m-1} and p_{3m}. Hence,

$$\{\{a(p_1), a(p_2), a(p_3)\}, \ldots, \{a(p_{3m-2}), a(p_{3m-1}), a(p_{3m})\}\}$$

is a solution for the 3-Partition instance $\mathcal{I}$. Using a similar construction we

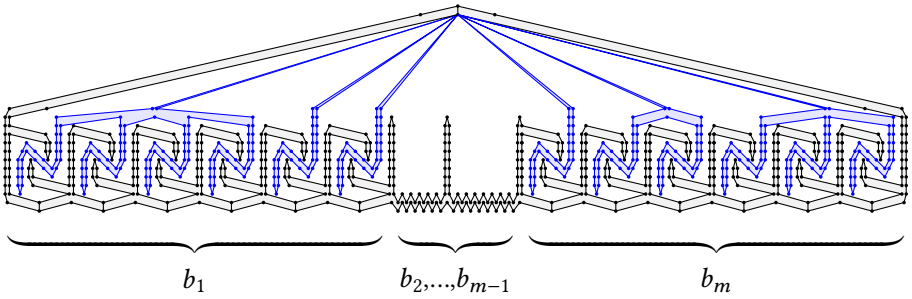


Figure 6.10: A connected bucket graph having a single cutvertex. The bucket structure is drawn in black. It consists of two interconnected cycles. The bucket separators are drawn as slightly larger black vertices. The plugs are drawn in blue. Note that every plug is a cycle. All of these cycles share one vertex, which is the only cutvertex in the graph.

can remove all cutvertices from the bucket graph, except one; see Figure 6.10. We conclude the following.

Theorem 10. *The PLP problem is NP-complete even for level graphs with a single cutvertex and a constant number of levels.*

We want to highlight the fact that because both the number of cutvertices and the number of levels is bounded by a constant, it is unlikely that PLP is fixed-parameter tractable with respect to either of these two parameters.

6.4.2 Planar Monotone 3-Sat Reduction

In this section, we present another reduction to show that PLP is NP-complete even when restricted to biconnected proper subdivisions of triconnected graphs with fixed combinatorial embedding and constant maximum vertex degree.

Recall that 3-Sat is the problem of deciding, given a set of variables V and a set of clauses $\mathcal{C}$ where every clause $C \in \mathcal{C}$ contains at most three literals, whether there is a Boolean truth assignment for the variables V so that every clause contains at least one literal that evaluates to true. A clause is called *positive* if it contains only positive literals, and *negative* if it only contains negative literals. A 3-Sat instance is monotone if it contains only positive and negative clauses. Further restrictions are expressed in terms of the *variable-clause graph*. The vertices of this graph are $V \cup \mathcal{C}$ and a vertex v and a clause C are connected by an edge if and only if v or $\neg v$ occurs in C. The variable-clause graph of Planar Monotone 3-Sat instances

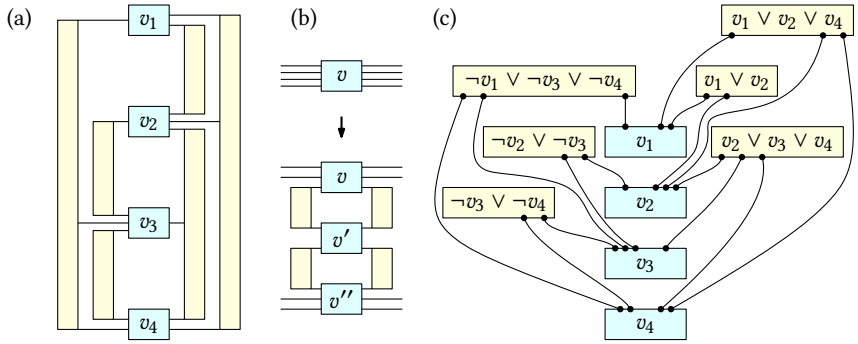


Figure 6.11: A variable-clause graph (a), the construction to ensure that each literal appears at most three times (note $v \Leftrightarrow v''$) (b) and the modified version with y-monotone Jordan arcs (c).

can be drawn in such a way that all variable vertices are drawn as squares on a vertical straight line, the *line of variables*, clauses are drawn as rectangles of variable height and all edges are drawn as horizontal straight lines, and positive clauses are drawn to the right of the line of variables and negative clauses are drawn to the left of the line of variables. Figure 6.11 (a) shows a variable-clause graph. De Berg and Khosravi proved that Planar Monotone 3-Sat is NP-complete [dK12, Theorem 1]. Note that every literal in a Planar Monotone 3-Sat instance can be assumed to occur in at most three clauses. See Figure 6.11 (b) to see how for a variable v whose literals appear more than three times new variables v', v'' and clauses $(v \vee v')$, $(\neg v \vee \neg v')$, $(v' \vee v'')$, $(\neg v' \vee \neg v'')$ can be created. Note $v \Leftrightarrow v''$, so some literals of v can be replaced by literals of v''. This construction can be repeated as many times until each literal appears in at most three clauses.

In order to transform a Planar Monotone 3-Sat instance $\mathcal{I}$ into a PLP instance $(G, \prec')$, parts of the variable clause graph are replaced by level graph gadgets together with partial drawings thereof. First, the drawing is altered so that all edges are drawn as y-monotone Jordan arcs that connect to the bottom of clause rectangles and to the top of variable rectangles; see Figure 6.11 (c). Then, every variable is replaced by a *variable gadget*, every clause is replaced by a *clause gadget* and every edge is replaced by a *pipe gadget*.

The pipe gadget consists of two channels that are fixed by the partial drawing, and one *conductor* that remains to be drawn. Figure 6.13 shows a pipe gadget, where the channels are drawn in black and the conductor is drawn in blue. Depending on the completion of the partial drawing, the conductor may flow through either the left

or the right channel. Note the construction in the middle, which serves to connect the inner part separating the two channels to the boundary. It splits the conductor into a lower and an upper part. The partial drawing is used to force the upper end of the lower part of the conductor between the claw-like structure at the lower end of the upper part of the conductor. This means that even though the conductor is split into two parts, both of them must lie in the same channel. Each pipe gadget is then merged with a clause gadget at the top and with variable gadget at the bottom. This way, information is relayed between clause gadgets and variable gadgets.

The variable gadget consists of six merged pipe ends, separated in the middle by a separator vertex, and three pins merged at their lower endpoint. This pin structure must lie either completely to the left of the separator, or completely to the right. Depending on this, the variable is configured as true or false. Figure 6.14 shows a variable gadget configured as false.

Lemma 45. *If a variable gadget is configured as false (true), the conductors of all pipes leading from it to the gadgets of positive (negative) clauses must run through the right channel. The channel choice of the conductors of all pipes leading to negative (positive) clauses is not restricted.*

The clause gadget consists of three merged pipe sources and a pin. Figure 6.12 shows a clause gadget where the first literal is satisfied. This pin must be drawn in one of the pipe sources, thereby forcing the conductor of that pipe to flow through the left channel.

Lemma 46. *For each clause gadget, the conductor of at least one of the three connected pipes must run through the left channel.*

Given an instance $\mathcal{I}$ of Planar Monotone 3-Sat, the corresponding PLP instance $(G, \prec')$ can clearly be computed in polynomial time. Assume that there exists a planar completion of $\prec'$. Consider a positive clause C. Lemma 46 states that the conductor of at least one of the connected pipes must run through the left channel. Suppose that the variable V at the other end of this pipe is configured as false. Then Lemma 45 states that the conductors of all pipes leading to positive clauses, including C, must run through the right channel. This is a contradiction, so V must be configured as true and C is satisfied. A similar argument can be made when C is a negative clause.

Conversely, suppose that all variable gadgets are configured according to a satisfying assignment. Consider a positive clause C. Since C is satisfied, at least one of its variables V is true. Let the conductor of the pipe P connecting C and V flow through the left channel. This is possible because according to Lemma 45, only the conductors of pipes leading to negative clauses must run through the right channel. A similar

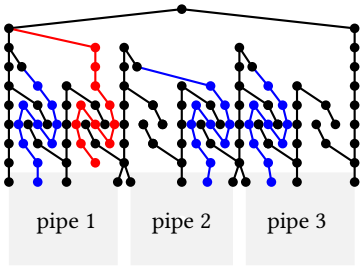


Figure 6.12: The clause gadget; pipes 1 and 3 transmit a literal value that satisfies the clause.

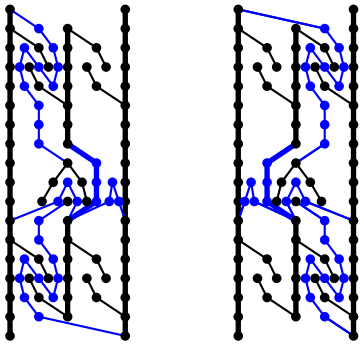


Figure 6.13: The two states of a pipe gadget. Note the construction in the middle, which serves to connect the inner part separating the two channels to the boundary.

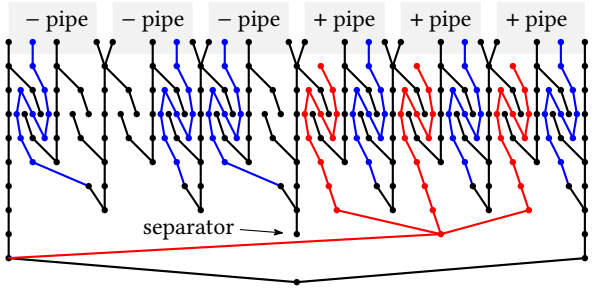


Figure 6.14: A variable gadget configured as false.

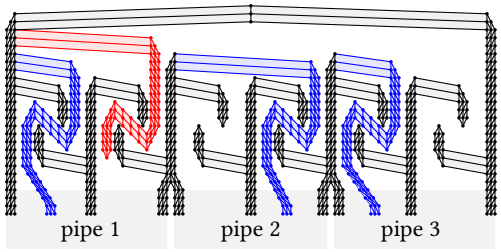


Figure 6.17: Augmented clause gadget where the graph forms a subdivision of a triconnected graph.

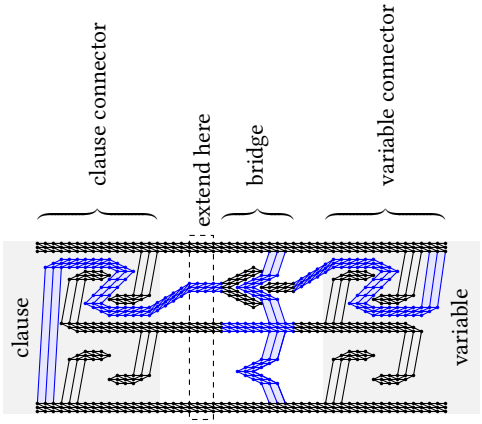


Figure 6.16: Augmented pipe gadget where the graph forms a subdivision of a triconnected graph.

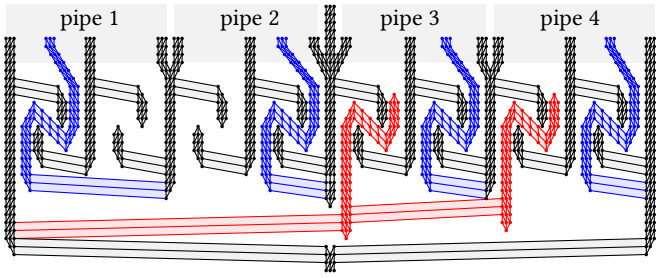


Figure 6.15: Augmented variable gadget where the graph forms a subdivision of a triconnected graph.

argument can be made when C is a negative clause. Any remaining conductors may flow through either channel.

Hence, a planar completion of $\prec'$ exists if and only if there is a satisfying variable assignment for $\mathcal{I}$. With the NP-completeness of Planar Monotone 3-Sat, this gives the following.

Theorem 11. *The problem PLP is NP-complete even for connected proper level graphs.*

The gadgets can be strengthened by replacing each vertex and each edge by a suitable internally triangulated graph (see Figures 6.15, 6.16 and 6.17) such that the resulting graph is a subdivision of a triconnected graph. Then the combinatorial embedding of the graph is fixed. Moreover, the maximum vertex degree is bounded, and there is only a constant number of sources per level. Note that here it is crucial that the graph we start with is connected.

Theorem 12. *The PLP problem remains NP-complete when restricted to biconnected proper subdivisions of triconnected graphs and constant maximum vertex degree.*

Because triconnected planar graphs have a fixed combinatorial embedding this result also implies NP-completeness of the PLP problem for level graphs with a fixed combinatorial embedding.

6.5 Conclusion

In this chapter we studied a constrained version of level planarity that allows to specify constraints on the linear order of vertices in the sought drawing. This problem contains as a special case the problem of extending a partial drawing in the level-planar setting, a problem that has recently received considerable interest for many other drawing styles and also other graph representations.

Our strong hardness results leave little room for polynomial-time algorithms for much larger classes of instances than single-source graphs. In view of Corollary 6, our most important open question is whether CLP is fixed-parameter tractable with respect to the number of sources in the graph.

Another interesting question is whether our techniques can be adapted for related drawing styles, in particular radial drawings, where levels are represented by concentric circles rather than horizontal lines, and upward drawings, where the levels of vertices are not fixed.

Acknowledgments: We thank Giordano Da Lozzo for the idea behind the plug/socket gadget.

7 An SPQR-Tree-Like Embedding Representation for Level Planarity

An SPQR-tree is a data structure that efficiently represents all planar embeddings of a biconnected planar graph. It is a key tool in a number of constrained planarity testing algorithms, which seek a planar embedding of a graph subject to some given set of constraints.

We develop an SPQR-tree-like data structure that represents all level-planar embeddings of a biconnected level graph with a single source, called the LP-tree, and give a simple algorithm to compute it in linear time. Moreover, we show that LP-trees can be used to adapt three constrained planarity algorithms to the level-planar case by using them as a drop-in replacement for SPQR-trees.

This chapter is based on joint work with Ignaz Rutter [BR20].

7.1 Introduction

Testing planarity of a graph and finding a planar embedding, if one exists, are classical algorithmic problems. For visualization purposes, it is often desirable to draw a graph subject to certain additional constraints, e.g., finding orthogonal drawings [Tam87] or symmetric drawings [HME06], or inserting an edge into an embedding so that few edge crossings are caused [GMW05]. Historically, these problems have been considered for embedded graphs. More recent research has attempted to optimize not only one fixed embedding, but instead to optimize across all possible planar embeddings of a graph. This includes (i) orthogonal drawings [BRW16], (ii) simultaneous embeddings, where one seeks to embed two planar graphs that share a common subgraph

such that they induce the same embedding on the shared subgraph (see [BKR13] for a survey), (iii) simultaneous orthogonal drawings [Ang+16], (iv) embeddings where some edge intersections are allowed [AB19], (v) inserting an edge [GMW05], a vertex [CGMW09], or multiple edges [CH16] into an embedding, (vi) partial embeddings, where one insists that the embedding extends a given embedding of a subgraph [Ang+15c], and (vii) finding minimum-depth embeddings [ADP11, BM90].

The common tool in all of these recent algorithms is the SPQR-tree data structure, which efficiently represents all planar embeddings of a biconnected planar graph G by breaking down the complicated task of choosing a planar embedding of G into the task of independently choosing a planar embedding for each triconnected component of G [DT89, DT90, DT96, HT73, Mac37, Tut66]. This is a much simpler task since the triconnected components have a very restricted structure, and so the components offer only basic, well-structured choices.

For directed graphs there are two different major notions of planarity, namely upward planarity and level planarity. Level planarity can be tested in linear time [FPSS11, JL02, JLM98, Ran+01]. Recently, the problem of extending partial embeddings for level-planar drawings has been studied [BR17]. While the problem is NP-hard in general, it can be solved in polynomial time for single-source graphs. Very recently, an SPQR-tree-like embedding representation for upward planarity has been used to extend partial upward embeddings [BHR19]. The construction crucially relies on an existing decomposition result for upward planar graphs [HL96]. No such result exists for level-planar graphs. Moreover, the level assignment leads to components of different "heights", which makes our decompositions significantly more involved.

Contribution. We develop the LP-tree, an analogue of SPQR-trees for level-planar embeddings of level graphs with a single source whose underlying undirected graph is biconnected. It represents the choice of a level-planar embedding of a level-planar graph by individual embedding choices for certain components of the graph, for each of which the embedding is either unique up to reflection, or allows to arbitrarily permute certain subgraphs around two pole vertices. Its construction is based on suitably modifying the SPQR-tree of G, which represents all planar embeddings of G, not just the level-planar ones, such that, eventually, the modified tree represents exactly the level-planar drawings of G. See Figure 7.1 (a, b) for examples of how level planarity is more restrictive than planarity. The size of the LP-tree is linear in the size of G and it can be computed in linear time. The LP-tree is a useful tool that unlocks the large amount of SPQR-tree-based algorithmic knowledge for easy translation to the level-planar setting. In particular, we obtain linear-time algorithms for partial and constrained level planarity for biconnected single-source level graphs, which improves upon the $O(n^2)$-time algorithm known to date [BR17]. Further, we describe

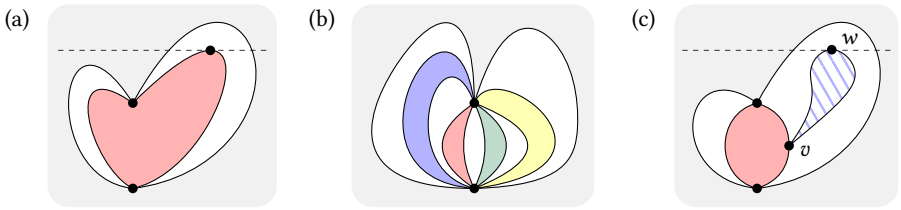


Figure 7.1: In (a), the height of the red component makes it impossible to flip it. In (b), note that the red and green components can be exchanged, as can the blue and yellow components, but neither the blue nor the yellow component can be embedded between the red and green component. In (c), set the demand of v as $d(v) = \ell(w)$ in the LP-tree that represents the graph that consists of the red and gray part (but not the striped blue part). This models the restriction imposed on the embedding of the red subgraph by the striped blue biconnected component.

the first efficient algorithm for the simultaneous level planarity problem when the shared graph is a biconnected single-source level graph.

We first introduce important concepts and notation that we use throughout this chapter in Section 7.2. We show the existence of LP-trees in Section 7.3. The proof is constructive and immediately gives a polynomial-time algorithm, which we then improve to run in linear time. In Section 7.4, we present three applications of LP-trees. Finally, we give some concluding remarks in Section 7.5.

7.2 Preliminaries

Let $G = (V, E)$ be a connected level graph. For each vertex $v \in V$ let $d(v) \geq \ell(v)$ denote the *demand* of v. Demands provide an interface to model the restrictions imposed on the embeddings of one biconnected component by other biconnected components; see Figure 7.1 (c). An *apex* of some vertex set $V' \subseteq V$ is a vertex $v \in V'$ whose level is maximum. The *demand* of V', denoted by $d(V')$, is the maximum demand of a vertex in V'. An apex of a face f is an apex of the vertices incident to f. A *path* is a sequence of vertices $(v_1, v_2, \ldots, v_j)$ so that for $1 \leq i < j$ either (v_i, v_{i+1}) or (v_{i+1}, v_i) is an edge in E. A *directed path* is a sequence $(v_1, v_2, \ldots, v_j)$ of vertices so that for $1 \leq i < j$ it is $(v_i, v_{i+1}) \in E$. A vertex u *dominates* a vertex v if there exists a directed path from u to v. A vertex is a *sink* if it dominates no vertex except for itself. A vertex is a *source* if it is dominated by no vertex except for itself. An *st-graph* is a graph with a single source and a single sink, usually denoted by s and t, respectively. Throughout this chapter all graphs are assumed to have a single source s. For the

remainder of this chapter we restrict our considerations to level-planar drawings of G where each vertex $v \in V$ that is not incident to the outer face is incident to some inner face f so that each apex a of the set of vertices on the boundary of f satisfies $d(v) < \ell(a)$. We will use demands in Section 7.4 to restrict the admissible embeddings of biconnected components in the presence of cutvertices. Note that setting $d(v) = \ell(v)$ for each $v \in V$ gives the conventional definition of level-planar drawings. Note that for single-source level graphs, level-planar embeddings are equivalence classes of topologically equivalent level-planar drawings.

Lemma 47. *The level-planar drawings of a single-source level graph correspond bijectively to its level-planar combinatorial embeddings with s on the outer face.*

Proof. Let $G = (V, E)$ be a single-source k-level graph. Assume without loss of generality that $G = (V, E)$ is proper, i.e., for each edge $(u, v) \in E$ it is $\ell(u) + 1 = \ell(v)$. Let $u, v \in V_i$ be two vertices on level i with $1 \leq i \leq k$. Further, let w be a vertex of G so that there are disjoint directed paths p_u and p_v from w to u and v, respectively. Because G is a single-source graph, such a vertex must exist. Let e and f denote the first edge on p_u and p_v, respectively. Further, let $\prec$ be a level-planar drawing of G and let $\mathcal{G}$ be a level-planar combinatorial embedding of G. If w is not the single source of G, it has an incoming edge g. Then it is $u \prec_i v$ if and only if e, f and g appear in that order around w. Otherwise, if w is the source of G, let g denote the edge (w, t), which exists by construction. Because g is embedded as the leftmost edge, it is $u \prec_i v$ if and only if g, e and f appear in that order around w. The claim then follows easily. □

To make some of the subsequent arguments easier to follow, we preprocess our input level graph G on k levels to a level graph G' on $d(V)+1$ levels as follows. We obtain G' from G by adding a new vertex t on level $d(V)+1$ with demand $d(t) = d(V)+1$, connecting it to all vertices on level k and adding the edge (s, t). Note that G' is generally not an st-graph. Let H be a graph with a level-planar embedding Λ and let H' be a supergraph of H with a level-planar embedding Λ'. The embeddings of G' where the edge (s, t) is incident to the outer face and the embeddings of G are, in a sense, equivalent.

Lemma 48. *An embedding Γ of G is level-planar if and only if there exists a level-planar embedding Γ' of G' that extends Γ where (s, t) is incident to the outer face.*

Proof. Let $G = (V, E)$ be a k-level graph, and let G' be the supergraph of G as described above together with a level-planar embedding Γ'. Because G is a subgraph of G', restricting Γ' to G immediately gives a level-planar embedding Γ of G that is extended by Γ'.

Now let Γ be a level-planar embedding of G. Since all apices of V lie on the outer face, the newly added vertex t can be connected to those vertices without causing any edge crossings. Then, because s is the single source of G and t is the sole apex of $V(G')$, the edge (s,t) can be drawn into the outer face as a y-monotone curve without causing edge crossings. Let Γ' refer to the resulting embedding. Then Γ' is a level-planar embedding of G' that extends Γ. □

To represent all level-planar embeddings of G, it is sufficient to represent all level-planar embeddings of G' and to remove t and its incident edges from all embeddings. It is easily observed that if G is a biconnected single-source graph, then so is G'. We assume from now on that the vertex set of our input graph G has a unique apex t and that G contains the edge (s,t). We still refer to the highest level as level k, i.e., the apex t lies on level k.

Level-planar embeddings of a graph have an important relationship with level-planar embeddings of st-supergraphs thereof. We use Lemmas 49 and 50, and a novel characterization of single-source level planarity in Lemma 51 to prove that certain planar embeddings are also level planar.

Lemma 49. *Let $G = (V,E)$ be a single-source level graph with a unique apex. Further, let Γ be a level-planar embedding of G. Then there exists an st-graph $G_{st} = (V, E \cup E_{st})$ together with a level-planar embedding Γ_{st} that extends Γ.*

Proof. We prove the claim by induction over the number of sinks in G. Note that because t is an apex of G, it must be a sink. So G has at least one sink. If G has one sink, the claim is trivially true for $E_{st} = \emptyset$. Now suppose that G has more than one sink. Let $w \neq t$ be a sink of G. In some drawing of G with embedding Γ, walk up vertically from w into the incident face above w. If a vertex v or an edge (u,v) is encountered, set $E_{st} = \{(w,v)\}$. If no vertex or edge is encountered, w lies on the outer face of Γ. Then set $E_{st} = \{(w,t)\}$. Note that in both cases the added edges can be embedded into Γ as y-monotone curves while maintaining level planarity. Then extend E_{st} inductively, which shows the claim. □

Next we establish a characterization of the planar embeddings that are level planar. The following lemma is implicit in the planarity test for st-graphs by Chiba [CNAO85] and the work on upward planarity by Di Battista and Tamassia [DT88].

Lemma 50. *Let G be an st-graph. Then each planar embedding Γ of G is also a level-planar embedding of G in which (s,t) is incident to the outer face, and vice versa.*

Proof. Consider a vertex $v \neq s,t$ of G. Then the incoming and outgoing edges appear consecutively around v in Γ. To see this, suppose that there are four vertices $w,x,y,z \in V$ with edges $(w,v),(v,x),(y,v),(v,z) \in E$ that appear in that

counter-clockwise cyclic order around v in Γ. Because G is an st-graph there are directed paths p_w and p_y from s to w and y, respectively, and directed paths p_x and p_z from x and z to t, respectively. Moreover, $p \in \{p_w, p_y\}$ and $p' \in \{p_x, p_z\}$ are disjoint and do not contain v. Then some $p \in \{p_w, p_y\}$ and $p' \in \{p_x, p_z\}$ must intersect, a contradiction to the fact that Γ is planar; see Fig. 7.2 (a).

Let $e_1, e_2, \ldots, e_i, e_{i+1}, \ldots, e_n$ denote the counter-clockwise cyclic order of edges around v in Γ so that $e_1, \ldots, e_i$ are incoming edges and $e_{i+1}, \ldots, e_n$ are outgoing edges. Let $e_1, \ldots, e_i$ denote the left-to-right order of incoming edges and let $e_n, e_{n-1}, \ldots, e_{i+1}$ denote the left-to-right order of outgoing edges. Split the clockwise cyclic order of edges around s at (s, t) to obtain the left-to-right order of outgoing edges. Symmetrically, split counter-clockwise order of edges around t at (s, t) to obtain the left-to-right order of incoming edges.

Create a level-planar embedding Γ' of G step by step as follows; see Fig. 7.2. Draw vertices s and t on levels $\ell(s)$ and $\ell(t)$, respectively, and connect them by a straight line segment. Call the vertices s, t and the edge (s, t) *discovered*. Call the path s, t the *right frontier*. Call a vertex on the right frontier *settled* if all of its outgoing edges are discovered.

More generally, let $s = u_1, u_2, \ldots, u_n = t$ denote the right frontier. Modify the right frontier while maintaining that (i) the right frontier is a directed path from s to t, (ii) any edge (u_a, u_{a+1}) on the right frontier is the rightmost discovered outgoing edge around u_a, and (iii) the right frontier is incident to the outer face of Γ'.

Let u_i denote the vertex on the right frontier closest to t that is not settled. Discover the leftmost undiscovered outgoing edges starting from u_i to construct a directed path $v_1 = u_i, v_2, \ldots, v_m$, where v_m is the first vertex that had been discovered before. Because G has a single sink such a vertex exists. Because Γ is planar v_m lies on the right frontier, i.e., $v_m = u_j$ for some j with $i < j \leq n$. Insert the vertices $v_2, \ldots, v_{m-1}$ and the edges (v_a, v_{a+1}) for $1 \leq a < m$ to the right of the path $u_i, \ldots, u_j$ into Γ' (Property (iii) of the invariant), maintaining level planarity of Γ'. This creates a new face f of Γ' whose boundary is $u_i, u_{i+1}, \ldots, u_j = v_m, v_{m-1}, \ldots, v_1 = u_i$.

We show that f is a face of Γ. Because u_a is settled there cannot be an undiscovered outgoing edge between (u_{a-1}, u_a) and (u_a, u_{a+1}) in the counter-clockwise order of edges around u_a in Γ for $i < a < j$ (see edge g in Fig. 7.2 (b)). There can also not be a discovered outgoing edge because of Property (ii) of the invariant (see edge e in Fig. 7.2 (b)). Because the leftmost undiscovered edge is chosen there is no undiscovered outgoing edge between (v_a, v_{a+1}) and (v_{a-1}, v_a) in the counter-clockwise order of edges around v_a in Γ for $1 < a < m$ (see edge h in Fig. 7.2 (b)). There can also not be a discovered outgoing edge because v_a was not discovered before (see edge q in Fig. 7.2 (b)). There can be no outgoing edge between (v_1, v_2) and (u_i, u_{i+1}) in the counter-clockwise order of edges around $v_1 = u_i$ because either such an edge would be discovered contradicting Property (ii), or not, contradicting the fact

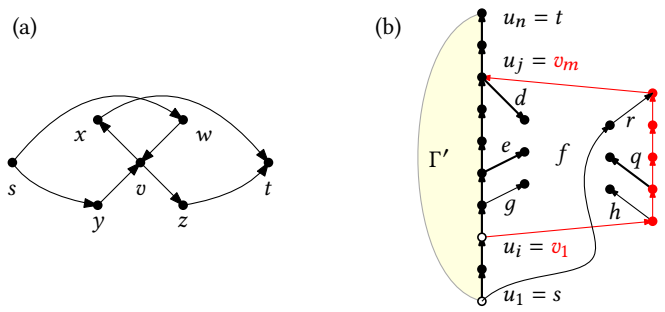


Figure 7.2: Proof of Lemma 50. The incoming and outgoing edges around each vertex are consecutive (a). Creating the level-planar embedding Γ' by attaching the path $v_1, v_2, \ldots, v_m$ (drawn in red) to the right frontier $u_1, u_2, \ldots, u_n$, thereby creating a new face f. Discovered edges are drawn thickly. The edges e, g, h, q, r, d cannot exist.

that (v_1, v_2) is chosen as the leftmost undiscovered outgoing edge of v_1. There can be no outgoing edge between (u_{j-1}, u_j) and (v_{m-1}, v_m) in the counter-clockwise order of edges around $u_j = v_m$ because either $u_j = v_m = t$ is a sink, or the incoming and outgoing edges appear consecutively around $u_j = v_m$ in Γ (see edge d in Fig. 7.2 (b)).

There can also be no incoming edge (u, v) between any of these edge pairs (see edge r in Fig. 7.2 (b)). This is because G has a single source s, so there exists a directed path p from s to u. Because u lies inside of f the path p must contain a vertex x on the boundary of f. Then p would also contain an outgoing edge of x which we have just shown to be impossible.

Let $s = u_1, u_2, \ldots, u_i = v_1, v_2, \ldots, v_m = u_j, \ldots, u_n = t$ denote the new right frontier. Note that the invariant holds for this modified right frontier. Because G has a single-source all vertices and edges are drawn in this way. Because Γ and Γ' have the same faces they are the same embedding. Finally, Γ' is level planar by construction, which shows the claim. □

Thus, a planar embedding Γ of a graph G is level-planar if and only if it can be augmented to an st-graph $G' \supseteq G$ such that all augmentation edges can be embedded in the faces of Γ without crossings. This gives rise to the following characterization.

Lemma 51. *Let G be a single-source k-level graph with a unique apex t. Then G is level planar if and only if it has a planar embedding where every vertex v with $\ell(v) < k$ is incident to at least one face f so that v is not an apex of f.*

Proof. Let Γ_l be a level-planar drawing of G. Consider a vertex v such that it is $\ell(v) < \ell(t)$. If v has an outgoing edge (v, w), then v and w are incident to

some shared face f. Because it is $\ell(v) < \ell(w)$, vertex v is not an apex of f. If v has no outgoing edges, start walking upwards from v in a straight line. Stop walking upwards if an edge (u, w) or a vertex w is encountered. Then v and w are again incident to some shared face f. Moreover, it is $\ell(v) < \ell(w)$, and therefore v is not an apex of f. If no edge or vertex is encountered when walking upwards, v must lie on the outer face. Because t lies on the outer face and it is $\ell(v) < \ell(t)$, vertex v is not an apex of the outer face. Finally, because Γ_l is level planar it is, of course, also planar.

Now let Γ_p be a planar embedding of G. The idea is to augment G and Γ_p by inserting edges so that G becomes an st-graph together with a planar embedding Γ_p. To that end, consider a sink $v \neq t$ of G. By assumption, v is incident to at least one face f so that v is not an apex of f. Hence, it is $\ell(v) < \ell(\text{apex}(f))$. So the augmentation edge $e = (v, \text{apex}(f))$ can be inserted into G without creating a cycle. Further, e can be embedded into f. Because all augmentation edges embedded into f have endpoint $\text{apex}(f)$, the embedding Γ_p of G remains planar. This means that G can be augmented so that t becomes the only sink while maintaining the planarity of Γ_p. Because G also has a single source, G is now an st-graph and it follows from Lemma 50 that Γ_p is not only planar, but also level planar. □

7.3 A Decomposition Tree for Level Planarity

We construct a decomposition tree of a given single-source level graph G whose underlying undirected graph is biconnected that represents all level-planar embeddings of G, called the *LP-tree*. As noted in the Preliminaries, we assume that G has a unique apex t, for which $\ell(t) = d(t)$ holds true. The LP-tree for G is constructed based on the SPQR-tree for G. We keep the notion of S-, P-, Q- and R-nodes and construct the LP-tree so that the nodes behave similarly to their namesakes in the SPQR-tree. The skeleton of a P-node consists of two vertices that are connected by at least three parallel virtual edges that can be arbitrarily permuted. The skeleton of an R-node μ is equipped with a *reference embedding* Γ_μ, and the choice of embeddings for such a node is limited to either Γ_μ or its reflection. Unlike in SPQR-trees, the skeleton of μ need not be triconnected, instead it can be an arbitrary biconnected planar graph. The embedding of R-node skeletons being fixed up to reflection allows us to again use the equivalence of the arc-based and the skeleton-based embedding representations.

The construction of the LP-tree starts out with an SPQR-tree $\mathcal{T}$ of G. Explicitly label each node of $\mathcal{T}$ as an S-, P-, Q- or R-node. This way, we can continue to talk about S-, P-, Q- and R-nodes of our decomposition tree even when they no longer have their defining properties in the sense of SPQR-trees. Assume the edge (s, t) to be incident to the outer face of every level-planar drawing of G (Lemma 48), i.e., consider $\mathcal{T}$ rooted at the Q-node corresponding to (s, t). The construction of

our decomposition tree works in two steps. First, decompose the graph further by decomposing P-nodes in order to disallow permutations that lead to embeddings that are not level planar. Second, contract arcs of the decomposition tree, each time fixing a reference embedding for the resulting node, so that we can consider it as an R-node, such that the resulting decomposition tree represents exactly the level-planar embeddings of G. The remainder of this section is structured as follows. The details and correctness of the first step are given in Section 7.3.1. Section 7.3.2 gives the algorithm for constructing the final decomposition tree $\mathcal{T}$. It follows from the construction that all embeddings it represents are level-planar, and Section 7.3.3 shows that, conversely, it also represents every level-planar embedding. In Section 7.3.4, we present a linear-time implementation of the construction algorithm.

7.3.1 P-Node Splits

In SPQR-trees, the children of P-nodes can be arbitrarily permuted. We would like P-nodes of the LP-tree to have the same property. Hence, we decompose skeletons of P-nodes to disallow orders that lead to embeddings that are not level planar. The decomposition is based on the height of the child virtual edges, which we define as follows. Let μ be a node of a rooted decomposition tree and let u and v be the poles of μ. Define $V(\mu) = V(G(\mu)) \setminus \{u, v\}$. The *height* of μ and of the child virtual edge e with $\text{corr}(e) = \mu$ is $d(\mu) = d(e) = d(V(\mu))$.

Now let μ be a P-node, and let Γ be a level-planar embedding of G. The embedding Γ induces a linear order of the child virtual edges of μ. This order can be obtained by splitting the combinatorial embedding of $\text{skel}(\mu)$ around u at the parent edge. Then the following is true.

Lemma 52. *Let $\mathcal{T}$ be a decomposition tree of G, let μ be a P-node of $\mathcal{T}$ with poles u, v, and let $e_{\max}$ be a child virtual edge of μ with maximal height. Further, let Γ be a level-planar embedding of G that is represented by $\mathcal{T}$. If the height of $e_{\max}$ is at least $\ell(v)$, then $e_{\max}$ is either the first or the last edge in the linear ordering of the child virtual edges induced by Γ.*

Proof. Let $v = \text{corr}_\mu(e_{\max})$. Further, let $G_{\max} = G(e_{\max})$, and let $w \in V(v)$ such that $d(w) = d(v)$. If $d(w) < \ell(v)$, the statement of the lemma is trivially satisfied, so assume $d(w) \geq \ell(v)$ and suppose that $e_{\max}$ is not the first edge or last edge. Let Γ_μ be the embedding of $\text{skel}(\mu)$ in the corresponding skeleton-based representation of Γ. Then there are child virtual edges e_1, e_2 immediately preceding and succeeding edge $e_{\max}$ in Γ_μ, respectively. By construction of the embedding Γ via contractions from the embeddings of skeletons, it follows that w shares a face only with the inner vertices of $G(e_i)$ for $i = 1, 2$, the inner vertices of $G_{\max}$, and u and v. By the choice of $e_{\max}$ it follows that $d(w) \geq \ell(w')$ for all inner vertices w' of $G(e_i)$, $i = 1, 2$, and

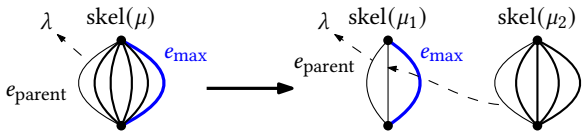


Figure 7.3: Result of a P-node μ split with parent λ and child with maximum height ν. Note that after the split, μ_1 is an R-node and μ_2 has one less child than μ had.

the choice of w guarantees that $d(w) \geq \ell(w')$ for all inner vertices w' of $G(e_{\max})$. Moreover, it is $d(w) \geq \ell(v) \geq \ell(u)$ by assumption. It follows that w is not incident to any face that has an apex a with $d(w) < \ell(a)$. Because w is an inner vertex of $G_{\max}$ it is not incident to the outer face. Thus, Γ is not level-planar by Lemma 51, a contradiction. □

Lemma 52 motivates the following modification of a decomposition tree $\mathcal{T}$. Take a P-node μ with poles u, v that has a child edge whose height is at least $\ell(v)$. Denote by λ the parent of μ. Further, let $e_{\max}$ be a child virtual edge with maximum height and let e_{parent} denote the parent edge of skel(μ). Obtain a new decomposition tree $\mathcal{T}'$ by splitting μ into two nodes μ_1 and μ_2 representing the subgraph H_1 consisting of the edges $e_{\max}$ and e_{parent}, and the subgraph H_2 consisting of the remaining child virtual edges, respectively; see Fig. 7.3. Note that the skeleton of μ_1, which corresponds to H_1, has only two child virtual edges. We therefore define it to be an R-node. Moreover, observe that in any embedding of skel(μ) that is obtained from choosing embeddings for skel(μ_1) and skel(μ_2) and contracting the arc (μ_1, μ_2), the edge $e_{\max}$ is the first or last child edge. Conversely, because μ_2 is a P-node, all embeddings where $e_{\max}$ is the first or last child edge are still represented by $\mathcal{T}'$. Apply this decomposition iteratively, creating new R-nodes on the way, until each P-node μ with poles u and v has only child virtual edges e that have height at most $\ell(v) - 1$. We say that a node ν with poles x, y has *I shape* when the height of $G(\nu)$ is less than $\ell(y)$. The following theorem sets the stage to prove that after this decomposition, the children of P-nodes can be arbitrarily permuted.

Theorem 13. *Let G be a biconnected single-source graph with unique apex t. There exists a decomposition tree $\mathcal{T}$ that represents all level-planar embeddings of G such that all children of P-nodes in $\mathcal{T}$ have I shape.*

We see that this property ensures that P-nodes in our decomposition of level-planar graphs work analogously to those of SPQR-trees for planar graphs. Namely, if we have a level-planar embedding Γ of G and consider a new embedding Γ' that is obtained from Γ by reordering the children of P-nodes, then also Γ' is level-planar.

We show that the st-augmentation from Lemma 49 can be assumed to have certain useful properties. The proof that the children of P-nodes can be arbitrarily permuted then uses Lemma 50 and the fact that the children of P-nodes in SPQR-trees can be arbitrarily permuted.

Lemma 53. *Let Γ be a level-planar embedding of $G = (V, E)$ and let μ be a node of $\mathcal{T}$ with poles u, v such that $G(\mu)$ has I shape. Then there exists a planar st-augmentation $G' = (V, E \cup E_{st}), \Gamma'$ of G and Γ so that u, v separates $V(G(\mu))$ from $V \setminus V(G(\mu))$ in G'.*

Proof. Let Γ' and G' be an st-augmentation of Γ and G where u, v is not a cutpair. Modify G', Γ' so that they remain an st-augmentation of G, Γ and no edge in E_{st} has exactly one endpoint in $V(G(\mu))$. Let f be a μ-incident arc in Γ. Let $E(f)$ denote the set of augmentation edges embedded into f to obtain Γ'. Call an edge $(w, x) \in E(f)$ *critical* if w or x lies in $V(\mu)$. Remove all critical edges from Γ' and G. Note that because u, v is a cutpair in G, the endpoints of all critical edges are now incident to the same face f'. Observe that v is also incident to f'. Consider a critical edge (w, x) that was removed. Because $G(\mu)$ has I shape, it follows from $w \in V(\mu)$ that it is certainly $\ell(w) < \ell(v)$. If it is $w \notin V(\mu)$, then it must be $x \in V(\mu)$ and certainly $\ell(x) < \ell(v)$. With $\ell(w) < \ell(x)$ it follows that $\ell(w) < \ell(v)$. So for each critical edge (w, x) the non-critical edge (w, v) can be added to Γ' and G'. Because all endpoints are incident to f' and all inserted edges share the endpoint v this preserves the planarity of Γ' and G'. Therefore, Γ' and G' is now an st-augmentation of Γ and G once more. Finally, u and v separate $V(G(\mu))$ from $V \setminus V(G(\mu))$ in G' because G' contains no critical edge. □

This sets the stage for the correctness proof. The idea is to transform any given st-augmentation to one that satisfies the conditions from Lemma 53. Then the graphs corresponding to child virtual edges can be permuted arbitrarily while preserving planarity. Lemma 50 then gives that all these embeddings are also level planar.

Lemma 54. *Let Γ be a level-planar embedding of G and let $\mathcal{T}$ be a decomposition tree of G whose skeletons are embedded according to Γ. Further, let μ be a P-node of $\mathcal{T}$. Let Γ' be the planar embedding obtained by arbitrarily permuting the child virtual edges of μ. Then Γ' is level planar.*

Proof. Let Γ' and G' be an st-augmentation obtained from Γ and G according to Lemma 53. Note that (u, v) separates $G'(\nu)$ from the rest of G' for each child ν of μ. Consider the SPQR-tree $\mathcal{T}'$ of G'. Then u, v are the poles of a P-node μ' in $\mathcal{T}'$ with the same neighbors as μ in $\mathcal{T}$. Then the child virtual edges of $\text{skel}(\mu')$ can be arbitrarily permuted to obtain a planar embedding. Because G' is an st-graph, Lemma 50 gives that any planar embedding of G' is also level planar. □

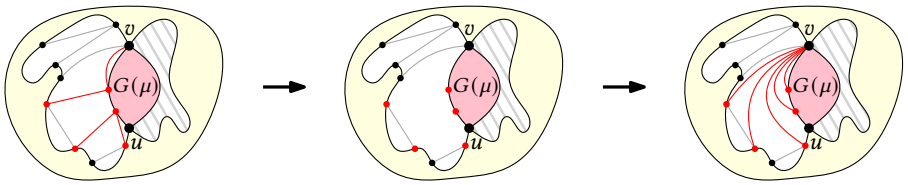


Figure 7.4: The three steps in the proof of Lemma 53. The subgraph $G(\mu)$ is drawn in pink, the μ-incident face f is drawn in white. Critical augmentation edges in $E(f)$ are drawn in red, and non-critical augmentation edges are drawn in gray. In the first step, remove all critical edges, this gives the drawing in the middle. Note that the red vertices and v are incident to a shared face. Finally, attach all red vertices to v, this gives the drawing on the right. The same process would then be repeated for the other μ-incident face, drawn with gray stripes.

This completes the proof that in our decomposition the children of P-nodes can be arbitrarily permuted.

Theorem 14. *Let G be a biconnected single-source graph with a unique apex. There exists a decomposition tree $\mathcal{T}$ that (i) represents all level-planar embeddings of G (plus some planar, non-level-planar ones), and (ii) if all skeletons of the nodes of $\mathcal{T}$ are embedded so that contracting all arcs of $\mathcal{T}$ yields a level-planar embedding, then the children of all P-nodes in $\mathcal{T}$ can be arbitrarily permuted and then contracting all arcs of $\mathcal{T}$ still yields a level-planar embedding of G.*

7.3.2 Arc Processing

In this section, we finish the construction of the LP-tree. The basis of our construction is the decomposition tree $\mathcal{T}$ from Theorem 13, which represents a subset of the planar embeddings of G that contains all level-planar embeddings, and moreover all children of P-nodes have I shape. We now restrict $\mathcal{T}$ even further until it represents exactly the level-planar embeddings of G. As of now, all R-node skeletons have a planar embedding that is unique up to reflection, as they are either triconnected or consist of only three parallel edges. By assumption, G is level-planar, and there exists a level-planar embedding Γ of G. Recall that our definition of level-planar embeddings involves demands. Computing a level-planar embedding Γ of G with demands reduces to computing a level-planar embedding of the supergraph G' of G obtained from G by attaching to each vertex v of G with $d(v) > \ell(v)$ an edge to a vertex v' with $\ell(v') = d(v)$ without demands. Because G' is a single-source graph whose size is linear in the size of G this can be done in linear time [DN88]. We equip

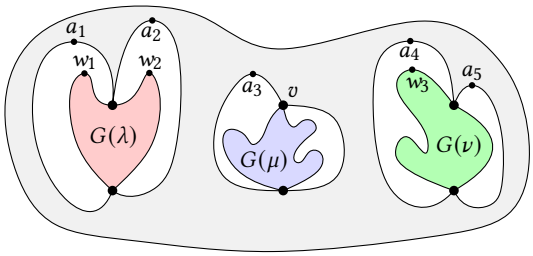


Figure 7.5: The height of $G(\lambda)$ is at least $\ell(w_1) = \ell(w_2)$, the height of $G(\mu)$ is at most $\ell(v) - 1$ and the height of $G(\nu)$ is at least $\ell(w_3)$. The space around λ is $\ell(a_1)$, the space around μ is $\ell(v)$ and the space around ν is $\ell(a_5)$.

the skeleton of each node μ with the reference embedding Γ_μ such that contracting all arcs yields the embedding Γ. For the remainder of this section we will work with the arc-based embedding representation. As a first step, we contract any arc (λ, μ) of $\mathcal{T}$ where λ is an R-node and μ is an S-node and label the resulting node as an R-node. Note that, since S-nodes do not offer any embedding choices, this does not change the embeddings that are represented by $\mathcal{T}$. This step makes the correctness proof easier. Any remaining arc (λ, μ) of $\mathcal{T}$ is contracted based upon two properties of μ, namely the height of $G(\mu)$ and the space around μ in the level-planar embedding Γ, which we define next. The resulting node is again labeled as an R-node. Let μ be a node of $\mathcal{T}$ with poles u and v. We denote by $\Gamma \circ \mu$ the embedding obtained from Γ by contracting $G(\mu)$ to the single edge $e = (u, v)$. We call the faces f_1, f_2 of Γ that induce the incident faces of e in $\Gamma \circ \mu$ the *μ-incident faces*. The *space around μ in* Γ is $\min\{\ell(\text{apex}(f_1)), \ell(\text{apex}(f_2))\}$; see Fig. 7.5. For the time being we will consider the embeddings of P-node skeletons as fixed. Then all the remaining embedding choices are done by choosing whether or not to flip the embedding for the incoming arc of each R-node. Let A denote the set of arcs in $\mathcal{T}$. For each arc $a = (\lambda, \mu) \in A$ let $\text{space}(\mu)$ denote the space around μ in Γ. We label a as *rigid* if $d(\mu) \geq \text{space}(\mu)$ and as *flexible* otherwise.

Let $\mathcal{T}'$ be the decomposition tree obtained by contracting all rigid arcs and equipping each R-node skeleton with the reference embedding obtained from the contractions. We now release the fixed embedding of the P-nodes, allowing to permute their children arbitrarily. The resulting decomposition tree is called the *LP-tree* of the input graph G. See Fig. 7.6 (d) for an example. Our main result is the following theorem.

Theorem 15. *Let G be a biconnected, single-source, level-planar graph. The LP-tree of G represents exactly the level-planar embeddings of G and can be computed in linear time.*

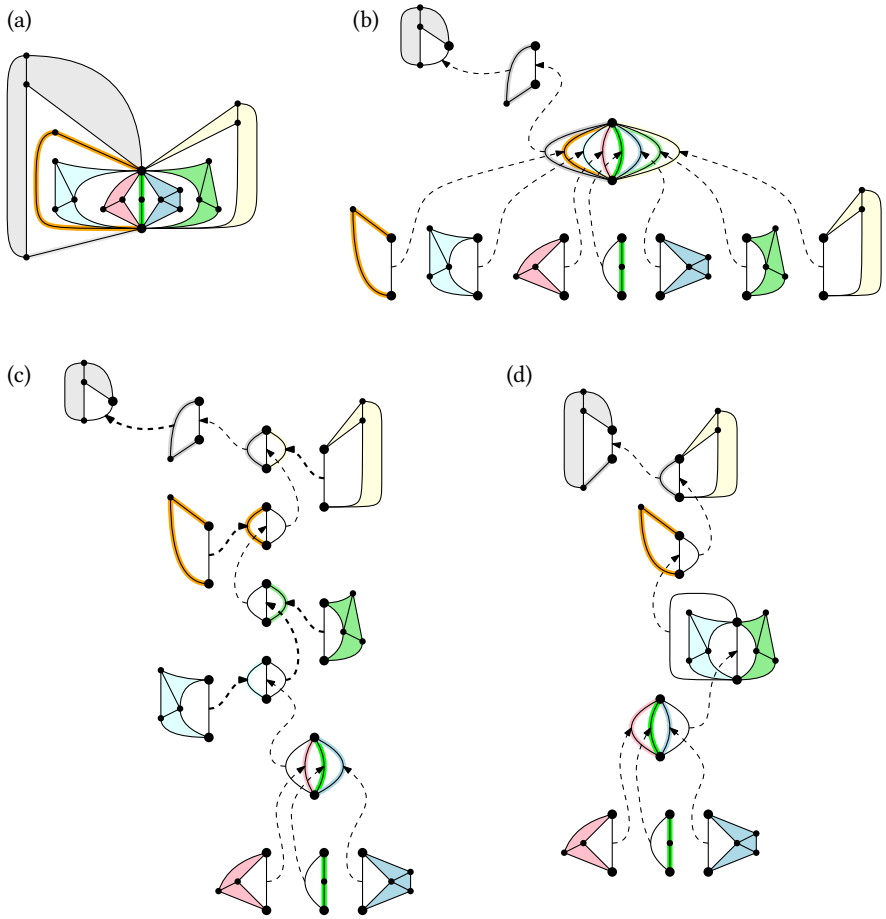


Figure 7.6: Example construction of the LP-tree for the graph G (a). We start with the SPQR-tree of G (b). Arcs are oriented towards the root. Next, we split the P-node, obtaining the tree shown in (c). Finally, we contract arcs that connect R-nodes with S-nodes and arcs that are found to be rigid (thick dashed lines). This gives the final LP-tree $\mathcal{T}$ for G (d).

The next subsection is dedicated to proving the correctness of Theorem 15. The above algorithm considers every arc of $\mathcal{T}$ once. The height of μ and the space around μ in Γ can be computed in polynomial time. Thus, the algorithm has overall

polynomial running time. In Section 7.3.4, we present a linear-time implementation of this algorithm.

7.3.3 Correctness

Process the arcs in top-down order $\alpha_1, \dots, \alpha_m$. For each $i = 0, 1, \dots, m$ define the set $A_i = \{\alpha_1, \dots, \alpha_i\}$ as the first i processed arcs for $i = 0, \dots, m$. Note that $A_0 = \emptyset$ and $A_m = A$. Denote by R_i and F_i the arcs in A_i that are labeled rigid and flexible, respectively. We now introduce a refinement of the embeddings represented by a decomposition tree. Namely, a *restricted decomposition tree* $\mathcal{T}$ is a decomposition tree together with a subset of its arcs that are labeled as flexible, and, in the arc-based view, the embeddings represented by $\mathcal{T}$ are only those that can be created by flipping only at flexible arcs. We denote by $\mathcal{T}_i$ the restricted decomposition tree obtained from $\mathcal{T}$ by marking only the edges in F_i as flexible.

Initially, $F_0 = \emptyset$, and therefore $\mathcal{T}$ represents exactly the reference embedding Γ_{ref} and its reflection. Since all children of P-nodes have I shape and each P-node has I shape, no arc incident to a P-node is labeled *rigid*. Therefore, if such an edge is contained in A_i, it is flexible. In particular, only arcs between adjacent R-nodes are labeled rigid. As we proceed and label more edges as *flexible*, more and more embeddings are represented. Each time, we justify the level planarity of these embeddings. As a first step, we extend the definition of space from the previous subsection, which strongly depends on the initial level-planar embedding Γ, in terms of all level-planar embeddings represented by the restricted decomposition tree $\mathcal{T}_i$. Let μ be a node of $\mathcal{T}_i$ with poles u, v. The *space around* μ is the minimum space around μ in any level-planar embedding represented by the restricted decomposition tree $\mathcal{T}_i$. Now let Γ be a planar embedding of G and let Π be a planar embedding of $G(\mu)$ where u and v lie on the outer face. Because u and v is a cutpair that disconnects $G(\mu)$ from the rest of G and $G(\mu)$ is connected, the embedding of $G(\mu)$ in Γ can be replaced by Π. Let $\Gamma + \Pi$ refer to the resulting embedding. Now let Γ be a planar embedding of G and let μ be a node of $\mathcal{T}$. Let Π denote the restriction of Γ to $G(\mu)$ and let $\bar{\Pi}$ be the reflection of Π. *Reflecting* μ in T corresponds to replacing Π by $\bar{\Pi}$ in Γ, obtaining the embedding $\Gamma + \bar{\Pi}$ of G.

The idea is to show that if there is (is not) enough space around a node μ to reflect it, it can (cannot) be reflected regardless of which level-planar embedding is chosen for $G(\mu)$. So, the algorithm always labels arcs correctly. We use the following invariant.

Lemma 55. *The restricted decomposition tree $\mathcal{T}_i$ satisfies the following five conditions.*

1. *All embeddings represented by $\mathcal{T}_i$ are level planar.*
2. *Let (λ, μ) be an arc that is labeled as flexible. Let Γ be an embedding represented*

by $\mathcal{T}_{i-1}$ and let Π be any level-planar embedding of $G(\mu)$. Then $\Gamma + \Pi$ and $\Gamma + \bar{\Pi}$ are level planar.

3. *Let (λ, μ) be an arc that is labeled as rigid. Let Γ be an embedding represented by $\mathcal{T}_{i-1}$ and let Π be a level-planar embedding of $G(\mu)$ so that $\Gamma + \Pi$ is level planar. Let all skeletons of $\mathcal{T}_i$ be embedded according to $\Gamma + \Pi$. Then $\mathrm{skel}(\mu)$ has the reference embedding and $\Gamma + \bar{\Pi}$ is not level planar.*
4. *The space around each node μ of $\mathcal{T}_i$ is the same across all embeddings represented by $\mathcal{T}_i$.*
5. *Let Γ be a level-planar embedding of G so that there exists a level-planar embedding Γ_p of G that (i) is obtained from Γ by reordering the children of P-nodes, and (ii) satisfies $\Gamma_p = \Gamma_{\mathrm{ref}}(\pi_1, \pi_2, \ldots, \pi_m)$ where π_j indicates whether arc $\alpha_j = (\lambda_j, \mu_j)$ should be flipped ($\pi_j = \bar{\alpha}_j$) or not ($\pi_j = \alpha_j$), and it is $\pi_j = \alpha_j$ for $j > i$. Then Γ is represented by $\mathcal{T}_i$.*

Proof. For $i = 0$, no arc of the restricted decomposition tree $\mathcal{T}_0$ is labeled as flexible. So $\mathcal{T}_0$ only represents the reference embedding Γ_{ref} and its reflection $\bar{\Gamma}_{\mathrm{ref}}$. Both of these are level planar by assumption, so condition 1 is satisfied. Because $A_0 = \emptyset$, no arc has been labeled as flexible or rigid, so conditions 2 and 3 are trivially satisfied. Because the incidences of vertices and faces are the same in Γ and its reflection $\bar{\Gamma}$, condition 4 is also satisfied.

Now consider the case $i \geq 1$. Let $\alpha_i = (\lambda, \mu)$. Let u, v be the poles of μ. Let Γ be an embedding represented by $\mathcal{T}_{i-1}$ and let Π be any level-planar embedding of $G(\mu)$. Consider the embedding $\Gamma + \Pi$. Let f_1, f_2 be the μ-incident faces of $\Gamma + \Pi$. For $j = 1, 2$, let W_j be the subset of vertices of $G(\mu)$ that are incident to f_j, except for u and v. And let V_j be all other vertices incident to f_j, including u and v. Now consider the embedding $\Gamma + \bar{\Pi}$. Again, let f_1', f_2' be the μ-incident faces of $\Gamma + \bar{\Pi}$. Then $V_1 \cup W_2$ and $V_2 \cup W_1$ are the set of vertices incident to f_1' and f_2', respectively. Note that all faces in $\Gamma + \bar{\Pi}$ except for f_1', f_2' appear identically in $\Gamma + \Pi$. Let a_1 and a_2 denote the apices of f_1 and f_2, respectively. Then the space around μ in $\Gamma + \Pi$, denoted by $\mathrm{space}(\mu)$, is $\min(\ell(a_1), \ell(a_2))$. Distinguish two cases, namely $\mathrm{height}(\mu) < \mathrm{space}(\mu)$ and $\mathrm{height}(\mu) \geq \mathrm{space}(\mu)$. Note that because of condition 4, the same case applies for any embedding represented by $\mathcal{T}_{i-1}$.

1. Consider the case $\mathrm{height}(\mu) < \mathrm{space}(\mu)$. This implies $a_1 \in V_1$ and $a_2 \in V_2$. We have to show that both $\Gamma + \Pi$ and $\Gamma + \bar{\Pi}$ are level planar. To this end, use Lemma 51. By assumption, Π is a level-planar embedding of $G(\mu)$. So the condition of Lemma 51 is satisfied for any vertex of G whose incident faces are all inner faces of Π in $\Gamma + \Pi$ (or of $\bar{\Pi}$ in $\Gamma + \bar{\Pi}$). By condition 1, Γ is a level-planar embedding of G. So the condition of Lemma 51 is satisfied for any vertex of $G \setminus G(\mu)$ that is not incident to f_1 and f_2. It remains to be shown that the condition of Lemma 51 is satisfied for the vertices in $(V_1 \setminus \{a_1\} \cup (V_2 \setminus \{a_2\}) \cup W_1 \cup W_2$.

- Suppose $w \in V_1 \setminus \{a_1\}$. Then w is incident to f_1', as are the vertices in V_1. In particular, because $a_1 \in V_1$, the apex a_1 is incident to f_1'. And because a_1 is the unique apex of f_1, it is $\ell(w) < \ell(a_1)$. The argument works analogously for $w \in V_2 \setminus \{a_2\}$.
- Otherwise, it is $w \in W_1$.
  - Consider $\Gamma + \Pi$. Then w is incident to f_1', as are the vertices in V_1. In particular, a_1 is incident to f_1'. Further, $\text{space}(\mu) = \min(\ell(a_1), \ell(a_2))$. So it is
    $$\ell(w) \leq \text{height}(\mu) < \text{space}(\mu) \leq \ell(a_1)$$
    and it follows that $\ell(w) < \ell(a_1)$.
  - Consider $\Gamma + \bar{\Pi}$. Then w is incident to f_2', as are the vertices in V_2. In particular, a_2 is incident to f_2'. Further, $\text{space}(\mu) = \min(\ell(a_1), \ell(a_2))$. So it is
    $$\ell(w) \leq \text{height}(\mu) < \text{space}(\mu) \leq \ell(a_2)$$
    and it follows that $\ell(w) < \ell(a_2)$.

  The argument works analogously for $w \in W_2$.

This shows that the condition in Lemma 51 is satisfied for all vertices in $\Gamma + \Pi$ and $\Gamma + \bar{\Pi}$. As a result, both of these embeddings are level planar.

2. Consider the case $\text{height}(\mu) \geq \text{space}(\mu)$. Then the algorithm will find the arc α_i to be rigid and we have to show that this is the correct choice. Note that as observed above, the fact that α_i is labeled as rigid means that μ is an R-node. Recall that u, v are the poles of μ and let $w \neq u, v$ be a vertex of $G(\mu)$ so that $\ell(w)$ equals $\text{height}(\mu)$. Note that it is $w \neq v$ by definition of $\text{height}(\mu)$ and $w \neq u$ because of $\text{height}(\mu) \geq \text{space}(\mu)$. Again, because of $\text{height}(\mu) \geq \text{space}(\mu)$, the apex w lies on the outer face of Π. Either w is a vertex on the outer face of $\text{skel}(\mu)$, or w belongs to $G(e)$ for some child virtual edge e on the outer face of $\text{skel}(\mu)$. Because μ is an R-node, its skeleton is biconnected and therefore w is incident to either f_1 or f_2, but not both, and this choice depends entirely on the embedding of $\text{skel}(\mu)$. By assumption $\Gamma + \Pi$ is level planar and it remains to be shown that $\Gamma + \bar{\Pi}$, is not level planar. Note that $\Gamma + \bar{\Pi}$ is the embedding that is obtained by reflecting μ so that $\text{skel}(\mu)$ does not have the reference embedding. Assume $w \in W_1$ without loss of generality. It is $\ell(w) = \max\{\ell(x) \mid x \in W_1\}$. Because w is an apex of $V(\mu)$, face f_1 must be the face incident to w of which w is not an apex. Now consider $\Gamma + \bar{\Pi}$. Now w is incident to face f_2' which is incident to the vertices $V_2 \cup W_1$. Because $\text{height}(\mu) \geq \text{space}(\mu)$ it is $\ell(w) \geq \max\{w \in V_2 \cup W_1\}$. This means

that w is an apex of all its incident faces. Then $\Gamma + \bar{\Pi}$ cannot be level planar by Lemma 51.

This means that if a is labeled as flexible, then $G(\mu)$ can be reflected in all embeddings represented by $\mathcal{T}_{i-1}$. And if a is labeled as rigid, then $G(\mu)$ cannot be reflected in any embedding represented by $\mathcal{T}_{i-1}$. This shows that $\mathcal{T}_i$ satisfies conditions 1 through 3. Next, we show that the space around nodes of $\mathcal{T}$ is the same across all embeddings represented by $\mathcal{T}_i$. Again, distinguish the two cases $\text{height}(\mu) < \text{space}(\mu)$ and $\text{height}(\mu) \geq \text{space}(\mu)$.

1. Consider the case $\text{height}(\mu) < \text{space}(\mu)$. Let Γ be an embedding represented by $\mathcal{T}_{i-1}$ and let Γ' be the embedding obtained by reflecting μ in Γ. See Fig. 7.7. We show that the space around each node ν of $\mathcal{T}_i$ is identical in Γ and Γ'. Let x, y be the poles of ν and let f_1, f_2 be the μ-incident faces in Γ. Further, let f_1', f_2' be the μ-incident faces in Γ'. As previously discussed, all faces in Γ and Γ' are identical, except for f_1, f_2, f_1', f_2'. Suppose that both ν-incident faces in Γ are neither f_1 nor f_2. Then the faces around ν do not change and therefore the space around ν does not change. Conversely, suppose that the ν-incident faces are f_1 and f_2. Then the space around ν in Γ is $\min(\ell(a_1), \ell(a_2))$. And because $a_j \in V_j$ for $j = 1, 2$, the space around ν in Γ' is $\min(\ell(a_1), \ell(a_2))$ as well.

   Otherwise, exactly one ν-incident face in Γ is either f_1 or f_2. Without loss of generality, let f_1 be that face. Then exactly one ν-incident face in Γ' is either f_1' or f_2'. Assume that face is f_1'. Because the apex of f_1' and f_1 are identical, the space around ν in Γ' is the same as in Γ. Now assume that f_2' is the face. Then the space around ν is bounded by a vertex $z \in V(G(\mu))$ and $\text{height}(\mu) < \text{space}(\mu)$ implies that $\ell(z) \leq \text{height}(\mu) < \text{space}(\mu)$. So the space around ν is bounded by z in Γ and Γ'.

   Again, because of condition 4, the argument can be made for any embedding represented by $\mathcal{T}_{i-1}$, and therefore the claim follows for all embeddings represented by $\mathcal{T}_i$.

2. Consider the case $\text{height}(\mu) \geq \text{space}(\mu)$. Then $\mathcal{T}_i$ represents the same embeddings as $\mathcal{T}_{i-1}$ and so condition 4 is trivially satisfied.

Now we show that permuting the children of P-nodes does not change the space around any node of $\mathcal{T}_i$. Recall Theorem 13, which states that all children of P-nodes have I shape. Take any two adjacent children of a P node μ and merge them, creating a new R-node child ν of the P-node. Then ν has I shape. Therefore it can be reflected. Further reflecting both children of ν, which is possible because they too have I shape,

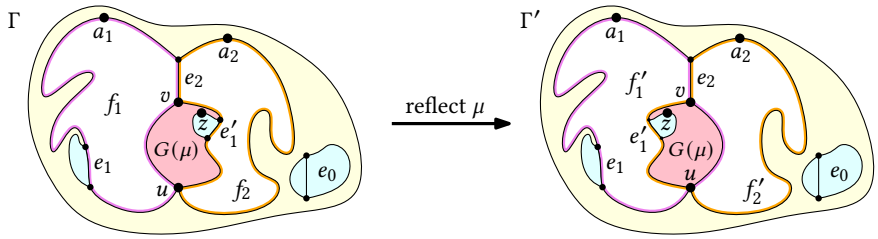


Figure 7.7: Proof of Lemma 55, Property 4 for the case height(μ) < space(μ). The edge e_0 is not incident to any μ-incident face, the edges e_1, e'_1 are incident to exactly one μ-incident face and the edge e_2 is incident to both μ-incident faces. The space around all nodes of $\mathcal{T}$ does not change when reflecting μ.

means that in the resulting embedding the two children are reversed. Note that any permutation can be realized by a number of exchanges of adjacent pairs, which shows that condition 4 remains satisfied when permuting the children of P-nodes. This shows that condition 4 is satisfied for $\mathcal{T}_i$.

As the final step, we prove that condition 5 is satisfied for $\mathcal{T}_i$. Recalling Theorem 14 and the equivalence of the skeleton-based and arc-based representations, we have that for every level-planar embedding Γ of G there exists a level-planar embedding Γ_p that is obtained from Γ by reordering the children of P-nodes such that it is $\Gamma_p = \Gamma_{\text{ref}}(\pi_1, \pi_2, \ldots, \pi_m)$ where it is $\alpha_j = (\lambda_j, \mu_j)$ and $\pi_j = \alpha_j$ or $\pi_j = \bar{\alpha}_j$ denotes whether the embedding of $G(\mu_i)$ should remain unchanged or be flipped, respectively. Now let $\pi_j = \alpha_j$ for $j > i$ as required by the invariant. We show that Γ_p is represented by $\mathcal{T}_i$, Theorem 14 then implies that Γ is represented by $\mathcal{T}_i$ as well.

In the base case $i = 0$ no arc is flipped, i.e., we have $\Gamma_p = \Gamma_{\text{ref}}$, which is the level-planar embedding of G represented by $\mathcal{T}_0$ by definition. In the inductive case $i > 0$, we distinguish two cases based on whether it is $\pi_i = \alpha_i$ or $\pi_i = \bar{\alpha}_i$. Define

$$\Gamma_p^1 = \Gamma_{\text{ref}}(\pi_1, \pi_2, \ldots, \pi_{i-1}, \alpha_i, \alpha_{i+1}, \ldots, \alpha_m) \quad \text{and}$$
$$\Gamma_p^2 = \Gamma_{\text{ref}}(\pi_1, \pi_2, \ldots, \pi_{i-1}, \neg\alpha_i, \alpha_{i+1}, \ldots, \alpha_m).$$

Observe that Γ_p^1 is represented by $\mathcal{T}_{i-1}$ by induction on condition 5. Then Γ_p^1 is also represented by $\mathcal{T}_i$, which shows the claim for $\Gamma_p = \Gamma_p^1$. Otherwise, it is $\Gamma_p = \Gamma_p^2$. Let Π denote the restriction of Γ_p^1 to $G(\mu_i)$. Then it is $\Gamma_p^1 = \Gamma_p^1 + \Pi$ and flipping α_i reflects Π, i.e., $\Gamma_p^2 = \Gamma_p^1 + \bar{\Pi}$. We now distinguish two cases based on whether α_i is labeled as flexible or rigid. If α_i is labeled as flexible, $\Gamma_p^2 = \Gamma_p$ is represented by $\mathcal{T}_i$. Otherwise, α_i is labeled as rigid. Recall that Γ_p^1 is represented by $\mathcal{T}_{i-1}$ and $\mathcal{T}_i$. Then condition 3 gives that $\Gamma_p = \Gamma_p^2 = \Gamma_p^1 + \bar{\Pi}$ is not level planar, a contradiction. □

The restricted decomposition tree $\mathcal{T}_m$ represents only level-planar embeddings by Property 1 of Lemma 55. Because no arc of $\mathcal{T}_m$ is unlabeled, it also follows that all level-planar embeddings of G are represented by $\mathcal{T}_m$. Contracting all arcs labeled as rigid in $\mathcal{T}_m$ gives the LP-tree for G, which concludes our proof of Theorem 15.

7.3.4 Construction in Linear Time

The algorithm described in Section 7.3 clearly has polynomial running time. In this section, we describe an implementation of it that has linear running time. Starting out, the preprocessing step where the apex t and the edge (s, t) is added to G is feasible in linear time. Next, the SPQR-tree $\mathcal{T}$ of this modified graph G can be computed in linear time [GM00, HT73]. Then, a level-planar embedding Γ of G is computed in linear time [DN88] and all skeletons of $\mathcal{T}$ are embedded accordingly.

For each node μ of $\mathcal{T}$ the height of $G(\mu)$ needs to be known. The heights for all nodes are computed bottom-up. Note that the height of an edge $e = (u, v)$ of G is $\ell(u)$. This means that the heights for all leaf Q-nodes can be easily determined. In general, to determine the height for a node μ of $\mathcal{T}$, proceed as follows. Assume the heights are known for all children. Let E_μ be the child virtual edges of $\text{skel}(\mu)$ and let $h(e)$ denote the height of $G(\nu)$ with $\text{corr}_\mu(e) = \nu$. Then the height of μ is $\max\{\{d(e) \mid e \in E_\mu\} \cup \{d(w) \mid w \in V(\mu)\}\}$. Thus, the running time spent to determine the height of μ when the heights of all its children is known is linear in the size of $\text{skel}(\mu)$. Because the sum of the sizes of all skeletons of $\mathcal{T}$ is linear in n, all heights can be computed in linear time.

The next step is to split P-nodes. Let μ be a P-node. One split at μ requires to find the child with the greatest height. Because Γ is a level-planar embedding, Lemma 52 gives that this is one of the outermost children. By inspecting the two outermost children of μ, the child ν with greatest height can be found, or it is found that all children of μ have I shape and μ does not need to be split. A P-node split is a constant-time operation. Because there are no more P-node splits than nodes in $\mathcal{T}$, all P-node splits are feasible in linear time.

The final step of the algorithm is to process all arcs. For this the space around each node needs to be known. The space around a node μ depends on the apices of the μ-incident faces in Γ. Fortunately, these can be easily computed bottom-up. Start by labeling every face f of Γ with its apex by walking around the cycle that bounds f. For every edge e of G the apices on both sides of e can then be looked up in Γ. So the incident apices are known for each Q-node of $\mathcal{T}$. Let μ be a node of $\mathcal{T}$ so that for each child ν of μ the apices of the ν-incident faces are known. Then the apices of the μ-incident faces can be determined from the child virtual edges of $\text{skel}(\mu)$ that share a face with the parent virtual edge of μ. The running time of this procedure is linear in the sum of sizes of all skeletons, i.e., linear in n. To process

the arcs, simply walk through $\mathcal{T}$ from the top down. Compute the space around each child node ν from the available apices of the ν-incident faces and compare it with the precomputed height of $G(\mu)$. Finally, contract all arcs marked as rigid, which again is feasible in overall linear time. This proves the running time claimed in Theorem 15.

7.4 Applications

We use the LP-tree to translate efficient algorithms for constrained planarity problems to the level-planar setting. First, we extend the partial planarity algorithm by Angelini et al. [Ang+15c] to solve partial level planarity for biconnected single-source level graphs. Second, we adapt this algorithm to solve constrained level planarity. In both cases we obtain a linear-time algorithm, improving upon the best previously known running time of $O(n^2)$, though that algorithm also works in the non-biconnected case [BR17]. Third, we translate the simultaneous planarity algorithm due to Angelini et al. [Ang+12] to the simultaneous level planarity problem when the shared graph is a biconnected single-source level graph. Previously, no polynomial-time algorithm was known for this problem.

7.4.1 Partial Level Planarity

Angelini et al. define partial planarity in terms of the cyclic orders of edges around vertices (the "edge-order definition") as follows. A partially embedded graph (PEG) is a triple $(G, H, \mathcal{H})$ that consists of a graph G and a subgraph H of G together with a planar embedding $\mathcal{H}$ of H. The task is to find an embedding $\mathcal{G}$ of G that extends $\mathcal{H}$ in the sense that any three edges e, f, g of H that are incident to a shared vertex v appear in the same order around v in $\mathcal{G}$ as in $\mathcal{H}$. The algorithm works by representing all planar embeddings of G as an SPQR-tree $\mathcal{T}$ and then determining whether there exists a planar embedding of G that extends the given partial embedding $\mathcal{H}$ as follows. Recall that e, f, g correspond to distinct Q-nodes μ_e, μ_f and μ_g in $\mathcal{T}$. There is exactly one node ν of $\mathcal{T}$ that lies on all paths connecting two of these Q-nodes. Furthermore, e, f, g belong to the expansion graphs of three distinct virtual edges $\hat{e}, \hat{f}, \hat{g}$ of skel(ν). The order of e, f and g in the planar embedding represented by $\mathcal{T}$ is determined by the order of $\hat{e}, \hat{f}, \hat{g}$ in skel(ν), i.e., by the embedding of skel(ν). Fixing the relative order of e, f, g therefore imposes certain constraints on the embedding of skel(μ). Namely, an R-node can be constrained to have exactly one of its two possible embeddings and the admissible permutations of the neighbors of a P-node can be constrained as a partial ordering. To model the embedding $\mathcal{H}$ consider for each vertex v of H each triple e, f, g of consecutive edges around v and fix their order as in $\mathcal{H}$. The algorithm collects these linearly many constraints and then checks whether they can

be satisfied simultaneously.

Define partial level planarity analogously, i.e., a *partially embedded level graph* is a triple $(G, H, \mathcal{H})$ of a level graph G, a subgraph H of G and a level-planar embedding $\mathcal{H}$ of H. Again the task is to find an embedding $\mathcal{G}$ of G that extends $\mathcal{H}$ in the sense that any three edges e, f, g of H that are incident to a shared vertex v appear in the same order around v in $\mathcal{G}$ as in $\mathcal{H}$. This definition of partial level planarity is distinct from but (due to Lemma 47 (⋆)) equivalent to the one given in [BR17], which is a special case of constrained level planarity as presented in the next section. LP-trees exhibit all relevant properties of SPQR-trees used by the partial planarity algorithm. Ordered edges e, f, g of G again correspond to distinct Q-nodes of the LP-tree $\mathcal{T}'$ for G. Again, there is a unique node ν of $\mathcal{T}'$ that has three virtual edges $\hat{e}, \hat{f}, \hat{g}$ that determine the order of e, f, g in the level-planar drawing represented by $\mathcal{T}'$. Finally, in LP-trees just like in SPQR-trees, R-nodes have exactly two possible embeddings and the virtual edges of P-nodes can be arbitrarily permuted. Using the LP-tree as a drop-in replacement for the SPQR-tree in the partial planarity algorithm due to Angelini et al. gives the following, improving upon the previously known best algorithm with $O(n^2)$ running time (although that algorithm also works for the non-biconnected case [BR17]).

Theorem 16. *Partial level planarity can be solved in linear running time for biconnected single-source level graphs.*

Angelini et al. extend their algorithm to the connected case [Ang+15c]. This requires significant additional effort and the use of another data structure, called the enriched block-cut tree, that manages the biconnected components of a graph in a tree. Some of the techniques described in this chapter, in particular our notion of demands, may be helpful in extending our algorithm to the connected single-source case. Consider a connected single-source graph G. All biconnected components of G have a single source and the LP-tree can be used to represent their level-planar embeddings. However, a vertex v of some biconnected component H of G may be a cutvertex in G and can dominate vertices that do not belong to H. Depending on the space around v and the levels on which these vertices lie this may restrict the admissible level-planar embeddings of H. Let $X(v)$ denote the set of vertices dominated by v that do not belong to H. Set the demand of v to $d(v) = d(X(v))$. Computing the LP-tree with these demands ensures that there is enough space around each cutvertex v to embed all components connected at v. The remaining choices are into which faces of H incident to v such components can be embedded and possibly nesting biconnected components. These choices are largely independent for different components and only depend on the available space in each incident face. This information is known from the LP-tree computation. In this way it may be possible to extend the steps for handling non-biconnected graphs due to Angelini et al. to the level planar setting.

7.4.2 Constrained Level Planarity

A *constrained level graph* (CLG) is a tuple $(G, \{\prec'_1, \prec'_2, \ldots, \prec'_k\})$ that consists of a k-level graph G and partial orders $\prec'_i$ of V_i for $i = 1, 2, \ldots, k$ (the "vertex-order definition") [BR17]. The task is to find a drawing of G, i.e., total orders $\prec_i$ of V_i that extend $\prec'_i$ in the sense that for any two vertices $u, v \in V_i$ with $u \prec'_i v$ it is $u \prec_i v$.

Theorem 17. *Constrained level planarity can be solved in linear running time for biconnected single-source level graphs.*

Proof. Translate the given vertex-order constraints into edge-order constraints. This translation is justified by Lemma 47. We now show that all vertex-order constraints can be translated in linear time. For any pair u, v with $u \prec'_i v$ we start by finding a vertex w so that there are disjoint paths p_u and p_v from w to u and v. This can be achieved by using the algorithm of Harel and Tarjan on a depth-first-search tree $\mathcal{D}$ of G [HT84] in linear time. Mark w with the pair u, v for the next step. Then, we find the edges e and f of p_u and p_v incident to w, respectively. To this end, we proceed similarly to a technique described by Bläsius et al. [BKR18]. At the beginning, every vertex of G belongs to its own singleton set. Proceed to process the vertices of G bottom-up in $\mathcal{D}$, i.e., starting from the vertices on the greatest level. When encountering a vertex w marked with a pair u, v, find the representatives of u and v, denoted by u' and v', respectively. Observe that it is $e = (w, u')$ and $f = (w, v')$, and that both e and f are tree edges of $\mathcal{D}$. Then unify the sets of all of its direct descendants in $\mathcal{D}$ and let w be the representative of the resulting union. Because all union operations are known in advance we can use the linear-time union-find algorithm of Gabow and Tarjan [GT85]. Finally, pick some incoming edge around w as g, or the edge (s, t) if $w = s$. In this way, we translate the constraint of the form $u \prec'_i v$ to a constraint on the order of the edges e, f and g around w. Apply this translation for each constraint in the partial orders $\prec'_i$.

In a similar fashion we can find the node ν of the LP-tree $\mathcal{T}$ and the three virtual edges $\hat{e}, \hat{f}$ and $\hat{g}$ of skel(ν) so that the relative position of $\hat{e}, \hat{f}$ and $\hat{g}$ in the embedding of skel(ν) determines the relative position of e, f and g in the embedding represented by $\mathcal{T}$. We can the use a similar technique as the one described for partial level planarity. □

7.4.3 Simultaneous Level Planarity

We translate the simultaneous planarity algorithm of Angelini et al. [Ang+12] to solve simultaneous level planarity for biconnected single-source graphs. They define simultaneous planarity as follows. Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two graphs with the same vertices. The *inclusive* edges $E_1 \cap E_2$ together with V make up the

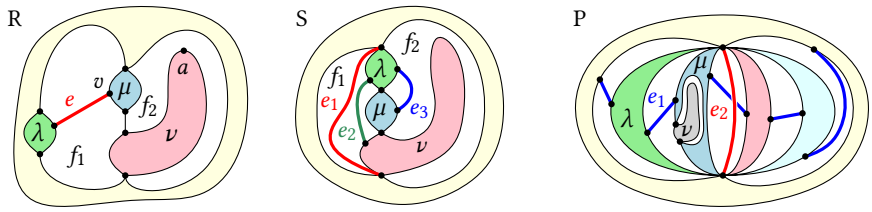


Figure 7.8: In the R-node, e fixes the relative embeddings of $G(\lambda)$ and $G(\mu)$. In the level-planar setting, e also fixes the embedding of $G(\nu)$. In the S-node, e_2 and e_3 fix the relative embeddings of $G(\lambda), G(\nu)$ and $G(\lambda), G(\mu)$, respectively. In the level-planar setting, e_1 also fixes the embedding of $G(\nu)$. In the P-node, e_1 fixes the relative embeddings of $G(\lambda)$ and $G(\mu)$. In the level-planar setting, e_1 also fixes the embedding of $G(\nu)$.

intersection graph $G_{1\cap 2}$, or simply G for short. All other edges are *exclusive*. The graphs G_1 and G_2 admit *simultaneous* embeddings $\mathcal{E}_1, \mathcal{E}_2$ if the relative order of any three distinct inclusive edges e, f and g with a shared endpoint is identical in $\mathcal{E}_1$ and $\mathcal{E}_2$. The algorithm of Angelini et al. works by building the SPQR-tree for the shared graph G and then expressing the constraints imposed on G by the exclusive edges as a 2-SAT instance S that is satisfiable iff G_1 and G_2 admit a simultaneous embedding. We give a very brief overview of the 2-SAT constraints in the planar setting. In an R-node, an exclusive edge e has to be embedded into a unique face. This potentially restricts the embedding of the expansion graphs $G(\lambda), G(\mu)$ that contain the endpoints of e, i.e., the embedding of $G(\lambda)$ and $G(\mu)$ is fixed with respect to the embedding of the R-node. Add a variable x_μ to S for every node of $\mathcal{T}$ with the semantics that x_μ is true if skel(μ) has its reference embedding Γ_μ, and false if the embedding of skel(μ) is the reflection of Γ_μ. The restriction imposed by e on $G(\lambda)$ and $G(\mu)$ can then be modeled as a 2-SAT constraint on the variables x_λ and x_μ. For example, in the R-node shown in Fig. 7.8 on the left, the internal edge e must be embedded into face f_1, which fixes the relative embeddings of $G(\lambda)$ and $G(\mu)$. In an S-node, an exclusive edge e may be embedded into one of the two candidate faces f_1, f_2 around the node. The edge e can conflict with another exclusive edge e' of the S-node, meaning that e and e' cannot be embedded in the same face. This is modeled by introducing for every exclusive edge e and candidate face f the variable x_e^f with the semantics that x_e^f is true iff e is embedded into f. The previously mentioned conflict can then be resolved by adding the constraints $x_e^{f_1} \vee x_e^{f_2}$, $x_{e'}^{f_1} \vee x_{e'}^{f_2}$ and $x_e^{f_1} \neq x_{e'}^{f_1}$ to S. Additionally, an exclusive edge e whose endpoints lie in different expansion graphs can restrict their respective embeddings. For example, in the S-node shown in Fig. 7.8

in the middle, the edges e_2 and e_3 may not be embedded into the same face. And e_2 and e_3 fix the embeddings of $G(\lambda)$ and $G(\nu)$ and of $G(\lambda)$ and $G(\mu)$, respectively. This would be modeled as $x_\lambda = x_\nu$ and $x_\lambda = x_\mu$ in S. In a P-node, an exclusive edge can restrict the embeddings of expansion graphs just like in R-nodes. Additionally, exclusive edges between the poles of a P-node can always be embedded unless all virtual edges are forced to be adjacent by internal edges. For example, in the P-node shown in Fig. 7.8 on the right, e_1 fixes the relative embeddings of $G(\lambda)$ and $G(\mu)$. And e_2 can be embedded if and only if one of the blue edges does not exist.

Adapt the algorithm to the level-planar setting. First, replace the SPQR-tree with the LP-tree $\mathcal{T}$. The satisfying truth assignments of S then correspond to simultaneous planar embeddings $\mathcal{E}_1, \mathcal{E}_2$ of G_1, G_2, so that their shared embedding $\mathcal{E}$ of G is level planar. However, due to the presence of exclusive edges, $\mathcal{E}_1$ and $\mathcal{E}_2$ are not necessarily level planar. To make sure that $\mathcal{E}_1$ and $\mathcal{E}_2$ are level planar, we add more constraints to S. Consider adding an exclusive edge e into a face f. This splits f into two faces f', f''. The apex of at least one face, say f'', remains unchanged. As a consequence, the space around any virtual edge incident to f'' remains unchanged as well. But the apex of f' can change, namely, the apex of f' is an endpoint of e. Then the space around the virtual edges incident to f' can decrease. This reduces the space around the virtual edge associated with ν. In the same way as described in Section 7.3.2, this restricts some arcs in $\mathcal{T}$. This can be described as an implication on the variables x_e^f and x_ν. For an example, see Fig. 7.8. In the R-node, adding the edge e with endpoint v into f_1 creates a new face f_1' with apex v. This forces $G(\nu)$ to be embedded so that its apex a is embedded into face f_2. Similarly, in the S-node and in the P-node, adding the edge e_1 restricts $G(\nu)$. We collect all these additional implications of embedding e into f and add them to the 2-SAT instance S. Each exclusive edge leads to a constant number of 2-SAT implications. To find each such implication $O(n)$ time is needed in the worst case. Because there are at most $O(n)$ exclusive edges this gives quadratic running time overall. Clearly, all implications must be satisfied for $\mathcal{E}_1$ and $\mathcal{E}_2$ to be level planar. On the other hand, suppose that one of $\mathcal{E}_1$ or $\mathcal{E}_2$, say $\mathcal{E}_1$, is not level planar. Because the restriction of $\mathcal{E}_1$ to G is level planar due to the LP-tree and planar due to the algorithm by Angelini et al., there must be a crossing involving an exclusive edge e of G_1. This contradicts the fact that we have respected all necessary implications of embedding e. We obtain Theorem 18.

Theorem 18. *Simultaneous level planarity can be solved in quadratic time for two graphs whose intersection is a biconnected single-source level graph.*

In the non-biconnected setting Angelini et al. solve the case when the intersection graph is a star. Haeupler et al. describe an algorithm for simultaneous planarity that does not use SPQR-trees, but they also require biconnectivity [HJL13]. The complexity of the general (connected) case remains open.

7.5 Conclusion

The majority of constrained embedding algorithms for planar graphs rely on two features of the SPQR-tree: they are decomposition trees and the embedding choices consist of arbitrarily permuting parallel edges between two poles or choosing the flip of of a skeleton whose embedding is unique up to reflection. We have developed the LP-tree, an SPQR-tree-like embedding representation that has both of these features. SPQR-tree-based algorithms can then usually be executed on LP-trees without any modification. The necessity for mostly minor modifications only stems from the fact that in many cases the level-planar version of a problem imposes additional restrictions on the embedding compared to the original planar version. Our LP-tree thus allows to leverage a large body of literature on constrained embedding problems and to transfer it to the level-planar setting. In particular, we have used it to obtain linear-time algorithms for partial and constrained level planarity in the biconnected case, which improves upon the previous best known running time of $O(n^2)$. Moreover, we have presented an efficient algorithm for the simultaneous level planarity problem. Previously, no polynomial-time algorithm was known for this problem. Finally, we have argued that an SPQR-tree-like embedding representation for level-planar graphs with multiple sources does not substantially help in solving the partial and constrained level planarity problems, is not efficiently computable, or does not exist.

8 An SPQR-Tree-Like Embedding Representation for Upward Planarity

The SPQR-tree is a data structure that compactly represents all planar embeddings of a biconnected planar graph. It plays a key role in constrained planarity testing.

We develop a similar data structure, called the UP-tree, that compactly represents all upward planar embeddings of a biconnected single-source directed graph. We demonstrate the usefulness of the UP-tree by solving the upward planar embedding extension problem for biconnected single-source directed graphs.

This chapter is based on joint work with Markus Himmel and Ignaz Rutter [BHR19].

8.1 Introduction

A natural extension of planarity to directed graphs (digraphs) is upward planarity. Whereas undirected graphs can be tested for planarity in linear time, upward planarity testing is NP-complete in general, though there are efficient algorithms for graphs with a single source [HL96, BDMT98] and graphs with a fixed embedding [BDLM94]. In the special case of *st-graphs*, i.e., graphs with a single source s and a single sink t with s and t on the same face, every planar embedding is also upward planar [Pla76], and hence upward planarity and planarity are equivalent.

A related but different planarity notion for digraphs is level planarity. Level planarity can be tested in linear time [JL02] by a quite involved algorithm, or in quadratic time by several simpler algorithms [BRS18, Ran+01, FPSŠ13].

In a constrained embedding problem, one seeks a planar embedding of a given graph that satisfies additional constraints. Typical examples are simultaneous embed-

dings with fixed edges [BKR13], cluster planarity [FCE95], constraints on the face sizes [DJKR14, DR18], optimizing the depth of the embedding [ADP11] and optimizing the bends in an orthogonal drawing [BLR16, BRW16, DLP18]. One of the most prominent examples of the last years is the *partial drawing extension* problem, which asks whether a given drawing of a subgraph can be extended to a planar drawing of the whole graph. The *partial embedding extension* problem is strongly related, here the input is a planar embedding of a subgraph and the question is whether it can be extended to a planar embedding of the whole graph. For undirected planar graphs the two problems are equivalent and can be solved in linear time [Ang+15c, JKR13].

One of the key tools for all of these applications is the SPQR-tree, which compactly represents all planar embeddings of a biconnected planar graph G and breaks down the complicated task of choosing a planar embedding of G into simpler independent embedding choices of its triconnected components [Mac37, Tut66, HT73, DT89, DT90, DT96]. In fact, these embeddings are either uniquely determined up to reversal, or they consist in arbitrarily choosing a permutation of parallel edges between two pole vertices. The common approach for attacking the above-mentioned constrained embedding problems is to project the constraints on the global embedding to local constraints on the skeleton embeddings that can then be satisfied by consistent local choices. While the implementation details are often highly technical and non-trivial, the approach has proven to be extremely successful.

In comparison, relatively little is known about constrained planarity problems for planarity notions of digraphs. Brückner and Rutter [BR17] study the problem of extending a given partial drawing of a level graph and Da Lozzo et al. [DDF20] study the same question for upward planarity. In general, extending a given partial upward planar drawing requires to determine an upward planar embedding that (i) extends the embedding of the partial drawing, and (ii) admits a drawing that extends the given drawing. Here step (i) requires solving the embedding extension problem but with additional constraints that ensure that a drawing extension is feasible. It is worth noting that for upward planarity the embedding extension problem and the drawing extension problem are distinct; Da Lozzo et al. show that, generally, even if an upward planar embedding of the whole graph is given, it is NP-complete to decide whether it can be drawn such that it extends a given partial drawing [DDF20, Theorem 2]. On the positive side, they present tractability results for directed paths and cycles with a given upward planar embedding, and for st-graphs. The restriction to st-graphs allows a relatively simple characterization of the upward planar embeddings that extend the given partial drawings [DDF20, Lemma 6], which yields an $O(n \log n)$-time algorithm for step (ii). For step (i), Da Lozzo et al. exploit the fact that for st-graphs, the choice of an upward planar embedding is equivalent to choosing a planar embedding, and hence the SPQR-tree allows to efficiently search for an upward planar embedding satisfying the additional constraints required by condition (ii).

In this chapter, we seek to generalize the approach of Da Lozzo et al. to biconnected single-source graphs. The key difficulty in this case is that neither do we have access to all the upward planar embeddings of such graphs, nor it is known what the necessary and sufficient conditions are for an upward planar embedding to admit a drawing that extends a given subdrawing.

Contribution and Outline. We construct a novel SPQR-tree-like embedding representation, called the UP-tree, that represents exactly the upward planar embeddings of a biconnected single-source graph in which some prescribed edge appears leftmost around the source. As in SPQR-trees, the embedding choices in the UP-tree are broken down into independent embedding choices of skeleton graphs that are either unique up to reversal or allow to arbitrarily permute parallel edges between two poles. As such, UP-trees can take the role of SPQR-trees for constrained embedding problems in upward planarity, making them a powerful tool with a broad range of applications. We demonstrate this by giving an quadratic-time algorithm for the upward planar embedding extension problem for biconnected single-source graphs where the partial embedding is connected.

We review the results on decomposing upward planar single-source digraphs due to Hutton and Lubiw [HL96] in Section 8.2. We proceed to extend this idea from a single decomposition to decomposition trees. In Section 8.3, we define the UP-tree and in Section 8.4 we use it to solve the partial upward embedding extension problem.

8.2 Decomposition Trees and Upward Planar Embeddings

Recall that for biconnected graphs we can decompose any planar embedding into planar embeddings of the skeletons of a decomposition tree; and symmetrically, we can compose a planar embedding of the whole graph from planar embeddings of the skeletons. In this section we find a similar relationship between upward planar embeddings of a biconnected single-source digraph G and upward planar embeddings of the skeletons of a suitably-defined decomposition tree of G.

8.2.1 Decompositions and Upward Planar Embeddings

In this section we review the decomposition result of Hutton and Lubiw and formulate the interface between their result and our results.

Let G be a biconnected single-source digraph together with an upward planar embedding $\mathcal{E}$. Further, let $e^\star$ denote the edge around the source of G that is leftmost in $\mathcal{E}$. Note that $e^\star$ is incident to the outer face. It will play a similar role as the

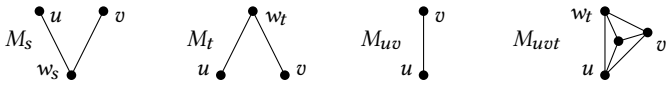


Figure 8.1: The four markers used by Hutton and Lubiw. The markers are digraphs; in the figure, all edges are directed upward.

edge $e^\star$ in rooted decomposition trees as described in Chapter 2. Now let H_1, H_2 be two subgraphs of G with (i) $H_1 \cup H_2 = G$, (ii) $H_1 \cap H_2 = \{u, v\}$, (iii) $e^\star \in H_1$ and (iv) $H_1 \setminus \{u, v\}$ or $H_2 \setminus \{u, v\}$ is connected.

Hutton and Lubiw construct two graphs H_1' and H_2' from H_1 and H_2 by including one of the *markers* shown in Figure 8.1. Markers are simple digraphs with two vertices u, v that connect the marker to the remaining graph. The marker in H_1' is designed to represent H_2 and the marker in H_2' is designed to represent H_1. If there exists a directed path from u to v we say that u *dominates* v and write $u < v$ for short. Otherwise u and v are *incomparable*. The vertex v is a *source* if it has no incoming edges in G, a *sink* if it has no outgoing edges in G and an *internal* vertex if it has both incoming and outgoing edges in G. Markers are determined based on whether $u < v$ and whether v is a source, sink or internal vertex in H_1 and H_2: If u and v are incomparable in G, set $H_1' = H_1 \cup M_t$ and $H_2' = H_2 \cup M_s$. Otherwise, assume $u < v$. Define H_1' as follows. If v is a source in H_2 set $H_1' = H_1 \cup M_t$. If v is a sink in H_2 set $H_1' = H_1 \cup M_{uv}$. Otherwise v is an internal vertex in H_2 and we set $H_1' = H_1 \cup M_{uvt}$. Define H_2' as follows. If v is a source in H_1 set $H_2' = H_2 \cup M_t$, otherwise set $H_2' = H_2 \cup M_{uv}$. See Figure 8.2 for example decompositions. Such a decomposition can be reversed by *composing* H_1' and H_2' to obtain G. To this end, remove the markers from H_1' and H_2' to obtain H_1 and H_2 and take their union to obtain G.

Recall that decomposition trees of planar graphs allow for (de-)composition of planar embeddings. Hutton and Lubiw provide a similar property for upward planar embeddings of G, H_1' and H_2'. *Decompose* an upward planar embedding of G into upward planar embeddings of H_1', H_2' as follows. To obtain the embedding of H_1', contract H_2 into the marker M in H_1' that corresponds to H_2'. The embedding of H_2' is obtained symmetrically. To *compose* upward planar embeddings of H_1', H_2', contract the markers into a single edge and then proceed with the composition for planar embeddings. Hutton and Lubiw state the following.

Theorem 19 (implicit in [HL96])**.** *Let $\mathcal{E}$ be an upward planar embedding of G with $e^\star$ as the leftmost edge around s. Then $\mathcal{E}$ induces upward planar embeddings $\mathcal{F}_1, \mathcal{F}_2$ of H_1', H_2', respectively with the following properties. In $\mathcal{F}_1$, $e^\star$ is the leftmost edge around the source of H_1'. The edges of the marker in H_2' are incident to the outer face*

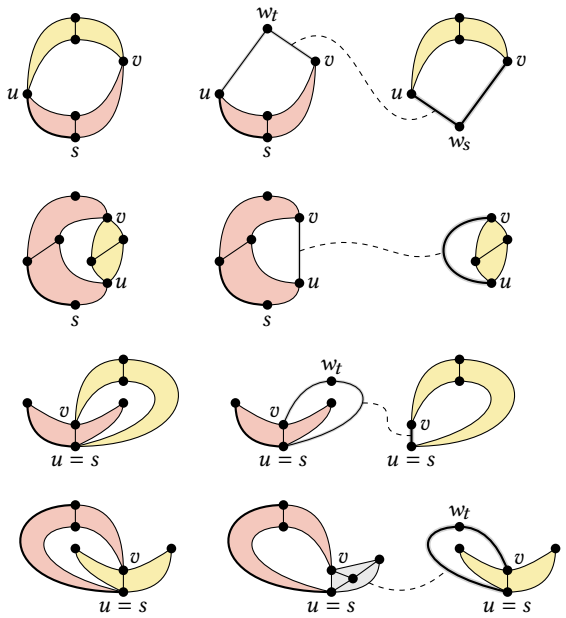


Figure 8.2: Example decompositions of upward planar graphs. The left column shows four examples of G with H_1 shaded in green and H_2 shaded in red. The right column shows the decomposition of G into H_1' and H_2', with the dashed edge connecting corresponding markers. The edge $e^\star$ is drawn more thickly.

in $\mathcal{F}_2$.

Conversely, if $\mathcal{F}_1$ and $\mathcal{F}_2$ are upward planar embeddings of H_1' and H_2' such that $e^\star$ is the leftmost edge around the source of H_1' in $\mathcal{F}_1$ and the edges of the marker in H_2' are incident to the outer face of $\mathcal{F}_2$, then the composition of these embeddings is upward planar.

Hutton and Lubiw do not explicitly state Theorem 19. Because they are only interested in testing upward planarity, they state that G is upward planar if and only if H_1' and H_2' are upward planar subject to certain restrictions. However, their proof shows that every upward planar embedding of G can be decomposed into upward planar embeddings $\mathcal{F}_1, \mathcal{F}_2$ of H_1', H_2', respectively. Theorems 6.5, 6.7, 6.8 and 6.9 by Hutton and Lubiw [HL96] distinguish cases based on which markers are used. For the forward direction and markers M_s, M_t and M_{uv} observe that because H_1 contains a subgraph that is a subdivision of the marker in H_2' that represents H_1 (and the same holds for H_2 and the marker in H_1' that represents H_2). The marker M_{uvt} requires

some extra treatment for the central vertex. The fact that $e^\star$ is the leftmost edge in $\mathcal{E}$ implies that (i) $e^\star$ is the leftmost edge of H_1' in $\mathcal{F}_1$, and (ii) the marker in H_2' is incident to the outer face in $\mathcal{F}_2$.

For the reverse direction, Hutton and Lubiw show that the composition $\mathcal{E}$ of upward planar embeddings $\mathcal{F}_1, \mathcal{F}_2$ of H_1', H_2', respectively, where the marker in H_2' is incident to the outer face in $\mathcal{F}_2$, yields an upward planar embedding of G. If $e^\star$ is the leftmost edge around s in $\mathcal{F}_1$, then it is also leftmost in $\mathcal{E}$. Our restrictions involving $e^\star$ imply that during the composition of $\mathcal{E}$ the embedding $\mathcal{F}_1$ is inserted into the outer face of $\mathcal{F}_2$. Hutton and Lubiw do not have a similar restriction and treat $\mathcal{F}_1$ and $\mathcal{F}_2$ more symmetrically. Due to this, if $u = s$, they have to carefully choose which component has to be inserted into the outer face of the other component. In our case, these roles are completely fixed by the choice of $e^\star$ and then coincide with the treatment of Hutton and Lubiw.

8.2.2 Decomposition Trees and Upward Planar Embeddings

The approach of Hutton and Lubiw is to decompose a single-source digraph G into two smaller single-source digraphs G_1, G_2 and use Theorem 19 to translate upward-planarity testing of G to upward-planarity testing of two smaller instances H_1', H_2'. Observe that the markers and the replacement rules are defined so that both H_1' and H_2' are single-source digraphs. This means that H_1' and H_2' can be recursively decomposed. Note that in the context of connectivity markers are treated simply as edges, i.e., markers are not decomposed further. When a graph cannot be further decomposed it is triconnected and therefore has a unique planar embedding which can be tested for upward planarity in linear time using the algorithm of Bertolazzi et al. [BDMT98]. In the context of upward planarity testing the full marker graph is considered. Upward planar embeddings of H_1' and H_2' can then be composed to an upward planar embedding of G. In the context of embedding composition markers are again treated simply as edges. In particular, it does not matter whether the clockwise order of the edges incident to u in M_{uvt} is $(u, v), (u, x), (u, w_t)$ or $(u, w_t), (u, x), (u, v)$.

We use a different approach. Instead of testing H_1' and H_2' for upward-planarity separately, we manage them as the skeletons of two nodes in a decomposition tree $\mathcal{T}$. Note that Theorem 19 requires $H_1 \setminus \{u, v\}$ or $H_2 \setminus \{u, v\}$ to be connected. We call such a decomposition *maximal*. We then decompose these skeletons further, which grows the decomposition tree. A *maximal-decomposition tree* is a decomposition tree obtained by performing only maximal decompositions. A *configuration* equips the skeleton of each node in the tree with an upward planar embedding. In this embedding, $e^\star$ or the marker that represents the component that contains $e^\star$ must be incident to the outer face and leftmost around the source of the skeleton. See Figure 8.6 (c) for an example of a maximal-decomposition tree. Applying Theorem 19

at each decomposition step gives the following.

Theorem 20. *Let G be a biconnected graph with a single source s, let $e^\star$ be an edge of G incident to s and let $\mathcal{T}$ denote a maximal-decomposition tree of G. Then the upward-planar embeddings of G in which $e^\star$ is the leftmost edge around s correspond bijectively to the configurations of $\mathcal{T}$.*

We could use Theorem 20 directly to represent all upward planar embeddings of a graph. But we also show that decomposition trees are uniquely defined by the decompositions that are executed, but not by the order of these decompositions. This means that just like we can talk about *the* SPQR-tree decomposition for a graph we will be able to talk about *the* UP-tree decomposition. The benefit of this is that we can use a UP-tree decomposition to determine that some constrained representation problem has no solution without having to consider other conceivable UP-tree decompositions.

To prove uniqueness, we show that the order of the decompositions is irrelevant. We then apply the decompositions as defined by the SPQR-tree decomposition, which is unique, and obtain the unique UP-tree decomposition. To this end, we prove that the marker replacement rules do not depend on the order of the decompositions. Recall that the marker replacement rules depend on vertex dominance and the local neighborhood of certain vertices. We prove Lemma 56, which states that decompositions preserve vertex dominance and Lemma 57, which states that decompositions preserve the local neighborhood of certain vertices.

Lemma 56. *Let G be a biconnected single-source digraph and let H_1', H_2' denote the result of decomposing along a cutpair $\{u, v\}$ of G. For $i = 1, 2$ and any two vertices x, y in H_i' it is $x < y$ in H_i' if and only if $x < y$ in G.*

Proof. Recall that the edges of G are partitioned across H_1 and H_2 and that u is a source in H_2.

1. u and v are incomparable in G. Let x, y be two vertices in H_1' with $x < y$ in G. Because u and v are incomparable in G the directed path from x to y in G cannot use an edge from H_2, i.e., it consists entirely of edges in H_1, i.e., it is $x < y$ in H_1'. Now let x, y be two vertices in H_1' with $x < y$ in H_1'. Because M_t contains no directed path between u and v this path consists entirely of edges in H_1. Then this same path exists in G, i.e., it is $x < y$ in G.

   A symmetric argument using the fact that M_s contains no directed path between u and v works for the case when x, y are vertices in H_2'.

2. $u < v$ in G. Let x, y be two vertices in H_1'. Follow the construction rules for H_1'.

a) v is a source in H_2. Recall that $H_1' = H_1 \cup M_t$. Assume $x < y$ in H_1'. Because M_t has no directed path between u and v any directed path from x to y in H_1' cannot use edges from M_t. This means that such a path consists entirely of edges in H_1, i.e., it also exists in G. Hence, it is $x < y$ in G. Now assume $x < y$ in G. Because v is a source in H_2 there exists no directed path from u to v in H_2. From $u < v$ in G it follows that there exists a directed path from u to v in H_1 and therefore in H_1'. Hence, it is $x < y$ in H_1'.

b) v is a sink in H_2. Because G is a single-source graph and it is $u < v$ there exists a directed path p from u to v in H_2. Recall that it is $H_1' = H_1 \cup M_{uv}$. Any directed path from x to y in H_1' either uses only edges in H_1 in which case the same path exists in G, or it uses the edge M_{uv} in which case it can be modified to a path that uses p in G. The same argument works in the reverse direction.

c) v is an internal vertex in H_2. Recall that $H_1' = H_1 \cup M_{uvt}$ and apply the same argument as in the previous case.

Now let x, y be two vertices in H_2' and follow the construction rules for H_2'.

a) v is a source in H_1. Recall that $H_2' = H_2 \cup M_t$ and follow the symmetric case $H_1' = H_1 \cup M_t$.

b) v is not a source in H_1. Recall that $H_2' = H_2 \cup M_{uv}$ and follow the symmetric cases $H_1' = H_1 \cup M_{uv}$ or $H_1' = H_1 \cup M_{uvt}$.

□

Lemma 57. *Let G be a biconnected single-source digraph and let H_1', H_2' denote the result of decomposing along a cutpair $\{u, v\}$ of G. For $i = 1, 2$ let $\{x, y\}$ denote a cutpair of H_i' that separates G into D_1 and D_2, and H_i' into F_1 and F_2. Then y is a source in D_1 if and only if y is a source in F_1. Moreover, y is a source, sink or internal vertex in D_2 if and only if y is a source, sink or internal vertex in F_2, respectively.*

Proof. The separation of G into D_1 and D_2 induces a separation of one of H_1' or H_2' into F_1 and F_2. Note that D_1, D_2 do not contain markers corresponding to each other, and neither do F_1 and F_2. The only markers present are the ones corresponding to H_1' and H_2'.

1. The induced separation lies in H_1'. Distinguish whether the marker M that corresponds to H_2' lies in F_2 or in F_1.

   a) F_2 contains the marker M corresponding to H_2'. See Figure 8.3 for an sketch of the scenario. Then $D_1 = F_1$, so y is a source in D_1 if and only

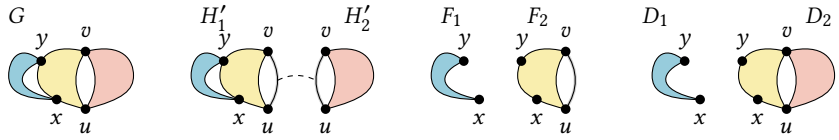


Figure 8.3: Case 1. (a) of the proof of Lemma 57. The graph G is decomposed into H_1', H_2' along $\{u, v\}$, the separation of G into D_1, D_2 along $\{x, y\}$ induces a separation of H_1' into F_1, F_2.

if y is a source in F_1. Moreover, D_2 is obtained from F_2 by replacing M with H_2'. Equivalently, F_2 is obtained from D_2 by replacing H_2 with M. We have to show that y is a source, sink or internal vertex in D_2 if and only if y is a source, sink or internal vertex in F_2, respectively.

If $M = M_t$ vertex $y = v$ is a source in H_2 and so exchanging $M = M_t$ with H_2 exchanges one outgoing edge with a non-empty set of outgoing edges. If $M = M_{uv}$ vertex $y = v$ is a sink in H_2 and so exchanging $M = M_{uv}$ with H_2 exchanges one incoming edge with a non-empty set of incoming edges. Finally, if $M = M_{uvt}$ vertex $y = v$ is an internal vertex in H_2 and so exchanging $M = M_{uvt}$ with H_2 exchanges three outgoing and three incoming edges with non-empty sets of outgoing and incoming edges. Now consider the case $y = u$. By definition u is a source in H_2 and in all candidate markers. Thus, when going from F_2 to D_2 and vice versa only edges of the same kind are exchanged, which shows the claim.

b) F_1 contains the marker M corresponding to H_2'. Then D_1 is obtained from F_1 by replacing M with H_2. Equivalently, F_1 is obtained from D_1 by replacing H_2 with M. We have to show that y is a source in D_1 if and only if y is a source in F_1. This works just like the previous case with the names of D_1 and D_2 and of F_1 and F_2 reversed, where we show the stronger statement that y is a source, sink or internal vertex in D_1 if and only if y is a source, sink or internal vertex in F_1, respectively.

Moreover, it is $D_2 = F_2$, so y is a source, sink or internal vertex in D_2 if and only if y is a source, sink or internal vertex in F_2, respectively.

2. The induced separation lies in H_2'. Because D_1 contains $e^\star$ the component F_1 contains the parent marker M corresponding to H_1'. Then D_1 is obtained from F_1 by replacing M with H_1. Equivalently, F_1 is obtained from D_1 by replacing H_1 with M. We have to show that y is a source in D_1 if and only if y is a source in F_1.

Note that if M is not adjacent to y the neighborhood of y remains unchanged and nothing further needs to be shown. Otherwise M is adjacent to y, i.e., it is $y = u$ or $y = v$. Consider the case $y = v$. If $v = y$ is a source in H_1 it is $M = M_t$ and exchanging $M = M_t$ with H_1 exchanges one outgoing edge with a non-empty set of outgoing edges. Otherwise $v = y$ is not a source in H_1 and it is $M = M_{uv}$. From $M = M_{uv}$ it follows that $v = y$ is not a source in F_1. The case $y = u$ cannot occur because u is the source of H_2'.

Moreover, it is $F_2 = D_2$, so y is a source, sink or internal vertex in D_2 if and only if y is a source, sink or internal vertex in F_2, respectively.

□

Lemmas 56 and 57 immediately give the following.

Lemma 58. *Let G be a biconnected graph with a single source s, let $e^\star$ be an edge of G incident to s and let $\mathcal{T}$ denote a decomposition tree of G. Then $\mathcal{T}$ relative to $e^\star$ is uniquely defined by the decompositions regardless of their order.*

A configuration of $\mathcal{T}$ can be computed as follows. Recall that all skeletons are single-source digraphs. We may therefore run the algorithm due to Bertolazzi et al. [BDMT98] on each skeleton. Observe that in a configuration of $\mathcal{T}$ relative to $e^\star$ the skeleton of each node μ of $\mathcal{T}$ must be embedded so that $e^\star$ or the marker that corresponds to the component that contains $e^\star$ must appear leftmost around the source of $\mathrm{skel}(\mu)$. We can enforce this by rooting the decomposition tree constructed by the algorithm of Bertolazzi et al. at the Q-node corresponding to $e^\star$ or an edge of the marker that corresponds to the component that contains $e^\star$.

8.3 UP-Trees

We are ready to construct the UP-tree, a maximal-decomposition tree designed to mimic the SPQR-tree. Let G be a biconnected directed single-source graph. The base of the construction is the decomposition tree obtained by performing the same set of decompositions as in the construction of the SPQR-tree decomposition of the underlying undirected graph of G. We then perform two additional steps. The first step is to split P-nodes into chains of smaller nodes. The second step is to determine whether skeletons of R-nodes can be reversed and to contract some arcs of the decomposition tree. In both steps, we reason about upward planarity of fixed embeddings with the following lemma due to Bertolazzi et al. [BDMT98].

Let G be a biconnected single-source graph together with a planar embedding. The *face-sink graph* F of G has the vertices and faces of G as its vertices. It contains

an undirected edge $\{f, v\}$ if f is a face of G and v is a vertex of G that is incident to f and both edges incident to v and f are directed towards v. The following lemma implies a linear-time algorithm that tests an embedding for upward planarity and outputs for each face whether it can be the outer face.

Lemma 59 ([BDMT98, Theorem 1])**.** *Let G be an embedded planar single-source digraph and let h be a face of G. Graph G has an upward planar drawing that preserves the embedding with outer face h if and only if all of the following is true: (i) the face-sink graph F of G is a forest (ii) there is exactly one tree T of F that contains no internal vertex of G, whereas each other tree contains exactly one internal vertex; (iii) h is in tree T; and (iv) the source of G is in the boundary of h.*

8.3.1 P-Node Splits

In SPQR-trees, the edges of P-nodes may be arbitrarily permuted. In decomposition trees for upward planar graphs there are stricter rules for the ordering of the markers in P-nodes. In this section, we determine these rules and find that by breaking up the P-nodes into chains of smaller nodes we obtain a decomposition tree for upward planarity whose P-nodes exhibit the same behavior as in SPQR-trees, i.e., their edges may be arbitrarily permuted. The idea is that certain kinds of markers must appear consecutively.

First, we argue that all M_{uv} markers must appear consecutively. To see this, note that if M_s appears between two M_{uv} markers then the outer face is not incident to the source of the skeleton, which is vertex w_s of M_s. If a marker M with $M = M_t$ or $M = M_{uvt}$ appears between two M_{uv} markers then the face incident to w_t of M and a marker M_{uv} is not connected to the outer face and not connected to an internal vertex. In all cases the conditions from Lemma 59 are violated.

Moreover, all M_{uv} and M_{uvt} markers must appear consecutively. To see this, note that if M_t appears between two markers M_{uv} or M_{uvt} the vertex w_t of M_t cannot be connected to an internal vertex of the outer face and apply Lemma 59.

These observations motivate the following restructuring of P-nodes. Let λ denote a P-node obtained from the SPQR-tree. The *parent marker* in skel(λ) is the marker that corresponds to the parent node of λ. Distinguish the three cases of whether the parent marker is M_s, M_t or M_{uv}. If the parent marker in skel(λ) is M_s all other markers must be M_t. In this case these markers can already be arbitrarily permuted and nothing further needs to be shown. Now consider the second case that the parent marker is M_t; see Figure 8.4 (a, b). Because all M_{uv} and M_{uvt} markers must appear consecutively, we create a new P-node μ that contains the parent marker of skel(λ), all M_t markers of skel(λ) and a single M_{uvt} marker to represent all M_{uv} markers (except for the parent marker) and all M_{uvt} markers of skel(λ). The M_{uvt}

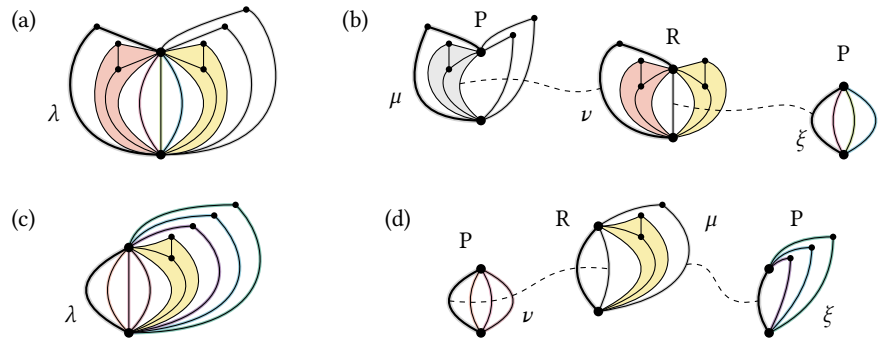


Figure 8.4: Splitting a P-node λ whose parent marker is M_t (a) into a chain of smaller nodes μ, ν, ξ (b), and splitting a P-node λ (c) whose parent marker is M_{uv} into smaller nodes μ, ν, ξ (d). The bold marker represents the component that contains the edge $e^\star$.

marker in skel(μ) corresponds to a new R-node ν that contains all M_{uvt} markers of skel(λ) and—because all M_{uv} markers must appear consecutively—a single M_{uv} marker. The M_{uv} marker in skel(ν) corresponds to a new P-node ξ that contains all M_{uv} markers of skel(λ). If skel(λ) contains no M_{uvt} marker we can include a M_{uv} marker instead of a M_{uvt} marker in skel(μ) and connect it directly to ξ, the node ν can then be omitted. Using Lemma 59 it can be verified that the markers in skel(μ) can indeed be arbitrarily permuted to obtain an upward-planar embedding of skel(μ), i.e., it satisfies the desired property of a P-node. Using Lemma 59 it can be verified that there can be at most two M_{uvt} markers in skel(ν) for skel(ν) to be upward planar. Therefore, if there are more than two M_{uvt} markers in skel(ν) then G is rejected as not upward planar and there exists no UP-tree for G. Otherwise skel(ν) has at most four markers. Recall that the parent marker is leftmost by convention. Moreover, if skel(ν) contains two M_{uvt} markers then Lemma 59 gives that the M_{uv} marker must appear between them. This means that the embedding of skel(ν) is fixed up to reversal, i.e., it satisfies the desired property of an R-node. Finally, the new node ξ also has the property that its markers can be arbitrarily permuted, i.e., it also satisfies the desired property of a P-node.

Finally, consider the third case that the parent marker is M_{uv}. See Figure 8.4 (c, d). Because all M_{uv} markers must appear consecutively, create a new P-node ν whose skeleton contains all M_{uv} markers of skel(λ) (except for the parent marker) and M_{uv} as the parent marker. Using Lemma 59 it can be verified that the markers in skel(ν) can be permuted arbitrarily. The fact that all M_{uv} and M_{uvt} markers must appear consecutively and the parent M_{uv} marker is leftmost implies that if G is upward planar

there can exist at most one M_{uvt} marker in skel(λ) and all M_t markers must appear consecutively (otherwise, G can be rejected as not upward planar). So, create a new P-node ξ whose skeleton contains all M_t markers of skel(λ) and M_{uv} as the parent marker. Again, using Lemma 59 it can be verified that the markers in skel(ξ) can be permuted arbitrarily. Lastly, create a new R-node μ whose skeleton contains a parent marker M_{uv} and markers M_{uv}, M_{uvt}, M_t in that order. The parent marker of skel(μ) corresponds to the parent of λ, the M_{uv} marker corresponds to ν, the M_{uvt} marker is the one from skel(λ) and the M_t marker corresponds to ξ. If skel(λ) contains no M_{uvt} marker then it can be omitted from skel(μ). Using Lemma 59 it can be verified that this (and only this) embedding of skel(μ) is upward planar. See Figure 8.6 (c) and (d) for a larger example. We conclude the following.

Lemma 60. *Let G be a biconnected digraph with a single source s and let $e^\star$ denote an edge incident to s. There exists a decomposition tree $\mathcal{T}$ that (i) represents all upward planar embeddings of G in which $e^\star$ is the leftmost edge around s, and (ii) the children of all P-nodes in $\mathcal{T}$ can be arbitrarily permuted.*

8.3.2 Arc Contractions

Recall that in SPQR-trees the skeletons of R-nodes are triconnected, i.e., their planar embedding is fixed up to reversal. So, every R-node offers one degree of freedom, namely, whether it has some reference embedding or the reversal thereof. In this section we alter our decomposition tree so that it has this same property.

By definition the marker corresponding to the parent node is leftmost in any embedding of a skeleton. Hence, this marker is incident to the outer face. Reversing the embedding of the skeleton is equivalent to choosing the other face incident to the marker as the outer face. Theorem 20 guarantees that any configuration of $\mathcal{T}$ can be composed to an upward planar embedding. This means that a skeleton can be reversed if and only if both faces incident to the parent marker can be chosen as the outer face. This can be checked with the upward planarity test for embedded single-source graphs due to Bertolazzi et al. [BDMT98], which also outputs the set of faces that can be chosen as the outer face. If both incident faces are candidates for the outer face this node does indeed offer a degree of freedom and we leave it unchanged. Otherwise, if only one incident face is a candidate for the outer face this node does not offer a degree of freedom. We then merge it with its parent node and contract the corresponding arc in the decomposition tree. This leads to an R-node with a larger skeleton.

See Figure 8.5 (a) for an upward planar graph G and (b) a decomposition tree thereof. Parts of the face sink graphs of skel(μ) and skel(ν) are shown in red, namely the two quadratic vertices dual to the faces incident to the parent marker and the edges

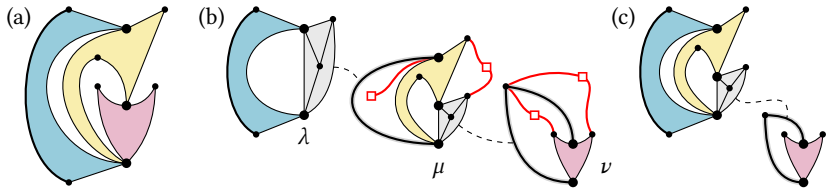


Figure 8.5: An upward planar graph G (a) with a decomposition tree (b). The node μ offers no degree of freedom so the arc (λ, μ) is contracted (c). The node ν does offer a degree of freedom and is therefore not contracted.

incident to those vertices. One criterion for a face to be a candidate for becoming the outer face due to Bertolazzi et al. is that there has to be a path from this face to the outer face in the face sink graph. This holds true for both faces incident to the parent marker in skel(ν), but not in skel(μ). Therefore the arc (μ, ν) is not contracted but the arc (λ, μ) is contracted. This leads to the decomposition tree shown in (c). See also Figure 8.6 (c) and (d) for a larger example.

Lemma 61. *Let G be a biconnected digraph with a single source s and let $e^\star$ denote an edge incident to s. There exists a decomposition tree $\mathcal{T}$ that (i) represents all upward planar embeddings of G in which $e^\star$ is the leftmost edge around s, and (ii) the children of all P-nodes in $\mathcal{T}$ can be arbitrarily permuted. (iii) the skeletons of all R-nodes in $\mathcal{T}$ can be reversed.*

We call the decomposition tree $\mathcal{T}$ the *UP-tree of G relative to $e^\star$*.

8.3.3 Computation in Linear Time

Let G be a biconnected digraph with a single source s and let $e^\star$ denote an edge incident to s. Recall that the construction of the UP-tree $\mathcal{T}$ of G relative to $e^\star$ consists of the following seven steps. 1. Construct the SPQR-tree $\mathcal{T}$ of G in linear time [HT73, GM00]. 2. For each pair of vertices u, v that are the poles of a marker in some skeleton of $\mathcal{T}$, we have to determine whether $u < v$ in G. To compute this information for all pairs in linear time, we use a union-find-based technique described by Bläsius et al. [BKR18]. Process all skeletons of $\mathcal{T}$ and for every pair of poles u, v that is encountered register v as a *candidate* at u and register u as a candidate at v. Next, initialize every vertex of G in its own singleton set. Then, process each vertex u in some reverse topological order of G. Unify the singleton set of u with the sets of its direct descendants in G. Now for any candidate v stored at u we can query in whether u and v belong to the same set, which is equivalent to $u < v$. Note that the

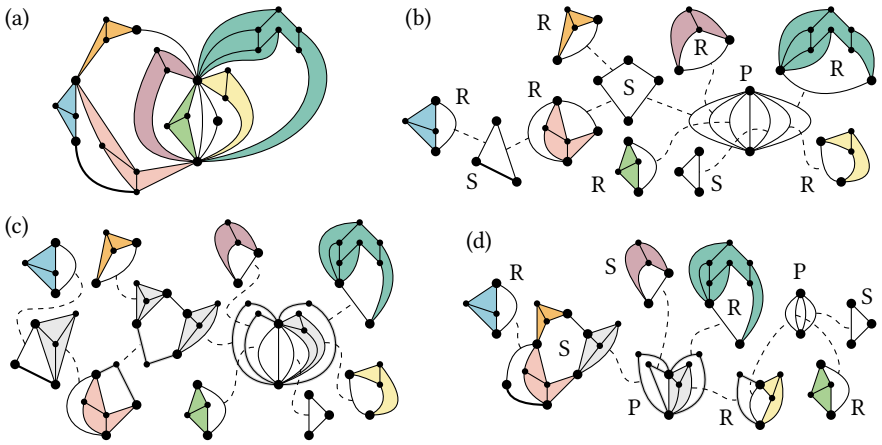


Figure 8.6: Construction of the UP-tree. An upward planar biconnected single-source graph (a), the SPQR-tree of its underlying undirected graph (b) with the Q-nodes omitted, the result of replacing virtual edges with markers (c) and the UP-tree after splitting P-nodes and contracting arcs (d).

operands to all unify operations are completely determined by the structure of G. We exploit this fact to run the linear-time union-find algorithm due to Gabow and Tarjan [GT85]. 3. For each arc $a = (\mu, \nu)$ of $\mathcal{T}$, decide whether the poles of a are sources, sinks or internal vertices in $G(\mu)$ and $G(\nu)$. This information can be found using a simple bottom-up technique. We first compute the indegree and outdegree of every node of G. We then perform a depth-first traversal of $\mathcal{T}$. We maintain a list of the number of incoming and outgoing edges for each node seen so far, which is updated when a Q-node is visited. Upon entering a subtree, we store these numbers for the poles of the arc leaving the subtree at the root of the subtree. Upon leaving a subtree, we can now calculate the differences between the current numbers and the stored numbers, which gives the in- and outdegree of the poles in the graph $G(\mu)$. Using the in- and outdegree of the poles in G computed earlier, we can also compute the in- and outdegree of the poles in $G(\nu)$. This step clearly takes linear time. 4. In each skeleton, replace all virtual edges with their respective markers. With the information that was computed in the previous step and the fact that all markers have constant size this step is feasible in linear time. 5. Construct a configuration of $\mathcal{T}$ by running the linear-time upward planar embedding algorithm of Bertolazzi et al. [BDMT98] on every skeleton. Because the size of all skeletons is linear in the size of G this step takes linear time. 6. Perform P-node splits. The running time spent on

one P-node is clearly linear in the size of its skeleton. This gives linear running time overall. 7. Perform arc contractions. The upward planarity test for fixed embeddings due to Bertolazzi et al. runs in linear time. Contracting an arc is feasible in constant time. This gives linear running time overall.

Theorem 21. *Let G be a biconnected digraph with a single source s and let $e^\star$ denote an edge incident to s. The UP-tree $\mathcal{T}$ of G relative to $e^\star$ is a decomposition tree whose internal nodes are (i) S-nodes whose skeletons have a fixed embedding, (ii) R-nodes whose skeletons have a fixed embedding up to reversal, or (iii) P-nodes where the markers can be arbitrarily permuted in the skeleton and whose leaves are Q-nodes that offer no embedding choice. The configurations of $\mathcal{T}$ correspond bijectively to the upward planar embeddings of G where $e^\star$ appears leftmost around s. Moreover, $\mathcal{T}$ can be computed in linear time.*

8.4 Partial Upward Embedding

In this section we apply the UP-tree to solve the partial upward embedding problem in quadratic time. A *partially embedded graph* is a tuple $(G, H, \mathcal{H})$, where G is a planar graph, H is a subgraph of G and $\mathcal{H}$ is a planar embedding of H. The *partial embedding problem* asks whether there exists an embedding $\mathcal{G}$ of G that extends $\mathcal{H}$. Angelini et al. solve the partial embedding problem in linear time [Ang+15c]. In the case that H is connected their algorithm considers every triple of edges (e, f, g) in H that share a common endpoint v and enforces the constraints imposed by these edges in the SPQR-tree $\mathcal{T}$. Note that e, f, g each correspond to a Q node in $\mathcal{T}$. Because $\mathcal{T}$ is a tree there is exactly one node μ in $\mathcal{T}$ so that the paths from μ to these Q nodes are disjoint. The relative order of e, f, g in the embedding represented by $\mathcal{T}$ is determined by the embedding of $\text{skel}(\mu)$. If $\text{skel}(\mu)$ offers no embedding choice (as in S nodes) determine whether the ordering of e, f, g given by $\mathcal{H}$ is the same as the one given by the unique embedding of $\text{skel}(\mu)$. If not, reject the instance. If $\text{skel}(\mu)$ has two possible embeddings (as in R nodes) the ordering of e, f, g given by $\mathcal{H}$ fixes one of the two embeddings of $\text{skel}(\mu)$ as the only candidate. Finally, if μ is a P node the ordering of e, f, g given by $\mathcal{H}$ restricts the set of admissible permutations of the virtual edges in $\text{skel}(\mu)$. The algorithm collects all these constraints and checks whether they can be fulfilled at the same time.

A *partially embedded upward graph* is defined as a tuple $(G, H, \mathcal{H})$, where G is an upward planar graph, H is a subgraph of H and $\mathcal{H}$ is an upward planar embedding of H. Note that UP-trees have all properties of SPQR-trees that are needed in the algorithm described above. In particular, the markers in P-nodes may be arbitrarily permuted, R-nodes may be reversed and all other nodes offer no embedding choice. Hence, we use the UP-tree as a drop-in replacement for the SPQR-tree in the algorithm

of Angelini et al. to obtain an algorithm that solves the partial upward embedding problem for biconnected single-source graphs. Note that the UP-tree is rooted at some edge that must be embedded as the leftmost edge around the source of the graph. We may have to try a linear number of candidate edges in the worst case. This gives the following.

Theorem 22. *The partial upward embedding problem can be solved in quadratic running time for biconnected single-source graphs and connected partial embeddings.*

8.5 Conclusion

We have developed the UP-tree, which is an SPQR-tree-like embedding representation for upward planarity. We expect that the UP-tree is a valuable tool that makes it possible to translate existing constrained planar embedding algorithms that use SPQR-trees to the upward planar setting. As an example, we have demonstrated how to use the UP-tree as a drop-in replacement for the SPQR-tree in the partial embedding extension problem, solving the previously open partial upward embedding extension problem for the biconnected single-source case.

Part III

Custom Drawing Styles

9 Multilevel Planarity

In this chapter, we introduce and study *multilevel planarity*, a generalization of upward planarity and level planarity. Let $G = (V, E)$ be a directed graph and let $\ell : V \to \mathcal{P}(\mathbb{Z})$ be a function that assigns a finite set of integers to each vertex. A multilevel-planar drawing of G is a planar drawing of G such that for each vertex $v \in V$ its y-coordinate $y(v)$ is in $\ell(v)$, and each edge is drawn as a strictly y-monotone curve.

We present linear-time algorithms for testing multilevel planarity of embedded graphs with a single source and of oriented cycles. Complementing these algorithmic results, we show that multilevel-planarity testing is NP-complete even in very restricted cases.

This chapter extends work initiated as part of Paul Jungeblut's bachelor's thesis [Jun17] and is based on joint work with Lukas Barth, Paul Jungeblut and Marcel Radermacher [BBJR19, BBJR21].

9.1 Introduction

Testing a given graph for planarity, and, if the graph is planar, finding a planar drawing of it, are classic algorithmic problems. However, one is often not interested in just any planar drawing, but in one that has some additional properties. Examples of such properties include that a given existing partial drawing should be extended [Ang+15c, JKR13] or that some parts of the graph should appear clustered together [DF09, Jel+09].

There also exist notions of planarity specifically tailored to directed graphs. While testing upward planarity of a graph is an NP-complete problem in general [GT01], efficient algorithms are known for outerplanar graphs, single-source graphs and for embedded graphs [Pap94, BDLM94, HL96, BDMT98]. One notable constrained version of upward planarity is that of level planarity. Level-planarity testing and drawing is feasible in linear time [JL02]. There exist further level-planarity variants on the cylinder and on the torus [Ang+20, BBF05] and there has been considerable research on further-constrained versions of level planarity. Examples include ordering the vertices on each level according to so-called constraint trees [Ang+15b, HH07], clustered level planarity [Ang+15b, FB04], partial level planarity [BR17] and ordered level planarity [KR19]. Finally, an undirected graph $G = (V, E)$ is *leveled planar* if there exists an assignment $\gamma : V \to \mathbb{Z}$ and a planar drawing Γ of G where for each vertex $v \in V$ its y-coordinate is $\gamma(v)$ and for each edge $\{u, v\}$ it is $|\gamma(u) - \gamma(v)| = 1$. Recognizing leveled planar graphs is NP-complete [HR92]. Recently, relationships between track layouts, layered pathwidth and leveled planarity have been studied, and bipartite outerplanar graphs and square graphs have been shown to be leveled planar [Ban+19].

Contribution and Outline. In this chapter, we introduce and study *multilevel planarity*. Let $\mathcal{P}(\mathbb{Z})$ denote the power set of integers. The input of the multilevel-planarity testing problem consists of a directed graph $G = (V, E)$ together with a function $\ell : V \to \mathcal{P}(\mathbb{Z})$, called a multilevel assignment, which assigns admissible levels, represented as a set of integers, to each vertex. A multilevel-planar drawing of G is a planar drawing of G such that for the y-coordinate of each vertex $v \in V$ it holds that $y(v) \in \ell(v)$, and each edge is drawn as a strictly y-monotone curve. Figure 9.1 shows in blue an example of a multilevel-planar graph. It visualizes an excerpt of the genealogy of European royalty [Wik19, DD11, Kam01]. In this graph, vertices are associated with individuals. The intervals assigned to a vertex are derived from the corresponding individual's lifespan and visualized as gray bars. Dashed edges represent marriages and solid lines connected to dashed edges represent descent of children. To improve readability the dashed edges are drawn as horizontal line segments even though the formal definition requires *strictly* y-monotone curves.

This chapter is structured as follows. We start by discussing some preliminaries, including the relationship between multilevel planarity and existing planarity variants in Section 9.2. Then, we present linear-time algorithms that test multilevel planarity of embedded single-source graphs and of oriented cycles with multiple sources in Sections 9.3 and 9.4, respectively. In Section 9.5, we complement these algorithmic results by showing that multilevel-planarity testing is NP-complete for abstract single-source graphs, for oriented trees and for embedded multi-source

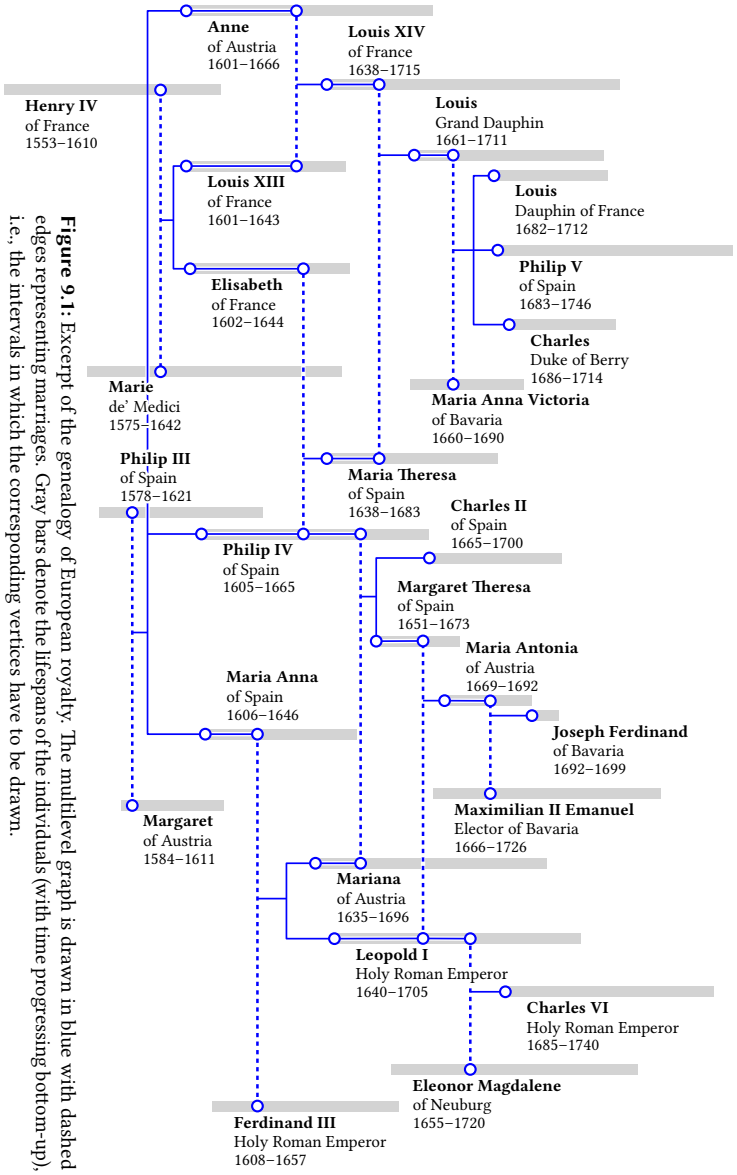


Figure 9.1: Excerpt of the genealogy of European royalty. The multilevel graph is drawn in blue with dashed edges representing marriages. Gray bars denote the lifespans of the individuals (with time progressing bottom-up), i.e., the intervals in which the corresponding vertices have to be drawn.

graphs where $|\ell(v)| \leq 2$ for all $v \in V$. We finish with some concluding remarks in Section 9.6.

9.2 Preliminaries

This section consists of three parts. First, we introduce basic terminology and notation. Second, we discuss the relationship between multilevel planarity and existing planarity variants for directed graphs. Third, we define a normal form for multilevel assignments, which simplifies the arguments in Sections 9.3 and 9.4.

Basic Terminology. We say that two planar drawings of a graph are *combinatorially equivalent* if they define the same combinatorial embedding and have the same outer face. A *multilevel assignment* $\ell : V \to \mathcal{P}(\mathbb{Z})$ assigns a finite set of integers to each vertex. An upward-planar drawing is *multilevel planar* if $y(v) \in \ell(v)$ for all $v \in V$. Note that any finite set of integers can be represented as a finite list of finite integer intervals. We choose this representation to be able to represent sets of integers that contain large intervals of numbers more efficiently.

A directed, acyclic and planar graph with a single source s is an *sT-graph*. An *sT*-graph with a single sink t and an edge (s, t) is an *st-graph*. In any upward-planar drawing of an *st*-graph, the unique source and sink are the lowest and highest vertices, respectively, and both are incident to the outer face. For a face f of a planar drawing, an incident vertex v is called *face source* (*face sink*) if all edges incident to f and v are outgoing (incoming). Note that a face source or face sink does not need to be a source or sink in G. We will frequently add incoming edges to sources and outgoing edges to sinks during later constructions, referring to this as *source canceling* and *sink canceling*, respectively. An *oriented path* of length k is a sequence of vertices $(v_1, v_2, \ldots, v_{k+1})$ such that for all $1 \leq i \leq k$ either the edge (v_i, v_{i+1}) or the edge (v_{i+1}, v_i) is in G. A *directed path* of length k is a sequence of vertices $(v_1, v_2, \ldots, v_{k+1})$ such that for all $1 \leq i \leq k$ the edge (v_i, v_{i+1}) is in G. Let $u, v \in V$ be two distinct vertices. Vertex u is a *descendant* of v in G, if there exists a directed path from v to u. A *topological ordering* is a function $\tau : V \to \mathbb{N}$ such that for every $v \in V$ and for each descendant u of v it is $\tau(v) < \tau(u)$.

Relationship to Existing Planarity Variants. Multilevel-planarity testing is a generalization of level planarity testing. To see this, let $G = (V, E)$ be a directed graph together with a level assignment $\gamma : V \to \mathbb{Z}$. Define $\ell(v) = \{\gamma(v)\}$ for all $v \in V$. It is readily observed that a drawing Γ of G is level planar with respect to γ if and only if Γ is multilevel planar with respect to ℓ. Therefore, level planarity reduces to multilevel planarity in linear time.

Multilevel-planarity testing is also a generalization of upward planarity testing. Without loss of generality, the vertices in an upward-planar drawing can be assigned unique integer y-coordinates so that there is at least one vertex on each level in $[1, |V|]$. Hence, upward planarity of G can be tested by setting $\ell(v) = [1, |V|]$ for all $v \in V$ and testing the multilevel planarity of G with respect to ℓ. Therefore, upward planarity reduces to multilevel planarity in linear time. By then restricting the multilevel assignment, multilevel planarity can also be seen as a constrained version of upward planarity. Garg and Tamassia [GT01] showed that upward-planarity testing is NP-complete. It is easy to see that multilevel-planarity testing is in NP and so we conclude the following.

Theorem 23. *Multilevel-planarity testing is NP-complete.*

Multilevel planarity is related to leveled planarity. Both notions ask about the existence of a certain y-coordinate assignment γ. However, multilevel planarity is defined for directed graphs, i.e., for two adjacent vertices it is known which one has the greater y-coordinate, whereas leveled planarity is defined for undirected graphs. And for adjacent vertices u, v in a leveled planar drawing it must be $|\gamma(u) - \gamma(v)| = 1$, whereas no such restriction exists for the multilevel-planar drawings.

Multilevel Assignment Normal Form. A multilevel assignment ℓ has *normal form* if for all $(u, v) \in E$ it is $\min \ell(u) < \min \ell(v)$ and $\max \ell(u) < \max \ell(v)$. Some proofs are easier to follow for multilevel assignments in normal form. The following lemma justifies that we may assume without loss of generality that ℓ has normal form.

Lemma 62. *Let $G = (V, E)$ be a directed graph together with a multilevel assignment ℓ. Then there exists a multilevel assignment ℓ' in normal form such that a drawing of G is multilevel planar with respect to ℓ if and only if it is multilevel planar with respect to ℓ'. Moreover, ℓ' can be computed in linear time.*

Proof. The idea is to convert $\ell(v)$ into $\ell'(v) \subseteq \ell(v)$ for all $v \in V$ by finding a lower bound l_v and an upper bound u_v for the level of v, and setting $\ell'(v) = \ell(v) \cap [l_v, u_v]$. If this set is empty there exists no multilevel-planar drawing. To find the lower bound, iterate over the vertices in increasing order with respect to some topological ordering τ of G. Because all edges have to be drawn as strictly y-monotone curves, for each vertex $v \in V$ it must be $y(v) > \max_{(w,v)\in E} l_w$. So, set l_v to the maximum of $\min \ell(v)$ and $\max_{(w,v)\in E} l_w + 1$. Analogously, to find the upper bound, iterate over V in decreasing order with respect to τ. Again, because edges are drawn as strictly y-monotone curves, for each vertex $v \in V$ it must hold true that $y(v) < \min_{(v,w)\in E} u_w$. Therefore, set u_v to the minimum of $\max \ell(v)$ and $\min_{(v,w)\in E} u_w - 1$. This means that in any

multilevel-planar drawing of G the y-coordinate of $v \in V$ is $y(v) \in \ell(v) \cap [l_v, u_v]$. So it follows that a drawing of G is multilevel planar with respect to ℓ if and only if it is multilevel planar with respect to ℓ'.

To see that the running time is linear, note that a topological ordering of G can be computed in linear time and every vertex and edge is handled at most twice during the procedure described above. Because every level candidate in ℓ is removed at most once, the total running time is $\mathcal{O}(n + \sum_{v \in V} |\ell(v)|)$, i.e., linear in the size of the input. □

9.3 Embedded sT-Graphs

In this section, we characterize multilevel-planar sT-graphs as subgraphs of certain planar st-graphs. Similar characterizations exist for upward planarity and level planarity [DT88, Lei98]. The idea behind our characterization is that for any given multilevel-planar drawing, we can find a set of edges that can be inserted without rendering the drawing invalid, and which make the underlying graph an st-graph. For these st-graphs multilevel planarity and planarity coincide and we can use existing linear-time algorithms to find (multilevel-)planar drawings. This technique is similar to the one found by Bertolazzi et al. [BDMT98], and in fact is built on top of it.

To use this characterization for multilevel-planarity testing, we cannot require a multilevel-planar drawing to be given. We show that if we choose the set of edges to be inserted carefully, the respective set of edges can be inserted into *any* multilevel-planar drawing for a fixed combinatorial embedding. An algorithm constructing such an edge set can therefore be used to test for multilevel planarity of embedded sT-graphs, resulting in Theorem 24. The algorithm is constructive in the sense that it finds a multilevel-planar drawing if one exists. In Section 9.5, we show that testing multilevel planarity of sT-graphs without a fixed combinatorial embedding is NP-hard. Recall that every multilevel-planar drawing is upward planar. We now prove that the vertex with the largest y-coordinate on the boundary of each face is the same across all combinatorially equivalent drawings. To this end we use the notion of large and small angles at sink switches that Bertolazzi et al. [BDMT98] have used for biconnected graphs. We extend this notion to simply-connected sT-graphs. Lemma 63 is used to argue that such angles are well-defined, Lemma 64 extends the observations of Bertolazzi et al. to simply-connected graphs and Lemma 65 sets the foundation to our st-graph extension.

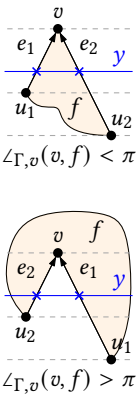
Lemma 63. *Let $G = (V, E)$ be an sT-graph together with an upward-planar drawing Γ. Further, let f be a face of Γ and let c denote a sink switch of f. Then c appears exactly once on the cyclic walk around the boundary of f.*

Proof. Suppose $w = \left[v_1 = c, v_2, \ldots, v_j = c, \ldots, v_m = v_1 = c\right]$ with $2 < j < m - 1$ is the cyclic walk around f. We show that c is not a sink switch. Construct a Jordan curve C by closely following w along $v_1 = c, v_2, \ldots, v_j = c$ within f and then crossing through c to close the curve. Let G_{outer} denote the subgraph of G induced by the vertices in the exterior of C and on C. Let G_{inner} denote the subgraph of G induced by the vertices in the interior of C and on C.

Consider the case that $H \in \{G_{\text{outer}}, G_{\text{inner}}\}$ does not contain s. Because G is an sT-graph there exists in G a directed path from s to every vertex in H. Because no edge crosses C and the only vertex that lies on C is c all these paths contain c. Therefore, there exists in H a directed path from c to every vertex of H, i.e., H is an sT-graph with source c. Either v_2 or v_{j+1} lies in H by construction and so one of the edges $(v_1 = c, v_2), (v_j = c, v_{j+1})$ exists in G, i.e., c is not a sink switch of f.

Now consider the other case, namely that both G_{outer} and G_{inner} contain s. By construction G_{outer} and G_{inner} share only the vertex c, i.e., $c = s$. Because s is the source of G it cannot be the sink switch of any face. □

Let v be a sink switch of a face f in Γ. Further, let $e_1 = (u_1, v), e_2 = (u_2, v)$ denote the edges incident to f and v. By Lemma 63 the choice of these edges is unique. Let f lie to the right of e_1 and to the left of e_2 with respect to the directions of those edges. Because Γ is upward there exists a horizontal line y that intersects both e_1 and e_2 exactly once but does not intersect v. For $i = 1, 2$, let x_i denote the x-coordinate of the intersection of y and e_i. Define $\angle_{\Gamma,f}(v)$ as *small* (written as $\angle_{\Gamma,f}(v) < \pi$) if $x_1 < x_2$ (see the upper part of the figure on the right) and as *large* (written as $\angle_{\Gamma,f}(v) > \pi$) if $x_1 \geq x_2$ (see the lower part of the figure on the right). Note that $x_1 = x_2$ implies $e_1 = e_2$, i.e., $u_1 = u_2$ is a cutvertex. This sets the stage for the following.



Lemma 64. *Let G be an sT-graph together with an upward-planar drawing Γ. Then the following properties hold:*

1. *For each sink switch v on the boundary of the outer face h it is $\angle_{\Gamma,h}(v) > \pi$.*
2. *For each inner face f of Γ there is exactly one sink switch t_f on the boundary of f with $\angle_{\Gamma,f}(t_f) < \pi$, namely the vertex with greatest y-coordinate among all vertices incident to f.*

Proof. Use induction over the number of biconnected components of G. In the base case of a biconnected graph both properties were observed by Bertolazzi et al. [BDMT98, page 138, Facts 2 and 3].

Let c be a cutvertex of G. Then there exists a face f of Γ such that c appears more than once on the cyclic walk around f. Let the cyclic walk around f be denoted by $w = \left[v_1 = c, v_2, \ldots, v_j = c, \ldots, v_m = v_1 = c\right]$, where $2 < j < m - 1$. Let C, G_1, G_2 be defined as in the proof of Lemma 63.

If G_i does not contain s we have already shown that it is an sT-graph with source c. Consider the case that G_i does contain s. Because G is an sT-graph there exists in G a directed path from s to every vertex in G_i. Suppose that such a directed path p contains a vertex not in G_i, i.e., a vertex in G_{3-i} other than c. Because no edge crosses C and the only vertex that lies on C is c this means that p must contain c twice. This contradicts the fact that G is acyclic. Hence, there exists in G_i a directed path from s to every vertex of of G_i, i.e., G_i is an sT-graph with source s.

The drawing Γ induces drawings Γ_1, Γ_2 of G_1, G_2, respectively. We have just shown that G_1 and G_2 are sT-graphs. Furthermore, each one has strictly fewer biconnected components than G. Therefore, properties 1 and 2 hold for G_1, G_2 by induction.

Observe that Γ can be obtained from Γ_1 by inserting Γ_2 into C. This changes no angles, except at c which is not a sink switch of f by Lemma 63. Every face of Γ except for f exists in one of Γ_1, Γ_2 and the claimed properties are true by induction. Let f_1 denote the face of Γ_1 that contains the interior of C. Let h_2 denote the outer face of Γ_2, i.e., the face of Γ_2 that contains the exterior of C. Face f in Γ is obtained from combining the faces f_1 in Γ and h_2 of Γ_2. Every sink switch v on the boundary of f in Γ is a sink switch on the boundary of either f_1 in Γ_1 or h_2 in Γ_2. If v is a sink switch on the boundary of h_2 Property 1 implies $\angle_{\Gamma_2,h_2}(v) > \pi$ and therefore $\angle_{\Gamma,f}(v) > \pi$. Now consider the case that v is a sink switch on the boundary of f_1. If f_1 is the outer face of Γ_1 then f is the outer face of Γ. Then Property 1 implies $\angle_{\Gamma_1,f_1}(v) > \pi$ and therefore $\angle_{\Gamma,f}(v) > \pi$. If f_1 is an inner face of Γ_1 then f is an inner face of Γ. Property 2 implies that there is exactly one sink switch v on the boundary of f_1 in Γ_1 with $\angle_{\Gamma_1,f_1}(v) < \pi$, namely the vertex with the greatest y-coordinate among all vertices incident to f_1. Then v is the only sink switch on the boundary of f in Γ with $\angle_{\Gamma,f}(v) < \pi$. Also, because C is contained within f_1 vertex v is the vertex with the greatest y-coordinate among all vertices incident to f. □

We are now ready to prove the following.

Lemma 65. *Let G be an sT-graph, let Γ, Γ' be combinatorially equivalent upward-planar drawings of G and let f be an inner face of Γ and Γ'. Then the vertex with the greatest y-coordinate among all vertices incident to f is the same in Γ and Γ'.*

Proof. Let t_f denote the vertex with the greatest y-coordinate among all vertices incident to f in Γ. Further let $e_1 = (v_1, t_f)$ and $e_2 = (v_2, t_f)$ be the edges incident to f and t_f (by Lemma 63 and because t_f is a sink switch e_1, e_2 are the only such edges). Property 2 of Lemma 64 states that $\angle_{\Gamma,f}(v) < \pi$, i.e., $e_1 \neq e_2$. Assume that t_f does

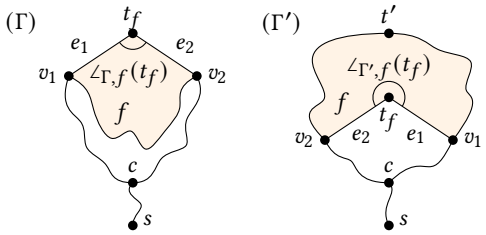


Figure 9.2: Proof of Lemma 65.

not have the greatest y-coordinate of all vertices incident to f in Γ'. See Figure 9.2. Because G has a single source s, there exist directed paths p_1 and p_2 from s to v_1 and v_2, respectively. Then the left-to-right order of the edges e_1 and e_2 in Γ and Γ' is determined by the order of the outgoing edges at the last common vertex c on p_1 and p_2. Because $e_1 \neq e_2$ it is $c \neq t_f$. Let $t' \neq t_f$ be the vertex with greatest y-coordinate of all vertices incident to f in Γ'. Then it holds that $\angle_{\Gamma',f}(t') < \pi$ and from Property 2 of Lemma 64 it follows that $\angle_{\Gamma',f}(t_f) > \pi$. Since Γ and Γ' have the same underlying combinatorial embedding, the clockwise cyclic walk around f is identical in both drawings. But because $\angle_{\Gamma,f}(t_f) < \pi$ and $\angle_{\Gamma',f}(t_f) > \pi$, the left-to-right order of the outgoing edges of c is different in Γ and Γ'. This means that Γ and Γ' are not combinatorially equivalent: If c has an incoming edge the cyclic order of the edges around c is different in Γ and Γ'. Otherwise it is $s = c$ and because (s, t) is the left-most edge by definition the cyclic order of the edges around c is different in Γ and Γ'. □

Bertolazzi et al. showed that any sT-graph with an upward-planar drawing can be extended to an st-graph with an upward-planar drawing that extends the original drawing [BDLM94, BDMT98]. More formally, let $G = (V, E)$ be an sT-graph together with an upward-planar drawing Γ. There exists an st-graph $G_{st} = (V \cup \{t\}, E \cup E_{st})$ where t is the unique sink together with an upward-planar drawing Γ_{st} that extends Γ. Moreover, G_{st} and Γ_{st} can be computed in linear time. Note that in general it is possible for a given E_{st} to choose an upward-planar drawing Γ of G so that the additional edges in E_{st} cannot be added into Γ as y-monotone curves. For an example, see Figure 9.3, where augmenting with the red and black edge works for the drawing shown in (a) but not for the one shown in (b), whereas augmenting with the blue and black edge works for both drawings. In Lemma 66 we show that there is a set E_{st} that can be added into any upward planar drawing with the same combinatorial embedding as Γ. In a way, this is the most general set E_{st}.

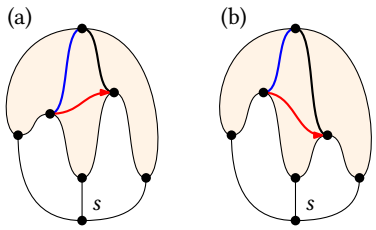


Figure 9.3: Some edges are not admissible for augmentation in Lemma 66.

Lemma 66. *Let $G = (V, E)$ be a directed sT-graph with a fixed combinatorial embedding. Then there exists an st-graph $G_{st} = (V \dot\cup \{t\}, E \dot\cup E_{st})$, where t is the unique sink, such that for any upward-planar drawing Γ of G there exists an upward-planar drawing Γ_{st} of G_{st} that extends Γ. Moreover, G_{st} can be computed in linear time.*

Proof. Start by finding an initial upward-planar drawing Γ of G in linear time using the algorithm due to Bertolazzi et al. [BDMT98]. The algorithm also outputs an embedded st-graph $G^0_{st} = (V \dot\cup \{t\}, E \dot\cup E^0_{st})$ together with an upward-planar drawing Γ^0_{st} that extends Γ. This means that Γ^0_{st} is obtained from Γ by drawing each edge $(u, v) \in E^0_{st}$ within some face $F^0(u, v)$ of Γ. Define a function T that maps the faces of Γ to the vertex set $V \dot\cup \{t\}$ as follows. If f is the outer face of Γ define $T(f) = t$ and if f is an inner face of Γ define $T(f) = t_f$, where t_f denotes the sink switch of f with $\angle_{\Gamma,f}(t_f) < \pi$, which is unique by Property 2 of Lemma 64 and the same for all combinatorially equivalent drawings by Lemma 65. We show that $E_{st} = \{(u, T(F^0(u, v))) \mid (u, v) \in E^0_{st}\}$ satisfies the claim.

For each face f of Γ the vertex $T(f)$ has the greatest y-coordinate among all vertices incident to f. This means that for each edge $(u, v) \in E_{st}$ the y-coordinate of u is smaller than the y-coordinate of v in Γ. Together with the fact that Γ is an upward-planar drawing this means that G_{st} is acyclic. Recall that G^0_{st} is an st-graph, i.e., every sink of G is incident to an outgoing edge in E^0_{st}. By construction of E_{st} every sink of G is then also incident to an outgoing edge in E_{st}. Therefore G_{st} has exactly one sink, namely t. Together with the fact that G_{st} is acyclic this shows that G_{st} is an st-graph.

Now let Γ' be an upward-planar drawing of G. Recall that the embedding of G is fixed, so Γ' is combinatorially equivalent to Γ. We show that we can extend Γ' to an upward-planar drawing of G_{st}. To this end, we insert the vertex t and then insert the edges in E_{st} into Γ' one after the other. Let Y denote the greatest y-coordinate of any vertex in Γ'. Insert t into Γ' with y-coordinate $Y + 1$.

Let (u, v) be an edge in E_{st}. The idea is that it is possible to walk from any vertex x on the boundary of f upwards to t_f. If x is incident to an outgoing edge (x, w) incident to f follow that edge to w. Because Γ' is an upward-planar drawing this segment is y-monotone. Then continue walking up from w to t_f. Otherwise, if x is not incident to an outgoing edge incident to f, it is a sink switch of f. There are two cases.

1. It is $\angle_{\Gamma',f}(x) > \pi$. Then walk up vertically from x within f. If f is an inner face of Γ', this walk will meet either a vertex w or an edge (x', w). In the former case continue walking up from w. The latter case has two subcases. The first subcase is $(x', w) \notin E_{st}$. Then (x', w) is incident to f, so follow the edge to w and then continue walking up from w. The second subcase is $(x', w) \in E_{st}$. Then (x', w) is an edge that was inserted into f in Γ' previously. Note that all edges inserted into f have endpoint t_f, i.e., $w = t_f = v$, so follow (x', w) to its endpoint w to complete the drawing of (u, v).

   If f is the outer face of Γ', either one of the previous situations occurs, or we could walk vertically up infinitely without meeting an edge or a vertex. Note that this means $v = t$. In this case stop walking up when the y-coordinate is Y, bend and then connect to t with a straight line segment.

2. It is $\angle_{\Gamma',f}(x) < \pi$. From Lemma 64 it follows that $x = t_f$.

□

We now have a set of edges that can be used to complete G into G_{st}. If a multilevel-planar drawing for the given combinatorial embedding of G respecting ℓ exists, then it must also exist for G_{st}. However, the property of ℓ being in normal form might not be fulfilled anymore in G_{st} because of the added edges. We therefore need to bring ℓ into normal form ℓ' again. Lemma 62 tells us that this does not impact multilevel planarity. We conclude that G is multilevel planar with respect to ℓ if and only if G_{st} is multilevel planar with respect to ℓ'. The final property we need is that G_{st} is level planar with respect to any level assignment γ (recall that γ is a level assignment of G_{st} if for each directed edge (u, v) in G_{st} it is $\gamma(u) < \gamma(v)$). The following is due to Leipert [Lei98, Theorem 5.1 and page 121], who notes that an algorithm for drawing upward planar graphs by Di Battista and Tamassia [DT88, Theorem 3.5] can be adapted for the level-planar setting.

Lemma 67. *Let G be an st-graph on n vertices together with a level assignment γ. Then for any combinatorial embedding of G there exists a combinatorially equivalent drawing of G that is level planar with respect to γ. Moreover, such a drawing has $\mathcal{O}(n)$ size and can be computed in $\mathcal{O}(n)$ time.*

If ℓ' is in normal form, $\ell'(v) \neq \emptyset$ is a necessary and sufficient condition that there exists a level assignment $\gamma : V \to \mathbb{Z}$ with $\gamma(v) \in \ell'(v)$ for all $v \in V$. Setting $\gamma(v) = \min \ell'(v)$ is one possible such level assignment. Then G is level planar with respect to γ and therefore multilevel planar with respect to ℓ, resulting in the characterization of multilevel-planar st-graphs:

Corollary 7. *Let G be an st-graph together with a multilevel assignment ℓ in normal form. Then there exists a multilevel-planar drawing for any combinatorial embedding of G if and only if $\ell(v) \neq \emptyset$ for all v.*

For a constructive multilevel-planarity testing algorithm, we now first take the edge set computed by the algorithm by Bertolazzi et al. [BDMT98] and modify it using Lemma 66 to complete any sT-graph to an st-graph. Note that for this step, we need a fixed combinatorial embedding to be given, as is required by the second property of Lemma 64. Once arrived at an st-graph, we check the premise of Corollary 7. Then, we either output that the graph is not multilevel planar or use Lemma 67 to find a multilevel-planar drawing in linear time. This concludes the testing algorithm:

Theorem 24. *Let G be an sT-graph with a multilevel assignment ℓ together with a combinatorial embedding and an outer face. Then it can be decided in linear time whether there exists a combinatorially equivalent multilevel-planar drawing of G. If so, such a drawing can be computed within the same running time.*

Our algorithm uses the fact that to augment sT-graphs to st-graphs, only edges connecting sinks to other vertices need to be inserted. For graphs with multiple sources and multiple sinks, further edges connecting sources to other vertices need to be inserted. The interactions that occur then are very complex: In Section 9.5, we show that deciding multilevel planarity is NP-complete for embedded multi-source graphs. In the next section, we identify oriented cycles as a class of multi-source graphs for which multilevel planarity can be decided efficiently.

9.4 Oriented Cycles

In this section, we present a constructive multilevel-planarity testing algorithm for oriented cycles, i.e., directed graphs whose underlying undirected graph is a simple cycle. We start by giving a condition for when an oriented cycle $G = (V, E)$ together with some level assignment γ admits a level-planar drawing. This condition yields an algorithm for the multilevel-planar setting.

In this section, γ is always a level assignment and ℓ is always a multilevel assignment. Define $\max \gamma = \max\{\gamma(v) \mid v \in V\}$ and $\min \gamma = \min\{\gamma(v) \mid v \in V\}$. Further set $\max \ell = \max\{\max \ell(v) \mid v \in V\}$ and $\min \ell = \min\{\min \ell(v) \mid v \in V\}$.

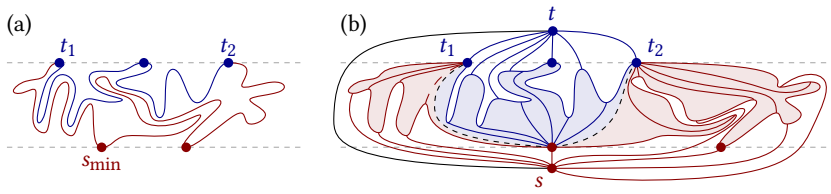


Figure 9.4: Augmenting the oriented cycle in (a) to an st-graph in (b). See the proof of Lemma 68.

Let $S_{\min} \subset V$ be sources with minimal level, i.e., $S_{\min} = \{v \in V \mid \gamma(v) = \min \gamma\}$, and let $T_{\max} \subset V$ be the sinks with maximal level. We call sources in $S_{\min}$ *minimal sources*, sinks in $T_{\max}$ are *maximal sinks*. Two maximal sinks $t_1, t_2 \in T_{\max}$ are *consecutive* if there is an oriented path between t_1 and t_2 that does not contain any vertex in $S_{\min}$. The set $T_{\max}$ is *consecutive* if all sinks in $T_{\max}$ are pairwise consecutive. We define consecutiveness for sources in $S_{\min}$ analogously. Because G is a cycle, consecutiveness of $T_{\max}$ also means that $S_{\min}$ is consecutive. If both $S_{\min}$ and $T_{\max}$ are consecutive, we say that γ is *separating*.

Lemma 68. *Let G be an oriented cycle with a level assignment γ. Then G is level planar with respect to γ if and only if γ is separating.*

Proof. For the "if" part, augment G to a planar st-graph as follows. See Figure 9.4 (a) for a cycle and (b) for the augmented st-graph. Let p_t be the oriented path of minimal length that contains all sinks in $T_{\max}$ and no vertex in $S_{\min}$, and let $t_1, t_2 \in T_{\max}$ denote its endpoints. Fix some vertex $s_{\min} \in S_{\min}$ and cancel every source v on p_t by adding an edge from $s_{\min}$ to v. Because $s_{\min} \in S_{\min}$ it is $\gamma(s_{\min}) < \gamma(v)$ and the graph remains acyclic. Introduce a new sink t with $\gamma(t) = \gamma(t_1) + 1$ and cancel every sink v on p_t by adding an edge from v to t. Because $\gamma(v) < \gamma(t)$ the graph remains acyclic. Let p_1 denote an oriented path from t_1 to $s_{\min}$ and let p_2 denote an oriented path from $s_{\min}$ to t_2 so that p_1 and p_2 share no edge. Note that the paths p_t, p_1, p_2 are pairwise disjoint except in common endpoints. Cancel every sink v on p_1 or p_2 by adding an edge from v to t_1 or t_2, respectively. Introduce a new source s with $\gamma(s) = \gamma(s_{\min}) - 1$ and cancel every source v on p_1 or p_2 by adding an edge from s to v. Because $\gamma(s) < \gamma(v)$ the graph remains acyclic. Finally add the edge (s, t) to make the graph an st-graph.

To see that the augmented graph is planar observe that the cycle is trivially planar. Furthermore, all augmentation edges incident to t are incident to vertices on p_t and no augmentation edges incident to s are incident to vertices on p_t (except, possibly, for t_1 and t_2). Moreover, the interior of the circle is partitioned into three regions corresponding to the oriented paths p_t, p_1, p_2. In Figure 9.4 these regions are separated

by the black dashed edges, the area corresponding to p_t is shaded in blue whereas the areas corresponding to p_1 and p_2 are shaded in red. The regions are disjoint and all augmentation edges in one region have the same endpoint, therefore they do not cross. Because p_t, p_1, p_2 are disjoint except in common endpoints the augmentation edges of different areas do not cross. Hence, the augmented graph is a planar st-graph and then Lemma 67 yields that G is level planar with respect to γ. The "only if" part is due to Healy et al. [HKL04, Theorem 7]. □

Recall that any multilevel-planar drawing is a level-planar drawing with respect to some level assignment γ. Lemma 68 states a necessary and sufficient condition for γ so that the drawing is level planar. Given a multilevel assignment ℓ, we therefore find a separating level assignment γ, or determine that no such level assignment exists. It must be $\ell(v) \neq \emptyset$ for all $v \in V$; otherwise, G admits no multilevel drawing. We find a level assignment γ that keeps the sets $S_{\min}$ and $T_{\max}$ as small as possible, because such a level assignment is, intuitively, most likely to be separating. To this end, let $S_{\text{may}} \subset V$ contain each source s' of G with $\min \ell(s') = \min \ell$. Further, let $S_{\text{must}} \subseteq S_{\text{may}}$ contain each source s'' of G with $\ell(s'') = \{\min \ell\}$. Likewise, let $T_{\text{may}} \subset V$ contain each sink t' of G with $\max \ell(t') = \max \ell$ and let $T_{\text{must}} \subseteq T_{\text{may}}$ contain each sink t'' of G with $\ell(t'') = \{\max \ell\}$.

Construct a level assignment ℓ' from ℓ as follows. First, consider the case that both S_{must} and T_{must} are not empty. Let p denote the unique inclusion-minimal path that contains all vertices in S_{must} and no vertex in T_{must}—if no such path exists there exists no separating level assignment, i.e., G is not multilevel planar by Lemma 68. Set $\ell'(s) = \{\min \ell\}$ for all vertices $s \in S_{\text{may}}$ that lie on p and $\ell'(s) = \ell(s) \setminus \{\min \ell\}$ for all vertices $s \in S_{\text{may}}$ that do not lie on p. Next, set $\ell'(t) = \ell(t) \setminus \{\max \ell\}$ for all vertices $t \in T_{\text{may}}$ that lie on p. And set $\ell'(t) = \{\max \ell\}$ for all vertices $t \in T_{\text{may}}$ that do not lie on p. Now consider the case $S_{\text{must}} = \emptyset$. If $S_{\text{must}} = \emptyset$, choose an arbitrary source $s \in S_{\text{may}}$, define p as the path that consists of only s and proceed as in the previous case. The case $T_{\text{must}} = \emptyset$ is symmetric by vertically mirroring. For each remaining vertex v set $\ell'(v) = \ell(v)$. Finally, bring ℓ' into normal form.

We now show that $\ell'(v) \neq \emptyset$ for all $v \in V$. Non-empty intervals are explicitly assigned to all vertices in S_{may} and T_{may}. We are left to show that bringing ℓ' into normal form does not create empty intervals. Changing the upper bound of a source's interval does not affect the intervals of that source's neighbors. The same applies to changing the lower bound of a sink's interval. Increasing the lower bound of a source's interval by one may increase the lower bound of the intervals of that source's neighbors (and, recursively, all vertices dominated by that source). However, because ℓ is in normal form this creates no empty intervals. Likewise, decreasing the upper bound of a sink's interval by one may decrease the upper bound of the intervals of all vertices that dominate that sink but creates no empty intervals. Moreover, there exists

no vertex for whose interval both the lower bound and the upper bound is changed in this way. To see this, observe that the existence of such a vertex is equivalent to the existence of a directed path from a source s with $\ell'(s) = \ell(s) \setminus \{\min \ell\}$, i.e., not on p to a sink t with $\ell'(t) = \ell(t) \setminus \{\max \ell\}$, i.e., on p. Such a directed path cannot exist because p is delimited by sources s_1, s_2 with $\ell'(s_1) = \ell'(s_2) = \{\min \ell\}$ by construction.

Together with the fact that every level assignment that can be obtained from ℓ' is separating by construction and Lemma 68 we conclude the following.

Theorem 25. *Let G be an oriented cycle together with a multilevel assignment ℓ. Then it can be decided in linear time whether G admits a drawing that is multilevel planar with respect to ℓ. Furthermore, if such a drawing exists, it can be computed within the same time.*

9.5 Hardness Results

Recall Theorem 23, which states that multilevel-planarity testing is in general NP-complete. Theorem 23 is a direct consequence of the fact that multilevel-planarity is a generalization of upward-planarity testing, which is known to be NP-complete [GT01]. We now show that multilevel-planarity testing is NP-complete even in very restricted cases, namely for sT-graphs without a fixed combinatorial embedding, for oriented trees and for embedded multi-source graphs with at most two possible levels for each vertex.

9.5.1 sT-Graphs with Variable Embedding

In Section 9.3, we showed that testing multilevel planarity of embedded sT-graphs is feasible in linear time, because for every inner sink there is a unique sink switch to cancel it with. We now show that dropping the requirement that the combinatorial embedding is fixed makes multilevel-planarity testing NP-hard. To this end, we reduce the Scheduling With Release Times And Deadlines (Srtd) problem, which is strongly NP-complete [GJ77], to multilevel-planarity testing. An instance of this scheduling problem consists of a set of tasks $T = \{t_1, \dots, t_n\}$ with individual release times $r_1, \dots, r_n \in \mathbb{N}_+$, deadlines $d_1, \dots, d_n \in \mathbb{N}_+$ and processing times $p_1, \dots, p_n \in \mathbb{N}_+$ for each task so that $\sum_{i=1}^{n} p_i$ is bounded by a polynomial in n. See Figure 9.5 (a) for an example. The question is whether there is a non-preemptive schedule $\sigma : T \to \mathbb{N}$ that specifies the start time for each task, such that for each $i \in \{1, \dots, n\}$ we get (1) $\sigma(t_i) \geq r_i$, i.e., no task starts before its release time, (2) $\sigma(t_i) + p_i \leq d_i$, i.e., each task finishes before its deadline, and

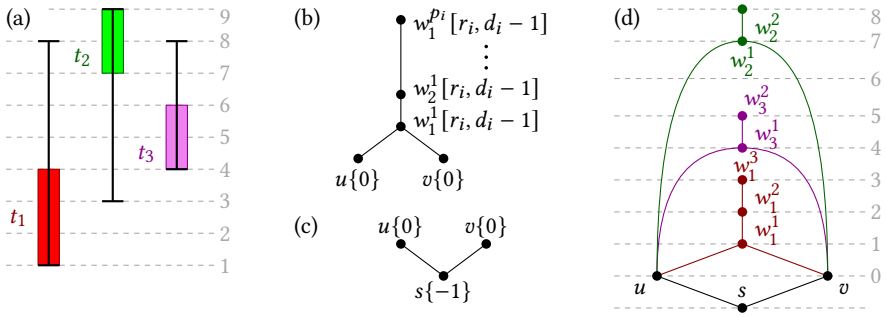


Figure 9.5: An instance of the SRTD problem (a) with tasks $T = \{t_1, t_2, t_3\}$, release times $r_1 = 1$, $r_2 = 3$, $r_3 = 4$, deadlines $d_1 = 8$, $d_2 = 9$, $d_3 = 8$ and processing times $p_1 = 3$, $p_2 = p_3 = 2$. A task gadget (b) for each task and one base gadget (c) that provides the single source are used to turn the SRTD instance (a) into a multilevel-planarity testing instance (d).

(3) $\sigma(t_i) < \sigma(t_j) \Rightarrow \sigma(t_i) + p_i \leq \sigma(t_j)$ for any $j \in \{1, \ldots, n\} \setminus \{i\}$, i.e., no two tasks are executed at the same time.

Create for every task $t_i \in T$ a *task gadget* $\mathcal{T}_i$ that consists of two vertices u, v together with a directed path $P_i = (w_i^1, w_i^2, \ldots, w_i^{p_i})$ of length $p_i - 1$; see Figure 9.5 (b). For each vertex w_i^j on P_i set $\ell(w_i^j) = [r_i, d_i - 1]$, i.e., all possible points of time at which this task can be executed. Set $\ell(u) = \ell(v) = \{0\}$. Join all task gadgets with a *base gadget*. The base gadget consists of three vertices s, u, v and two edges (s, u), (s, v), where u is placed to the left of v; see Figure 9.5 (c). Set $\ell(s) = \{-1\}$ and, again, set $\ell(u) = \ell(v) = \{0\}$. Identify the common vertices u and v of all gadgets; see Figure 9.5 (d). Because SRTD is strongly NP-complete, the size of the resulting graph is polynomial in the size of the input. The idea of the construction is that because the task gadgets may not cross in a planar drawing and because their common vertices u and v are identified, they are stacked on top of each other, inducing a valid schedule of the associated tasks. Contrasting linear-time tests of upward planarity and level planarity for sT-graphs we show the following.

Theorem 26. *Testing sT-graphs for multilevel planarity is NP-complete.*

Proof. Clearly, the problem is in NP. We reduce SRTD to multilevel-planarity testing. To this end, we show that the graph G as described above is multilevel planar if and only if there is a valid one-processor schedule for the SRTD instance. To see this, start with a valid schedule σ. Define a level assignment γ as follows. Start by setting $\gamma(s) = -1$, $\gamma(u) = \gamma(v) = 0$. And for $1 \leq i \leq n$ and $1 \leq j \leq p_i$,

set $\gamma(w_i^j) = \sigma(t_i) + j$. Since σ is non-preemptive, it induces a total order on the tasks, without loss of generality $\sigma(t_1) < \ldots < \sigma(t_n)$. Order the edges to the task gadgets $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ from right to left at u, and from left to right at v. Observe that any sink of G is the endpoint of a directed path of a task gadget. For $1 \leq i < n$ cancel the sink $w_i^{p_i}$ by connecting it to w_{i+1}^1. This is possible because the schedule is valid. Then Lemma 67 implies that there exists a drawing of G that is level-planar with respect to γ. Because it is $\gamma(v) \in \ell(v)$ for all $v \in V$ by construction, G is multilevel planar with respect to ℓ.

For the reverse direction, consider a drawing Γ of G that is multilevel planar with respect to ℓ. Let γ denote the level assignment induced by Γ. Further, let π be the permutation of $\{1, 2, \ldots, n\}$ so that the counter-clockwise order of edges around u is $(s, u), (u, w_{\pi(1)}^1), (u, w_{\pi(2)}^1), \ldots, (u, w_{\pi(n)}^1)$. For $1 \leq i < n$ let $j = \pi(i)$ and $j' = \pi(i+1)$. The vertices $w_j^{p_j}$ and $w_{j'}^1$ are incident to a common face f. Note that G is an sT-graph and because s is not incident to f it is an inner face, i.e., Property 2 of Lemma 64 applies. Because $w_j^{p_j}$ is incident to only one edge it is $\angle_{\Gamma,f}(w_j^{p_j}) > \pi$. Because $w_j^{p_j}$ and $w_{j'}^1$ are the only sink switches of f, we find that $\angle_{\Gamma,f}(w_{j'}^1) < \pi$, i.e., $w_{j'}^1$ is the vertex with the greatest y-coordinate among all vertices incident to f. In particular, it is $\gamma(w_j^{p_j}) < \gamma(w_{j'}^1)$. For $1 \leq i \leq n$ and $j = \pi(i)$ set $\sigma(t_j) = \gamma(w_i^1)$. For $j = \pi(i)$ with $i < n$ it is $\sigma(t_j) + p_j < \sigma(t_{j'})$. Moreover, $\sigma(t_j) \geq r_j$ and $\sigma(t_j) + p_j \leq d_j$ is ensured by the multilevel assignment. Hence, σ is a valid schedule. □

9.5.2 Oriented Trees

We can show NP-completeness of oriented trees with a very similar reduction as for sT-graphs without a fixed combinatorial embedding. As in Section 9.5.1 we reduce from Scheduling with Release Times and Deadlines, the required gadgets are only slightly different. Let $T = \{t_1, \ldots, t_n\}, r_1, \ldots, r_n, d_1, \ldots, d_n$ and $p_1, \ldots, p_n$ be such an instance with $\sum_{i=1}^n p$ bounded by a polynomial in n. Again we initialize G with the base gadget shown in Figure 9.6 (a) and for each task we add one task gadget as shown in Figure 9.6 (b). The base gadget consists of two vertices u and v on level 0 both connected to one vertex c_1 on level 1, which in turn is connected to the final vertex c_2 on level $d_{\max} + 1$, where $d_{\max} = \max_{i \in \{1,\ldots,n\}} d_i$ is the maximum deadline among all tasks. The task gadget is the same as in the previous section, except that v is replaced by a separate vertex a_i per gadget. The base gadget and all task gadgets share one common vertex u, which is identified in G. The resulting graph G is a tree and because Srtd is strongly NP-complete the size of G is polynomial in the size of the Srtd instance.

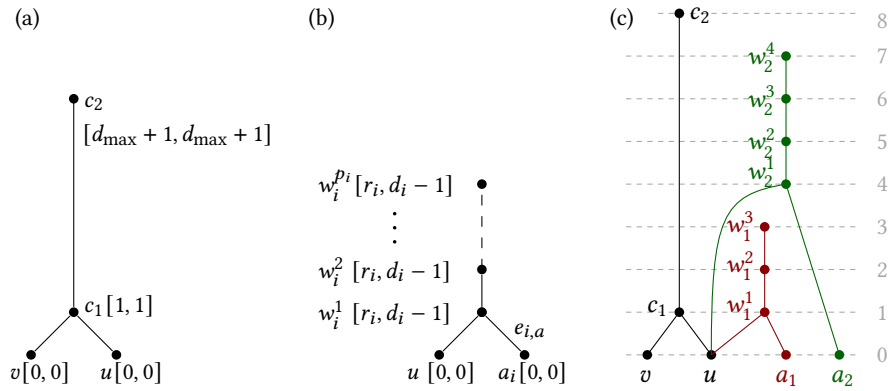


Figure 9.6: The base (a) and the task gadget (b) to transform a SRTD instance into a multilevel-planarity testing instance with G being a tree. An example (c) with two task gadgets (in red and green).

Theorem 27. *Testing oriented trees for multilevel planarity is NP-complete.*

Proof. The proof is very similar to the proof of Theorem 26. Clearly, the problem is in NP. Again, we reduce SRTD to multilevel-planarity testing. We show that the graph G as described above is a multilevel planar if and only if there is a valid one-processor schedule for the SRTD instance. To see this, start with a valid schedule σ. Define a level assignment γ as follows. Set $\gamma(u) = \gamma(v) = 0$ and for $i \in \{1, 2, \dots, n\}$ set $\gamma(a_i) = \gamma(b_i) = 0$. Moreover, set $\gamma(c_1) = 1$ and $\gamma(c_2) = d_{\max} + 1$. And for $1 \leq i \leq n$ and $1 \leq j \leq p_i$ set $\gamma(w_i^j) = \sigma(t_i) + j$. Since σ is non-preemptive it induces a total order on the tasks, without loss of generality $\sigma(t_1) < \sigma(t_2) < \cdots < \sigma(t_n)$. Place $v, u, a_1, a_2, \dots, a_n$ in this order from left to right on level 0. Connect the edges incident to u from left to right in the order $(u, c_1), (u, w_n^1), (u, w_{n-1}^1), \dots, (u, w_1^1)$. Observe that any sink of G except for c_2 is the endpoint of a directed path of a task gadget. For $1 \leq i < n$ cancel the sink $w_i^{p_i}$ by connecting it to w_{i+1}^1. This is possible because the schedule is valid. Cancel the sink $w_n^{p_n}$ by connecting it to c_2. Create a new vertex s and cancel all sources by connecting s to them. Then Lemma 67 implies that there exists a drawing of G that is level-planar with respect to γ. Since it is $\gamma(v) \in \ell(v)$ for all $v \in V$ by construction, G is multilevel planar with respect to ℓ.

For the reverse direction, consider a drawing Γ of G that is multilevel planar with respect to ℓ. Obtain an sT-graph G' together with a multilevel planar drawing Γ' from G and Γ by adding a new vertex s on level -1 and connecting it to u, v and all a_i. Let γ denote the level assignment induced by Γ'. Further, let π be the permutation of $\{1, 2, \dots, n\}$ so that the counter-clockwise order of edges around u

is $(u, c_1), (u, w^1_{\pi(1)}), (u, w^1_{\pi(2)}), \ldots, (u, w^1_{\pi(n)})$. For $1 \leq i < n$ define $j = \pi(i)$ and further $j' = \pi(i+1)$. The vertices $w^{p_j}_j$ and $w^1_{j'}$ are incident to a common face f of Γ'. Because $\gamma(c_2) = \max \gamma$ the edge (c_1, c_2) implies that all vertices a_i for $1 \leq i \leq n$ lie on the same side of u on level 0 and v lies on the other side. This means that f is an inner face. Together with the fact that G' is an sT-graph this means that Property 2 of Lemma 64 applies. Because $w^{p_j}_j$ is incident to only one edge it is $\angle_{\Gamma',f}(w^{p_j}_j) > \pi$. Because $w^{p_j}_j$ and $w^1_{j'}$ are the only sink switches of f it is $\angle_{\Gamma',f}(w^1_{j'}) < \pi$, i.e., $w^1_{j'}$ is the vertex with the greatest y-coordinate among all vertices incident to f. In particular, it is $\gamma(w^{p_j}_j) < \gamma(w^1_{j'})$. For $1 \leq i \leq n$ and $j = \pi(i)$ set $\sigma(t_j) = \gamma(w^1_i)$. For $j = \pi(i)$ with $i < n$ it is $\sigma(t_j) + p_j < \sigma(t_{j'})$. Moreover, $\sigma(t_j) \geq r_j$ and $\sigma(t_j) + p_j \leq d_j$ is ensured by the multilevel assignment. Hence, σ is a valid schedule. □

This also contrasts the results for upward planarity and level planarity, because every oriented tree is upward planar and all level graphs can be tested for level planarity in linear time.

9.5.3 Embedded Multi-Source Graphs

We show that multilevel-planarity testing for embedded graphs with multiple sources is NP-complete by reducing from PLANAR MONOTONE 3-SAT, which is known to be NP-complete [dK12]. An instance $\mathcal{I} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$ of this problem consists of a set of variables $\mathcal{V}$, a set of clauses $\mathcal{C}$ and a planar drawing $\mathcal{E}$ of a so-called *variable-clause graph*, and it obeys the following restrictions. Each clause consists of at most three literals and it is *monotone*, i.e., it is either positive or negative, meaning that it consists of either only positive or only negative literals, respectively. We also assume that $\mathcal{C}$ contains at least one positive and at least one negative clause. The *variable-clause graph* consists of the nodes $\mathcal{V} \cup \mathcal{C}$. Two nodes are connected by an undirected arc if one of the nodes is a clause and the other node is a variable that appears as a literal in the clause. The drawing $\mathcal{E}$ of the variable-clause graph is such that all variables lie on a horizontal straight line, positive and negative clauses are drawn as horizontal line segments with integer y-coordinates below and above that line, respectively, and arcs connecting clauses and variables are drawn as non-intersecting vertical line segments; see Figure 9.7. We call $\mathcal{E}$ a *planar rectilinear drawing*.

Transform the variable-clause graph and its rectilinear drawing $\mathcal{E}$ into a multilevel graph by replacing each positive or negative clause with a positive or negative clause gadget and identifying common vertices (namely those vertices that are variables). The drawing $\mathcal{E}$ directly induces a combinatorial embedding and an outer face of the multilevel graph obtained in this way. Figure 9.8 (a) shows the gadget for a positive clause $(x_a \vee x_b \vee x_c)$. The vertices x_a, x_b and x_c are variables in $\mathcal{V}$. We call vertex p_i the *pendulum*. A variable $x \in \mathcal{V}$ is set to true (false) if it lies on level 1 (level 0). In a

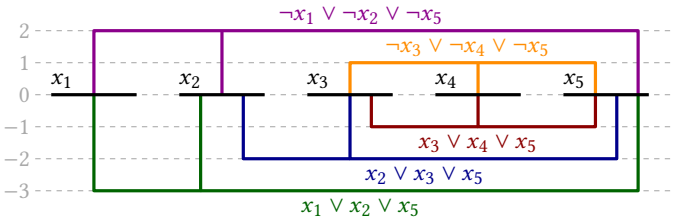


Figure 9.7: A planar monotone 3-SAT instance together with a rectilinear embedding of its variable-clause graph.

positive clause gadget p_i must lie on level 0. The idea is that it forces one variable to lie on level 1, i.e., be set to true. This is achieved by arranging the variables x_a, x_b, x_c together with auxiliary vertices $v_{i,1}, v_{i,2}, v_{i,3}$ on a cycle $v_{i,1}, x_a, v_{i,2}, x_b, v_{i,3}, x_c, v_{i,1}$. This cycle encloses an inner face, the edge $(v_{i,1}, p_i)$ connects the pendulum to the cycle and the fixed embedding around $v_{i,1}$ ensures that the pendulum lies within this face. The gadget for a negative clause $(\neg x_a \vee \neg x_b \vee \neg x_c)$ works symmetrically; the idea is that its pendulum forces one variable to lie on level 0, i.e., be set to false; see Figure 9.8 (b). To obtain the multilevel graph, replace each positive or negative clause with a positive or negative clause gadget and identify common vertices, namely those vertices that are variables. Figure 9.9 shows the multilevel graph obtained from the planar monotone 3-SAT instance shown in Figure 9.7. In order for this graph to be multilevel planar, it must be possible to place the vertices that are variables on of the two possible levels so that all pendulums can be embedded within their gadgets, i.e., all clauses are satisfied.

Theorem 28. *Testing embedded graphs for multilevel planarity is NP-complete, even when restricted to multilevel assignments ℓ with $|\ell(v)| \leq 2$ for each vertex v.*

Proof. Clearly, the problem is in NP. Reduce planar monotone 3-SAT to multilevel-planarity testing by showing that the multilevel graph G derived from $(\mathcal{V}, \mathcal{C}, \mathcal{E})$ is multilevel planar if and only if $(\mathcal{V}, \mathcal{C}, \mathcal{E})$ is satisfiable.

Suppose that φ is a satisfying truth assignment of the 3-SAT instance underlying $(\mathcal{V}, \mathcal{C}, \mathcal{E})$. Construct a drawing Γ of G that is multilevel planar with respect to ℓ by constructing a level assignment γ as follows. Let $v \in \mathcal{V}$ be a variable. Recall that v is a vertex in G. If $\varphi(v) = \text{true}$, set $\gamma(v) = 1$. Otherwise, set $\gamma(v) = 0$. Let $c_i \in \mathcal{C}$ be a positive clause. Draw the pendulum p_i of c_i below a vertex v_i with $\varphi(v_i) = \text{true}$. Because φ is a satisfying truth assignment such a v_i exists. Now let $c_j \in \mathcal{C}$ be a negative clause. Draw the pendulum p_j of c_j above a vertex v_j with $\varphi(v_j) = \text{true}$. Since c_j is a negative clause, a positive literal in c_j corresponds to a variable set to

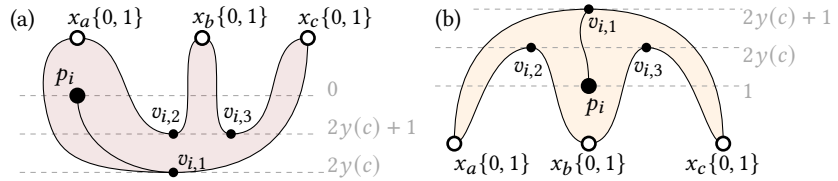


Figure 9.8: Gadgets for the clauses $(x_a \vee x_b \vee x_c)$ (a) and $(\neg x_a \vee \neg x_b \vee \neg x_c)$ (b).

false, and because φ is a satisfying truth assignment such a v_j exists. The resulting drawing is then level planar with respect to γ and therefore multilevel planar with respect to ℓ.

Now assume that Γ is a combinatorially equivalent drawing of G and is multilevel planar with respect to ℓ. Due to the construction rules and because $\mathcal{C}$ contains at least one positive clause and one negative clause there exists exactly one face that is incident to both the vertex $v_{\min}$ with $\ell(v_{\min}) = \min \ell$ and the vertex $v_{\max}$ with $\ell(v_{\max}) = \max \ell$. This face must be the outer face; see Figure 9.9. Let γ denote the level assignment induced by Γ. Construct a truth assignment φ as follows: Set the variable $v \in \mathcal{V}$ to true or false depending on whether it is $\gamma(v) = 1$ or $\gamma(v) = 0$, respectively. Because it is $\ell(v) = \{0, 1\}$, this always assigns a truth value to v. Consider the pendulum p_i of a positive clause $c_i \in \mathcal{C}$. In a positive gadget, p_i forces one of the variables in c_i, say v_i, to level 1, i.e., $\varphi(v_i) = \text{true}$. Because c_i is a positive clause it is then satisfied. In a negative gadget for a negative clause $c_j \in \mathcal{C}$, pendulum p_j forces one of the variables in c_j, say v_j, to level 0, i.e., $\varphi(v_j) = \text{false}$. Because c_j is a negative clause, it is then satisfied. Hence, φ is a satisfying truth assignment of $(\mathcal{V}, \mathcal{C}, \mathcal{E})$. □

9.6 Conclusion

We introduced and studied multilevel planarity, a generalization of both upward planarity and level planarity. We started by giving a linear-time algorithm to decide multilevel planarity of embedded sT-graphs. The correctness proof of this algorithm uses insights from both upward planarity and level planarity. In contrast to this, we showed that deciding the multilevel planarity of sT-graphs without a fixed embedding is NP-complete. We also gave a linear-time algorithm to decide multilevel planarity of oriented cycles, which is interesting because the existence of multiple sources makes many related problems NP-complete, e.g., testing upward planarity, partial level planarity or ordered level planarity. This positive result is contrasted by the fact that multilevel-planarity testing is NP-complete for oriented trees. Whether multilevel-planarity testing becomes tractable for trees with a given combinatorial

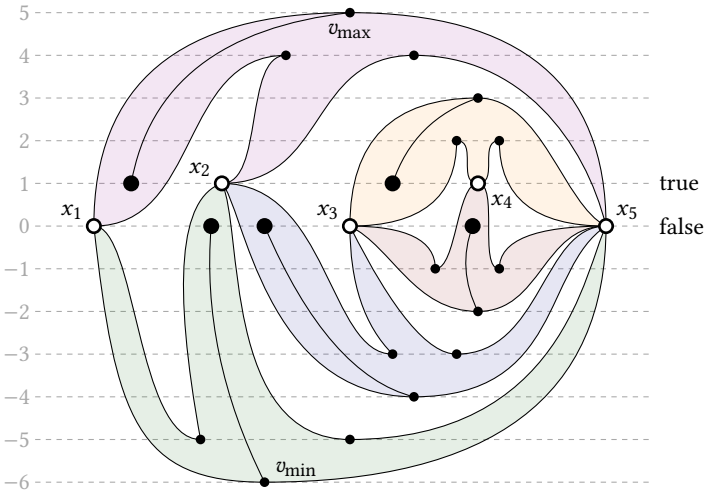


Figure 9.9: A multilevel-planar drawing of the graph constructed from the planar monotone 3-SAT instance shown in Figure 9.7. The shaded faces correspond to the gadgets that substitute the clauses. In this multilevel-planar drawing, vertices x_1, x_3 and x_5 are on level 0 so variables $x_1 = x_3 = x_5 =$ false. On the other hand vertices x_2 and x_4 are on level 1 so variables $x_2 = x_4 =$ true.

embedding remains an open question. Deciding multilevel planarity remains NP-complete for embedded multi-source graphs where each vertex is assigned either to exactly one level, or to one of two adjacent levels. This contrasts the existence of efficient algorithms for testing upward planarity and level planarity of embedded multi-source graphs. The following table summarizes our results for multilevel planarity and relates them to existing results for upward planarity and level planarity.

	fixed combinatorial embedding		
	st-Graphs	*sT*-Graphs	arbitrary
Upward Planarity	$O(1)$ [BDLM94]	$O(n)$ [BDLM94]	P [BDLM94]
Multilevel Planarity	$O(1)$ (Corollary 7)	$O(n)$ (Theorem 24)	NPC (Theorem 28)
Level Planarity	$O(1)$ [JL02]	$O(n)$ [JL02]	$O(n)$ (Chapter 3)
	not embedded		
	Cycles	*sT*-Graphs	Trees
Upward Planarity	$O(n)$ [BDMT98]	$O(n)$ [BDMT98]	$O(1)$ [DETT99]
Multilevel Planarity	$O(n)$ (Theorem 25)	NPC (Theorem 26)	NPC (Theorem 27)
Level Planarity	$O(n)$ [JL02]	$O(n)$ [JL02]	$O(n)$ [JL02]

10 Level-Planar Drawings with Few Slopes

We introduce and study level-planar straight-line drawings with a fixed number λ of slopes. For proper level graphs (all edges connect vertices of adjacent levels), we give an $O(n \log^2 n / \log \log n)$-time algorithm that either finds such a drawing or determines that no such drawing exists. Moreover, we consider the partial drawing extension problem, where we seek to extend an immutable drawing of a subgraph to a drawing of the whole graph, and the simultaneous drawing problem, which asks about the existence of drawings of two graphs whose restrictions to their shared subgraph coincide. We present $O(n^{4/3} \log n)$-time and $O(\lambda n^{10/3} \log n)$-time algorithms for these respective problems on proper level-planar graphs.

We complement these positive results by showing that testing whether non-proper level graphs admit level-planar drawings with λ slopes is NP-hard even in restricted cases.

This chapter extends work initiated as part of Nadine Davina Krisam's bachelor's thesis [Kri18]. It is based on joint work with Nadine Davina Krisam and Tamara Mchedlidze [BKM19].

10.1 Introduction

Directed graphs explaining hierarchy naturally appear in multiple industrial and academic applications. Some examples include PERT diagrams, UML component diagrams, text edition networks, text variant graphs [Jän+15], phylogenetic and

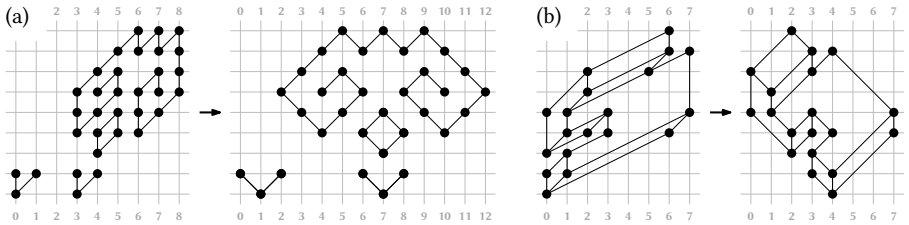


Figure 10.1: Shearing drawings to change the slopes. In (a), the left drawing with slopes 0 and 1 is transformed into the right orthogonal drawing, i.e., one with slopes -1 and 1. In (b), the left drawing with slopes 0, 1 and 2 is transformed into a drawing with slopes -1, 0 and 1.

neural networks. In these, and many other applications, it is essential to visualize the implied directed graph so that the viewer can perceive the hierarchical structure it contains. By far the most popular way to achieve this is to apply the *Sugiyama framework* – a generic network visualization algorithm that results in a drawing where each vertex lies on a horizontal line, called *layer*, and each edge is directed from a lower layer to a higher layer [HN13].

The Sugiyama framework consists of several steps: elimination of directed cycles in the initial graph, assignment of vertices to layers, vertex ordering and coordinate assignment. During each of these steps several criteria are optimized, by leading to more readable visualizations, see e.g. [HN13]. In this chapter we concentrate on the last step of the framework, namely coordinate assignment. Thus, the subject of our study are level graphs. A drawing is *straight-line* if the edges are straight-line segments. In this chapter, we refer to G together with a level-planar embedding to as *embedded level graph* $\mathcal{G}$. The third step of Sugiyama framework, vertex ordering, results in an embedded level graph.

The general goal of the coordinate assignment step is to produce a final visualization while further improving its readability. The criteria of readability that have been considered in the literature for this step include straightness and steepness of the edges [HN13]. Here we study the problem of coordinate assignment step with bounded number of slopes. The *slope* of an edge (u, v) in Γ is defined as $(\Gamma(v) - \Gamma(u))/(\ell(v) - \ell(u))$. For proper level graphs it is $\ell(v) - \ell(u) = 1$, the slope of (u, v) is then simply $\Gamma(v) - \Gamma(u)$. We restrict our study to drawings in which all slopes are non-negative; such drawings can be transformed into drawings with negative slopes by shearing; see Fig. 10.1. A level drawing Γ is a *λ-slope drawing* if all slopes in Γ appear in the set $\{0, 1, \ldots, \lambda - 1\}$.

We study embedding-preserving straight-line level-planar λ-slope drawings, or

λ-*drawings* for short and refer to the problem of finding these drawings as λ-Drawability. Since the possible edge slopes in a λ-drawing are integers all vertices lie on the integer grid.

Related Work. The number of slopes used for the edges in a graph drawing can be seen as an indication of the simplicity of the drawing. For instance, the measure *edge orthogonality*, which specifies how close a drawing is to an *orthogonal drawing*, where edges are polylines consisting of horizontal and vertical segments only, has been proposed as a measure of aesthetic quality of a graph drawing [Pur02]. In a similar spirit, Kindermann et al. studied the effect reducing the segment complexity on the aesthetics preference of graph drawings and observed that in some cases people prefer drawings using lower segment complexity [KMS18]. More generally, the use of few slopes for a graph drawing may contribute to the formation of "Prägnanz" ("good figure" in German) of the visualization, that accordingly to the Gestalt law of Prägnanz, or law of simplicity, contributes to the perceived quality of the visualizations. This design principle often guides the visualization of metro maps. See [Nöl14] for a survey of the existing approaches, most of which generate octilinear layouts of metro maps, and [NN20] for a recent model for drawing more general k-linear metro maps.

Level-planar drawing with few slopes have not been considered in the literature but drawings of undirected graphs with few slopes have been extensively studied. The *planar slope number* of a planar graph G is the smallest number s so that G has a planar straight-line drawing with edges of at most s distinct slopes. Special attention has been given to proving bounds on the (planar) slope number of undirected graph classes [BMW06, GLM15, DESW07, DSW07, GLM18, KPPT08, KPP13, KMW14, LLMN13, PP06]. Determining the planar slope number is hard in the existential theory of reals [Hof17]. The slope number has also been studied for *upward* planar drawings, that is, drawings of directed graphs where each edge is drawn as a y-monotone curve (but, unlike with level planarity, the y-coordinate of the vertices is not prescribed) [GLM20, Bek+18].

Several graph visualization problems have been considered in the partial and simultaneous settings. In the *partial drawing extension* problem, one is presented with a graph and an immutable drawing of some subgraph thereof. The task is to determine whether the given drawing of the subgraph can be completed to a drawing of the entire graph. The problem has been studied for the planar setting [MNR16, Pat06] and also the level-planar setting [BR17]. In the *simultaneous drawing* problem, one is presented with two graphs that may share some subgraph. The task is to draw both graphs so that the restrictions of both drawings to the shared subgraph are identical. We refer the reader to [BKR13] for an older literature overview. The problem has been considered for orthogonal drawings [Ang+16] and level-planar

drawings [Ang+20]. Up to our knowledge, neither partial nor simultaneous drawings have been considered in the restricted slope setting.

Contribution. We introduce and study the λ-Drawability problem. To solve this problem for proper level graphs, we introduce two models. In Section 10.3 we describe the first model, which uses a classic integer-circulation-based approach. This model allows us to solve the λ-Drawability in $O(n \log^3 n)$ time and obtain a λ-drawing within the same running time if one exists. In Section 10.4, we describe the second distance-based model. It uses the duality between flows in the primal graph and distances in the dual graph and allows us to solve the λ-Drawability in $O(n \log^2 n / \log\log n)$ time.

We also address the partial and simultaneous settings. The classic integer-circulation-based approach can be used to extend connected partial λ-drawings in $O(n \log^3 n)$ time. In Section 10.5, we build on the distance-based model to extend not-necessarily-connected partial λ-drawings in $O(n^{4/3} \log n)$ time, and to obtain simultaneous λ-drawings in $O(\lambda n^{10/3} \log n)$ time if they exist.

We complement these algorithmic results in Section 10.6 with a proof that 2-Drawability is NP-hard even for biconnected graphs where all edges have length one or two, and then finish with some concluding remarks in Section 10.7.

10.2 Preliminaries

Let Γ be a level-planar drawing of an embedded level-planar graph $\mathcal{G}$. The *width* of Γ is defined as $\max_{v \in V} \Gamma(v) - \min_{v \in V} \Gamma(v)$. An integer $\bar{x}$ is a *gap* in Γ if it is $\Gamma(v) \neq \bar{x}$ for all $v \in V$, $\Gamma(v_1) < \bar{x}$ and $\Gamma(v_2) > \bar{x}$ for some $v_1, v_2 \in V$, and $\Gamma(u) < \bar{x} < \Gamma(v)$ for no edge $(u, v) \in E$. For example, 2 is a gap in the left drawing in Fig. 10.1 (a). A drawing Γ is *compact* if it has no gap, e.g., the left drawing in Fig. 10.1 (b). Note that a λ-drawing of a connected level-planar graph is inherently compact. In a λ-drawing of a non-connected level-planar graph every gap can be eliminated by a horizontal shift. The fact that we only need to consider compact λ-drawings helps us to limit the drawing width. In particular, a compact λ-drawing has width at most $(\lambda - 1)(n - 1)$.

Let u and v be two vertices on the same level i. With $[u, v]_{\mathcal{G}}$ (or $[u, v]$ when $\mathcal{G}$ is clear from the context) we denote the set of vertices that contains u, v and all vertices in between u and v on level i in $\mathcal{G}$. We say that two vertices u and v are *consecutive in* $\mathcal{G}$ when $[u, v] = \{u, v\}$. Two edges $e = (u, w), e' = (v, x)$ are *consecutive in* $\mathcal{G}$ when the only edges with one endpoint in $[u, v]_{\mathcal{G}}$ and the other endpoint in $[w, x]_{\mathcal{G}}$ are e and e'. For example, in Fig. 10.2 (b), the vertices u and v are consecutive, the vertices x and w are consecutive, and the edges (u, w) and (v, x) are consecutive; and in Fig. 10.2 (c), the vertices v_{left} and v_{right} are not consecutive.

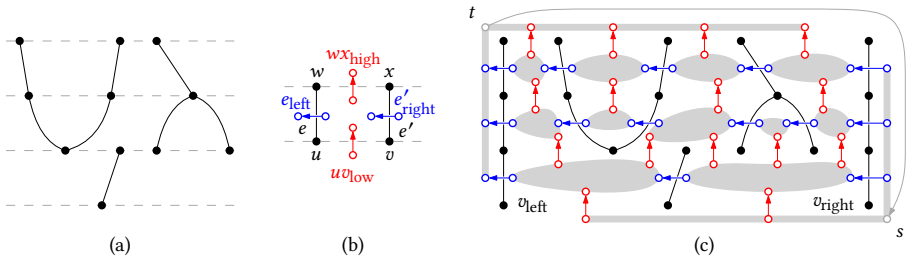


Figure 10.2: An embedded level graph $\mathcal{G}$ (a). The definition of the arcs of the flow network (b). The graph $\mathcal{G}$ together with the paths p_{left} and p_{right} in black (c). The resulting flow network $F_{\mathcal{G}}^{\lambda}$ consists of the blue slope arcs and the red space arcs, its nodes are formed by merging the nodes in the gray areas. The red space arcs have a demand of 1 and a capacity of $(\lambda - 1)(n - 1)$ and the blue slope arcs have a demand of zero and a capacity of $\lambda - 1$.

A *flow network* $F = (N, A)$ consists of a set of nodes N connected by a set of directed arcs A. A node is a *source* if it has no incoming arcs and it is a *sink* if it has no outgoing arcs. A flow network is an *st-graph* if it has exactly one source and exactly one sink. Each arc has a *demand* specified by a function $d : A \to \mathbb{N}_0$ and a *capacity* specified by a function $c : A \to \mathbb{N} \cup \{\infty\}$ where ∞ means unlimited capacity. A *circulation* in F is a function $\varphi : A \to \mathbb{N}_0$ that assigns an integral flow to each arc of F and satisifies the two following conditions. First, the circulation has to respect the demands and capacities of the arcs, i.e., for each arc $a \in A$ it is $d(a) \leq \varphi(a) \leq c(a)$. Second, the circulation has to respect flow conservation, i.e., for each node $v \in N$ it is $\sum_{(u,v)\in A} \varphi(u, v) = \sum_{(v,u)\in A} \varphi(v, u)$. Depending on the flow network no circulation may exist.

10.3 Flow Model

In this section, we model the λ-Drawability as a problem of finding a circulation in a flow network. Let $\mathcal{G}$ be an embedded proper k-level graph. As a first step, we add two directed paths p_{left} and p_{right} that consist of one vertex on each level from 1 to k to $\mathcal{G}$. Insert p_{left} and p_{right} into $\mathcal{G}$ to the left and right of all other vertices as the *left* and *right boundary*, respectively. See Fig. 10.2 (a) and (c). From now on, we assume that $\mathcal{G}$ contains the left and right boundary.

The flow network $F_{\mathcal{G}}^{\lambda}$ consists of nodes and arcs and is similar to a directed dual of $\mathcal{G}$ with the difference that it takes the levels of $\mathcal{G}$ into account. In particular, for

every edge e of $\mathcal{G}$, $F^\lambda_\mathcal{G}$ contains two nodes e_{left} and e_{right}, in the left and the right faces incident to e, and a dual *slope arc* $e^\star = (e_{\text{right}}, e_{\text{left}})$ with demand 0 and capacity $\lambda - 1$; see the blue arcs in Fig. 10.2 (b) and (c). The flow across $e^\star$ determines the slope of e. Additionally, for every pair of consecutive vertices u, v we add two nodes $[u,v]_{\text{low}}$ and $[u,v]_{\text{high}}$ to $F^\lambda_\mathcal{G}$ and connect them by a *space arc* $[u,v]^\star$; see the red arcs in Fig. 10.2 (b) and (c). The flow across $[u,v]^\star$ determines the space between u and v. The space between u and v needs to be at least one to prevent u and v from colliding and can be at most $(\lambda-1)(n-1)$ due to the restriction to compact drawings. So, assign to $[u,v]^\star$ a demand of one and a capacity of $(\lambda-1)(n-1)$. To obtain the final flow network we merge certain nodes. Let $e = (u, w)$ and $e' = (v, x)$ be consecutive edges. Merge the nodes $e_{\text{right}}, e'_{\text{left}}$, the nodes $\{\{u', v'\}_{\text{high}} : \forall u', v' \text{ consecutive in } [u,v]\}$ and the nodes $\{\{w', x'\}_{\text{low}} : \forall w', x' \text{ consecutive in } [w,x]\}$ into a single node. Next, merge all remaining source and sink nodes into one source node s and one sink node t, respectively. See Fig. 10.2 (c), where the gray areas touch nodes that are merged into a single node. Observe that flow network is a connected st-graph. Clearly s is a source and t is a sink. Each remaining node v corresponds to two consecutive edges of $\mathcal{G}$, so by construction it has exactly one incoming and one outgoing slope arc, so v is neither a source nor a sink. This also implies that there exists a directed path of slope arcs from s to v, and a directed path of slope arcs from v to t, so the flow network is connected. Finally, to admit non-trivial circulations, insert an arc from t to s with capacity ∞. Observe that $F^\lambda_\mathcal{G}$ is planar by its construction based on the planar embedding of $\mathcal{G}$.

The network $F^\lambda_\mathcal{G}$ is designed in such a way that the circulations in $F^\lambda_\mathcal{G}$ correspond bijectively to the λ-drawings of $\mathcal{G}$. Let Γ be a drawing of $\mathcal{G}$ and let x be the function that assigns to each vertex of $\mathcal{G}$ its x-coordinate in Γ. We define a dual circulation $x^\star$ as follows. Recall that each arc a of $F^\lambda_\mathcal{G} - (s,t)$ is a slope arc or a space arc. If a is a slope arc it is dual to an edge (u, w) of $\mathcal{G}$. Then define $x^\star(a) := x(w) - x(u)$. If a is a space arc it is dual to consecutive vertices u, v of $\mathcal{G}$, where u appears left of v. Then define $x^\star(a) := x(v) - x(u)$. We remark the following, although we defer the proof to the next section.

Lemma 69. *Let $\mathcal{G}$ be an embedded proper level-planar graph together with a λ-drawing Γ. The dual $x^\star$ of the function x that assigns to each vertex of $\mathcal{G}$ its x-coordinate in Γ is a circulation in $F^\lambda_\mathcal{G}$.*

In the reverse direction, given a circulation φ in $F^\lambda_\mathcal{G}$ we define a dual function $\varphi^\star$ that, when interpreted as assigning an x-coordinate to the vertices of $\mathcal{G}$, defines a λ-drawing of $\mathcal{G}$. Refer to the level-1-vertex of p_{right} as v_{right}. Start by setting $\varphi^\star(v_{\text{right}}) = 0$, i.e., the x-coordinate of v_{right} is 0. Process the remaining vertices of the right boundary in ascending order with respect to their levels. Let (u, v) be an edge of the right

boundary so that u has already been processed and v has not been processed yet. Then set $\varphi^\star(v) = \varphi^\star(u) + \varphi((u,v)^\star)$, where $(u,v)^\star$ is the slope arc dual to (u,v). Let w, x be a pair of consecutive vertices so that x has already been processed and w has not yet been processed yet. Then set $\varphi^\star(w) = \varphi^\star(x) - \varphi([w,x]^\star)$, where $[w,x]^\star$ is a space arc. It turns out that $\varphi^\star$ defines a λ-drawing of $\mathcal{G}$.

Lemma 70. *Let $\mathcal{G}$ be an embedded proper level-planar graph, let $\lambda \in \mathbb{N}$ and let φ be a circulation in $F^\lambda_\mathcal{G}$. Then the dual $\varphi^\star$, when interpreted as assigning an x-coordinate to the vertices of $\mathcal{G}^*$, defines a λ-drawing of $\mathcal{G}$.*

While both Lemma 69 and Lemma 70 can be proven directly, we defer their proofs to Section 10.4 where we introduce the distance model and prove Lemma 71 and Lemma 72, the stronger versions of Lemma 69 and Lemma 70, respectively. Combining Lemma 69 and Lemma 70 we obtain the following.

Theorem 29. *Let $\mathcal{G}$ be an embedded proper level-planar graph and let $\lambda \in \mathbb{N}$. The circulations in $F^\lambda_\mathcal{G}$ correspond bijectively to the λ-drawings of $\mathcal{G}$.*

Theorem 29 implies that a λ-drawing can be found by applying existing flow algorithms to $F^\lambda_\mathcal{G}$. For that, first transform our flow network with arc demands to the standard single-source single-sink maximum flow setting without arc demands using the construction due to Kleinberg and Tardos [KT06, Chapter 7.7]. This construction adds one new "super-source" s^* and one new "super-sink" t^* to $F^\lambda_\mathcal{G}$, and connects them with arcs to the other nodes in $F^\lambda_\mathcal{G}$. In particular, the size of the resulting flow network is linear in the size of $F^\lambda_\mathcal{G}$. Subdivide the arcs incident to s^*, t^* and then remove s^*, t^* from the flow network, obtaining an instance of the multiple-source multiple-sink maximum flow problem. Note that this network is planar. Use the $O(n \log^3 n)$-time multiple-source multiple-sink maximum flow algorithm due to Borradaile et al. [Bor+17] to find a circulation in $F^\lambda_\mathcal{G}$, or to determine that no circulation exists.

Corollary 8. *Let $\mathcal{G}$ be an embedded proper level-planar graph and let $\lambda \in \mathbb{N}$. It can be tested in $O(n \log^3 n)$ time whether a λ-drawing of $\mathcal{G}$ exists, and if so, such a drawing can be found within the same running time.*

10.3.1 Connected Partial Drawings

Recall that a partial λ-drawing is a tuple $(\mathcal{G}, \mathcal{H}, \Pi)$, where $\mathcal{G}$ is an embedded level-planar graph, $\mathcal{H}$ is an embedded subgraph of $\mathcal{G}$ and Π is a λ-drawing of $\mathcal{H}$. We say that $(\mathcal{G}, \mathcal{H}, \Pi)$ is *λ-extendable* if $\mathcal{G}$ admits a λ-drawing Γ whose restriction to $\mathcal{H}$ is Π. Here Γ is referred to as a *λ-extension* of $(\mathcal{G}, \mathcal{H}, \Pi)$.

In this section we show that in case $\mathcal{H}$ is connected, we can use the flow model to decide whether $(\mathcal{G}, \mathcal{H}, \Pi)$ is λ-extendable. Observe that when $\mathcal{H}$ is connected Π is completely defined by the slopes of the edges in $\mathcal{H}$ up to horizontal translation. Let $F_{\mathcal{G}}^{\lambda}$ be the flow network corresponding to $\mathcal{G}$. In order to fix the slopes of an edge e of $\mathcal{H}$ to a value ℓ, we fix the flow across the dual slope arc $e^{\star}$ in $\mathcal{H}$ to ℓ. Checking whether a circulation in the resulting flow network exists can be reduced to a multiple-source multiple-sink maximum flow problem, which once again can be solved by the algorithm due to Borradaile et al. [Bor+17].

Corollary 9. *Let $(\mathcal{G}, \mathcal{H}, \Pi)$ be a partial λ-drawing where $\mathcal{H}$ is connected. It can be tested in $O(n \log^3 n)$ time whether $(\mathcal{G}, \mathcal{H}, \Pi)$ is λ-extendable, and if so, a corresponding λ-extension can be constructed within the same running time.*

10.4 Dual Distance Model

A minimum cut (and, equivalently, the value of the maximum flow) of an st-planar graph G can be determined by computing a shortest $(s^{\star}, t^{\star})$-path in a dual of G [Hu69, IS79]. Hassin showed that to construct a flow, it is sufficient to compute the distances from $s^{\star}$ to all other vertices in the dual graph [Has81]. To the best of our knowledge, this duality has been exploited only for flow networks with arc capacities, but not with arc demands. In this section, we extend this duality to arcs with demands.

The resulting dual distance model improves the running time for the λ-Drawability, lets us test the existence of λ-extensions of partial λ-drawings for non-connected subgraphs, and allows us to develop an efficient algorithm for testing the existence of simultaneous λ-drawings.

We define $D_{\mathcal{G}}^{\lambda}$ to be the weighted directed dual of $F_{\mathcal{G}}^{\lambda}$ as follows. Let a be an arc of $F_{\mathcal{G}}^{\lambda}$ with demand $d(a)$ and capacity $c(a)$. Further, let f_{left} and f_{right} denote the left and the right faces of a in $F_{\mathcal{G}}^{\lambda}$, respectively. The dual $D_{\mathcal{G}}^{\lambda}$ contains f_{left} and f_{right} as vertices connected by the directed edge $(f_{\text{left}}, f_{\text{right}})$ with weight $c(a)$ and the directed edge $(f_{\text{right}}, f_{\text{left}})$ with weight $-d(a)$; see Fig. 10.3. Equivalently, $D_{\mathcal{G}}^{\lambda}$ is obtained directly from $\mathcal{G}$ as follows. Recall that if a is a slope arc, it is $d(a) = 0, c(a) = \lambda - 1$ and $f_{\text{left}}, f_{\text{right}}$ correspond to vertices u, w connected by the edge (u, w) in G. So, create for each directed edge (u, w) of $\mathcal{G}$ the weighted directed edges (u, w) with weight $\lambda - 1$ and (w, u) with weight 0 in $D_{\mathcal{G}}^{\lambda}$; see Fig. 10.3 (a). If a is not a slope arc then a is a space arc and it is $d(a) = 1, c(a) = (\lambda - 1)(n - 1)$ and $f_{\text{left}}, f_{\text{right}}$ correspond to consecutive vertices u, v in $\mathcal{G}$, where u appears to the left of v. So, create for each pair of consecutive vertices u, v in $\mathcal{G}$ where u appears to the left of v the weighted directed edges (u, v) with weight $(\lambda - 1)(n - 1)$ and (v, u) with weight -1 in $D_{\mathcal{G}}^{\lambda}$; see Fig. 10.3 (b). Observe that $D_{\mathcal{G}}^{\lambda}$ has the vertex set V of $\mathcal{G}$ and a superset of its

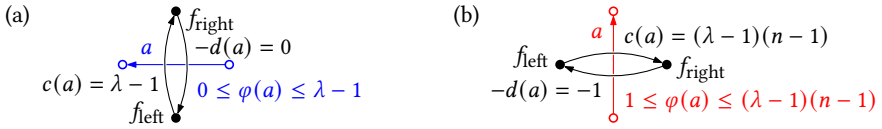


Figure 10.3: Definition of the dual edges for a flow network arc $a = (u, v)$ with demand $d(a)$ and capacity $c(a)$. Let f_{left} and f_{right} denote the vertices corresponding to the faces to the left and right of a in $F^{\lambda}_{\mathcal{G}}$. Then add the edge $(f_{\text{left}}, f_{\text{right}})$ with weight $c(a)$ and the reverse edge $(f_{\text{right}}, f_{\text{left}})$ with weight $-d(a)$. The edge weights depend on whether a is a slope arc (a) or a space arc (b).

edges. Fig. 10.4 shows the distance network obtained from the flow network shown in Fig. 10.2 (c).

A *distance labeling* is a function $x : V \to \mathbb{Z}$ that for every edge (u, v) of $D^{\lambda}_{\mathcal{G}}$ with weight l satisfies $x(v) \le x(u) + l$. We also say that (u, v) *imposes the distance constraint* $x(v) \le x(u) + l$. A distance labeling for $D^{\lambda}_{\mathcal{G}}$ is the x-coordinate assignment for a λ-drawing: For an edge (u, v) of $D^{\lambda}_{\mathcal{G}}$ where u, v are consecutive vertices in $\mathcal{G}$, the distance labeling guarantees $x(v) \le x(u) - 1$, i.e., the consecutive vertices are in the correct order and do not overlap. If an edge (u, v) between layers has weight $\lambda - 1$, then the distance labeling ensures $x(v) \le x(u) + \lambda - 1$, i.e., (u, v) has a slope in $\{0, \ldots, \lambda - 1\}$. Computing the shortest distances from v_{right} in $D^{\lambda}_{\mathcal{G}}$ to every vertex (if they are well-defined) gives a distance labeling that we refer to as the *shortest distance labeling*. A distance labeling of $D^{\lambda}_{\mathcal{G}}$ does not necessarily exist. This is the case when $D^{\lambda}_{\mathcal{G}}$ contains a negative cycle, e.g., when the in- or out-degree of a vertex in $\mathcal{G}$ is strictly larger than λ. For a distance labeling x of $D^{\lambda}_{\mathcal{G}}$ we define a dual circulation $x^{\star}$ as follows. Recall that each arc a of $F^{\lambda}_{\mathcal{G}} - (t, s)$ is a slope arc or a space arc. If a is a slope arc it is dual to an edge (u, w) of $\mathcal{G}$. Recall that u, w are vertices of $D^{\lambda}_{\mathcal{G}}$ and define $x^{\star}(a) := x(w) - x(u)$. If a is a space arc it is dual to consecutive vertices u, v of $\mathcal{G}$. Again, u, v are vertices of $D^{\lambda}_{\mathcal{G}}$, define $x^{\star}(a) := x(v) - x(u)$.

Lemma 71. *Let $\mathcal{G}$ be an embedded level-planar graph and Γ be a λ-drawing of $\mathcal{G}$. The function x that assigns to each vertex of $\mathcal{G}$ its x-coordinate in Γ is a distance labeling of $D^{\lambda}_{\mathcal{G}}$ and its dual $x^{\star}$ is a circulation in $F^{\lambda}_{\mathcal{G}}$.*

Proof. Since Γ preserves the embedding of $\mathcal{G}$, for each consecutive vertices v, u, with v preceding u in $\mathcal{G}$ it holds that $\Gamma(v) < \Gamma(u)$. Because Γ is a grid drawing $\Gamma(v) \le \Gamma(u) - 1$, which implies $x(v) \le x(u) + l$, where $l = -1$ is the weight of (u, v). Since Γ is a λ-drawing, i.e., every edge (u, v) between the two levels has a slope in $\{0, \ldots \lambda - 1\}$, it holds that $\Gamma(u) \le \Gamma(v) \le \Gamma(u) + \lambda - 1$, which implies $x(u) \le x(v) + 0$, for the

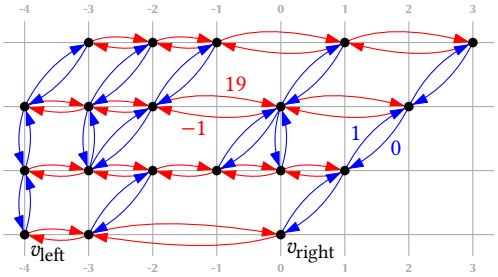


Figure 10.4: The distance network $D^2_{\mathcal{G}}$ obtained from the flow network $F^2_{\mathcal{G}}$ shown in Fig. 10.2 (c). The x-coordinate of every vertex is its distance from v_{right} in $D^2_{\mathcal{G}}$. Red arcs pointing right have weight $(\lambda-1)(n-1) = 19$, red arcs pointing left have weight -1. Blue arcs pointing up have weight $\lambda - 1 = 1$ and blue arcs pointing down have weight 0.

edge (v,u) of $D^\lambda_{\mathcal{G}}$ with weight zero and $x(v) \leq x(u) + \lambda - 1$ for the edge (u,v) of $D^\lambda_{\mathcal{G}}$ with weight $\lambda - 1$. Hence, x is a distance labeling of $D^\lambda_{\mathcal{G}}$.

We now show that $x^\star$ is a circulation in $F^\lambda_{\mathcal{G}}$. Let $f_1, f_2, \ldots, f_t, f_{t+1} = f_1$ be the faces incident to some node v of $F^\lambda_{\mathcal{G}}$ in counter-clockwise order. Let a be the arc incident to v and dual to the edge between f_i and f_{i+1} with $1 \leq i \leq t$. If a is an incoming arc, it adds a flow of $x(f_{i+1}) - x(f_i)$ to v. If a is an outgoing arc, it removes a flow of $x(f_i) - x(f_{i+1})$ from v, or, equivalently, it adds a flow of $x(f_{i+1}) - x(f_i)$ to v. Therefore, the flow through v is $\sum_i (x(f_{i+1}) - x(f_i))$. This sum cancels to zero, i.e., the flow is preserved at v. Now consider an arc a of $F^\lambda_{\mathcal{G}} - (t,s)$. If a is a slope arc it is dual to an edge (u,w) of $\mathcal{G}$. Then $D^\lambda_{\mathcal{G}}$ contains the edge (u,w) with weight $c(a)$, which ensures $x(w) \leq x(u) + c(a)$, so $x^\star(a) \leq c(a)$. It also contains the edge (w,u) with weight $-d(a)$, which ensures $x(u) \leq x(w) - d(a)$, so $x^\star(a) \geq d(a)$. If a is a space arc it is dual to consecutive vertices u, v of $\mathcal{G}$. Then $D^\lambda_{\mathcal{G}}$ contains the edge (u,v) with weight $c(a)$, which ensures $x(v) \leq x(u) + c(a)$, so $x^\star(a) \leq c(a)$. It also contains the edge (v,u) with weight $-d(a)$, which ensures $x(u) \leq x(v) - d(a)$, so $x^\star(a) \geq d(a)$. Finally, the arc (t,s) has demand 0 and capacity ∞, so its demand is satisfied and its capacity is not exceeded. Hence, $x^\star$ is indeed a circulation in $F^\lambda_{\mathcal{G}}$. □ □

Recall from Section 10.3 that for a circulation φ in $F^\lambda_{\mathcal{G}}$ we define a dual drawing $\varphi^\star$ by setting the x-coordinates of the vertices of $\mathcal{G}$ as follows. For the lowest vertex of the right boundary set $\varphi^\star(v_{\text{right}}) = 0$. Process the remaining vertices of the right boundary in ascending order with respect to their levels. Let (u,v) be an edge of the right boundary so that u has already been processed and v has not been processed

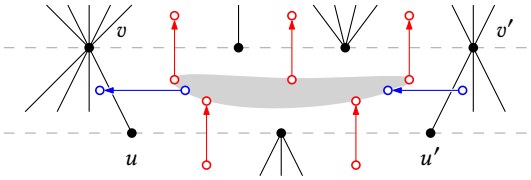


Figure 10.5: Proof of Lemma 72. Sets A and B contain the incoming and outgoing red flow network arcs incident to the gray oval, respectively.

yet. Then set $\varphi^\star(v) = \varphi^\star(u) + \varphi((u,v)^\star)$, where $(u,v)^\star$ is the slope arc dual to (u,v). Let w, x be a pair of consecutive vertices so that x has already been processed and w has not yet been processed yet. Then set $\varphi^\star(w) = \varphi^\star(x) - \varphi([w,x]^\star)$, where $[w,x]^\star$ is a space arc. It turns out that $\varphi^\star$ is a distance labeling of $D^\lambda_\mathcal{G}$ and a λ-drawing of $\mathcal{G}$.

Lemma 72. *Let $\mathcal{G}$ be an embedded level-planar graph, let $\lambda \in \mathbb{N}$, and let φ be a circulation in $F^\lambda_\mathcal{G}$. The dual $\varphi^\star$ is a distance labeling of $D^\lambda_\mathcal{G}$ and, when interpreted as assigning an x-coordinate to the vertices of $\mathcal{G}$, defines a λ-drawing of $\mathcal{G}$.*

Proof. We show that $\varphi^\star$ is a distance labeling in $D^\lambda_\mathcal{G}$. The algorithm described above assigns a value to every vertex of $D^\lambda_\mathcal{G}$. We now show that $\varphi^\star$ is indeed a distance labeling by showing that every edge satisfies a distance constraint.

Observe that the distance constraints imposed by edges dual to the space arcs are satisfied by construction. To show that the distance constraints imposed by edges dual to the slope arcs are also satisfied, we prove that for every edge (u,v), it holds that $\varphi^\star(v) = \varphi^\star(u) + \varphi((u,v)^\star)$. We refer to this as *condition $\mathcal{C}$* for short. Since $\varphi((u,v)^\star) \leq \lambda - 1$ and the weight l of (u,v) is $l = \lambda - 1$ we obtain that it is $\varphi^\star(v) = \varphi^\star(u) + \ell$, which implies that $\varphi^\star$ is a distance labeling of $D^\lambda_\mathcal{G}$.

The proof is by induction based on the bottom to top and right to left order among the edges of $D^\lambda_\mathcal{G}$. We say that (a,b) *precedes* (c,d) if either $\ell(a) < \ell(c)$, or $\ell(a) = \ell(c)$ and a is to the right of c, or $\ell(a) = \ell(c)$ and b is to the right of d (in case $a = c$). For the base case observe that the edges with both end-vertices on the first level and the edges of p_{right} satisfy condition $\mathcal{C}$ by the definition of $\varphi^\star$. Now let (u,v) be an edge not addressed in the base case and assume that for every edge (u',v') preceding edge (u,v) condition $\mathcal{C}$ holds. For the inductive step we show that condition $\mathcal{C}$ also holds for (u,v). Let (u',v') denote the edge to the right of (u,v) so that (u,v) and (u',v') are consecutive; see Fig. 10.5. Because v is not the rightmost vertex on its level this edge exists. Let A denote the set of space arcs $[u_1,u_2]^\star$ in $F^\lambda_\mathcal{G}$ with $u_1, u_2 \in [u',u]$. Analogously, let B denote the set of space arcs $[v_1,v_2]^\star$ in $F^\lambda_\mathcal{G}$ with $v_1, v_2 \in [v',v]$. It is $\varphi^\star(v) = \varphi^\star(v') - \sum_{b\in B}\varphi(b)$ by definition of $\varphi^\star$. Further, by induction hypothesis

and since (u', v') precedes (u, v) it is $\varphi^\star(v') = \varphi^\star(u') + \varphi((u', v')^\star)$. Inserting the latter into the former equation, we obtain

$$\varphi^\star(v) = \varphi^\star(u') + \varphi((u', v')^\star) - \sum_{b \in B} \varphi(b). \tag{10.1}$$

Again, by definition of $\varphi^\star$, it is $\varphi^\star(u) = \varphi^\star(u') - \sum_{a \in A} \varphi(a)$. Solving for $\varphi^\star(u')$ and inserting into (10.1) we obtain

$$\varphi^\star(v) = \varphi^\star(u) + \sum_{a \in A} \varphi(a) + \varphi((u', v')^\star) - \sum_{b \in B} \varphi(b). \tag{10.2}$$

Flow conservation on the vertex of $F^\lambda_{\mathcal{G}}$ to which edges of A and B are incident gives

$$\varphi((u', v')^\star) + \sum_{a \in A} \varphi(a) = \varphi((u, v)^\star) + \sum_{b \in B} \varphi(b). \tag{10.3}$$

Solving equation (10.3) for $\varphi((u', v')^\star)$ and then inserting it into equation (10.2) yields $\varphi^\star(v) = \varphi^\star(u) + \varphi((u, v)^\star)$, i.e., condition $\mathcal{C}$ holds for (u, v). Therefore $\varphi^\star$ is a distance labeling, which we have shown to define a λ-drawing of $\mathcal{G}$. □ □

Because $D^\lambda_{\mathcal{G}}$ is planar we can use the $O(n \log^2 n / \log \log n)$-time shortest path algorithm due to Mozes and Wulff-Nilsen [MW10] to compute the shortest distance labeling. This improves our $O(n \log^3 n)$-time algorithm from Section 10.3.

Theorem 30. *Let $\mathcal{G}$ be an embedded proper level-planar graph. The distance labelings of $D^k_{\mathcal{G}}$ correspond bijectively to the λ-drawings of $\mathcal{G}$. If such a drawing exists, it can be found in $O(n \log^2 n / \log \log n)$ time.*

10.5 Partial and Simultaneous Drawings

In this section we use the distance model from Section 10.4 to construct partial and simultaneous λ-drawings. We start with introducing a useful kind of drawing. Let Γ be a λ-drawing of $\mathcal{G}$. We call Γ a *λ-rightmost* drawing when there exists no λ-drawing Γ' with $\Gamma(v) < \Gamma'(v)$ for some $v \in V$. In this definition, we assume $x(\Gamma(v_{\text{right}})) = x(\Gamma'(v_{\text{right}})) = 0$ to exclude trivial horizontal translations. Hence, a drawing is rightmost when every vertex is at its rightmost position across all level-planar λ-slope grid drawings of $\mathcal{G}$. It is not trivial that a λ-rightmost drawing exists, but it follows directly from the definition that if such a drawing exists, it is unique. The following lemma establishes the relationship between λ-rightmost drawings and shortest distance labelings of $D^\lambda_{\mathcal{G}}$.

Lemma 73. *Let $\mathcal{G}$ be an embedded proper level-planar graph. If $D^\lambda_\mathcal{G}$ has a shortest distance labeling it describes the λ-rightmost drawing of $\mathcal{G}$.*

Proof. The shortest distance labeling of $D^\lambda_\mathcal{G}$ is maximal in the sense that for any vertex v there exists a vertex u and an edge (u, v) with weight l so that it is $x(v) = x(u)+l$. Recall that the definition of distance labelings only requires $x(v) \leq x(u) + l$. The claim then follows by induction over V in ascending order with respect to the shortest distance labeling. □ □

10.5.1 Partial Drawings

Let $(\mathcal{G}, \mathcal{H}, \Pi)$ be a partial λ-drawing. In Section 10.3.1 we have shown that the flow model can be adapted to check whether $(\mathcal{G}, \mathcal{H}, \Pi)$ has a λ-extension, in case $\mathcal{H}$ is connected. In this section, we show how to adapt the distance model to extend partial λ-drawings, including the case $\mathcal{H}$ is disconnected. Recall that the distance label of a vertex v is its x-coordinate. A partial λ-drawing fixes the x-coordinates of the vertices of $\mathcal{H}$. The idea is to express this with additional constraints in $D^\lambda_\mathcal{G}$. Let v_{ref} be a vertex of $\mathcal{H}$. In a λ-extension of $(\mathcal{G}, \mathcal{H}, \Pi)$, the relative distance along the x-axis between a vertex v of $\mathcal{H}$ and vertex v_{ref} should be $d_v = \Pi(v_{\text{ref}}) - \Pi(v)$. This can be achieved by adding an edge (v, v_{ref}) with weight d_v and an edge (v_{ref}, v) with weight $-d_v$. The first edge ensures that it is $x(v_{\text{ref}}) \leq x(v)+d_v$, i.e., $x(v) \geq x(v_{\text{ref}})-d_v$ and the second edge ensures $x(v) \leq x(v_{\text{ref}})-d$. Together, this gives $x(v) = x(v_{\text{ref}})-d_v$. Let $D^\lambda_{\mathcal{G},\Pi}$ be $D^\lambda_\mathcal{G}$ augmented by the edges $\{(v, v_{\text{ref}}), (v_{\text{ref},v}) : \forall v \in \mathcal{H}\}$ with weights as described above.

To decide existence of λ-extension and in affirmative construct the corresponding drawing we compute the shortest distance labeling in $D^\lambda_{\mathcal{G},\Pi}$. Observe that this network can contain negative cycles and therefore no shortest distance labeling. Unfortunately, $D^\lambda_{\mathcal{G},\Pi}$ is not planar, and thus we cannot use the embedding-based algorithm of Mozes and Wulff-Nilsen. However, since all newly introduced edges have v_{ref} as one endpoint, v_{ref} is an *apex* of $D^\lambda_\mathcal{G}$, i.e., removing v_{ref} from $D^\lambda_{\mathcal{G},\Pi}$ makes it planar. Therefore $D^\lambda_{\mathcal{G},\Pi}$ can be recursively separated by separators of size $O(\sqrt{n})$. The $O(n^{4/3} \log n)$-time shortest-path algorithm by Henzinger et al. [HKRS97] relies not on planarity but only on $O(\sqrt{n})$-sized separators [FR06, page 869]. So, run this algorithm to compute the shortest distance labeling of $D^\lambda_{\mathcal{G},\Pi}$.

Theorem 31. *Let $(\mathcal{G}, \mathcal{H}, \Pi)$ be a partial λ-drawing. In $O(n^{4/3} \log n)$ time it can be determined whether $(\mathcal{G}, \mathcal{H}, \Pi)$ has a λ-extension and in the affirmative the corresponding drawing can be computed within the same running time.*

10.5.2 Simultaneous Drawings

In the simultaneous λ-drawing problem, we are given a tuple $(\mathcal{G}_1, \mathcal{G}_2)$ of two embedded level-planar graphs that share a common subgraph $\mathcal{G}_{1\cap 2} = \mathcal{G}_1 \cap \mathcal{G}_2$. We assume w.l.o.g. that G_1 and G_2 share the same right boundary and that the embeddings of $\mathcal{G}_1$ and $\mathcal{G}_2$ coincide on $\mathcal{G}_{1\cap 2}$. The task is to determine whether there exist λ-drawings Γ_1, Γ_2 of $\mathcal{G}_1, \mathcal{G}_2$, respectively, so that Γ_1 and Γ_2 coincide on the shared graph $\mathcal{G}_{1\cap 2}$. The approach is the following. Start by computing the λ-rightmost drawings of $\mathcal{G}_1$ and $\mathcal{G}_2$. Then, as long as these drawings do not coincide on $\mathcal{G}_{1\cap 2}$ add necessary constraints to $D^\lambda_{\mathcal{G}_1}$ and $D^\lambda_{\mathcal{G}_2}$. This process terminates after a polynomial number of iterations, either by finding a simultaneous λ-drawing, or by determining that no such drawing exist.

Finding the necessary constraints works as follows. Suppose that Γ_1, Γ_2 are the λ-rightmost drawings of $\mathcal{G}_1, \mathcal{G}_2$, respectively. Because both $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same right boundary they both contain vertex v_{right}. We define the coordinates in the distance labelings of $D^\lambda_{\mathcal{G}_1}$ and $D^\lambda_{\mathcal{G}_2}$ in terms of this reference vertex.

Now suppose that for some vertex v of $\mathcal{G}_{1\cap 2}$ the x-coordinates in Γ_1 and Γ_2 differ, i.e., it is $\Gamma_1(v) \neq \Gamma_2(v)$. Assume $\Gamma_1(v) < \Gamma_2(v)$ without loss of generality. Because Γ_1 is a rightmost drawing, there exists no drawing of $\mathcal{G}_1$ where v has an x-coordinate greater than $\Gamma_1(v)$. In particular, there exist no simultaneous drawings where v has an x-coordinate greater than $\Gamma_1(v)$. Therefore, we must search for a simultaneous drawing where $\Gamma_2(v) \leq \Gamma_1(v)$. We can enforce this constraint by adding an edge (v_{right}, v) with weight $\Gamma_1(v)$ into $D^\lambda_{\mathcal{G}_2}$. We then attempt to compute the drawing Γ_2 of $\mathcal{G}_2$ defined by the shortest distance labeling in $D^\lambda_{\mathcal{G}_2}$. This attempt produces one of two possible outcomes. The first possibility is that there now exists a negative cycle in $D^\lambda_{\mathcal{G}_2}$. This means that there exists no drawing Γ_2 of G_2 with $\Gamma_2(v) \leq \Gamma(v)$. Because Γ_1 is a rightmost drawing, this means that no simultaneous drawings of $\mathcal{G}_1$ and $\mathcal{G}_2$ exist. The algorithm then terminates and rejects this instance. The second possibility is that we obtain a new drawing Γ_2. This drawing is rightmost among all drawings that satisfy the added constraint $\Gamma_2(v) \leq \Gamma_1(v)$. In this case there are again two possibilities. Either we have $\Gamma_1(v) = \Gamma_2(v)$ for each vertex v in $\mathcal{G}_{1\cap 2}$. In this case Γ_1 and Γ_2 are simultaneous drawings and the algorithm terminates. Otherwise there exists at least one vertex w in $\mathcal{G}_{1\cap 2}$ with $\Gamma_1(w) \neq \Gamma_2(w)$. We then repeat the procedure just described for adding a new constraint.

We repeat this procedure of adding other constraints. To bound the number of iterations, recall that we only consider compact drawings, i.e., drawings whose width is at most $(\lambda - 1)(n - 1)$. In each iteration the x-coordinate of at least one vertex is decreased by at least one. Therefore, each vertex is responsible for at most $(\lambda - 1)(n - 1)$ iterations. The total number of iterations is therefore bounded

by $n(\lambda-1)(n-1) \in O(\lambda n^2)$.

Note that due to the added constraints $D^\lambda_{\mathcal{G}_1}$ and $D^\lambda_{\mathcal{G}_2}$ are generally not planar. However, all newly inserted arcs are incident to v_{right}, so v_{right} is an apex of $D^\lambda_{\mathcal{G}_i}$ for $i = 1, 2$. As in the previous section, we apply the $O(n^{4/3} \log n)$-time shortest-path algorithm by Henzinger et al. to compute the shortest distance labelings. This gives the following.

Theorem 32. *Let $\mathcal{G}_1, \mathcal{G}_2$ be embedded level-planar graphs that share a common subgraph $\mathcal{G}_{1\cap 2}$. In $O(\lambda n^{10/3} \log n)$ time it can be determined whether $\mathcal{G}_1, \mathcal{G}_2$ admit simultaneous λ-drawings and if so, such drawings can be computed within the same running time.*

10.6 Complexity of the General Case

So far, we have considered λ-Drawability problem for proper level graphs, i.e., level graphs where all edges have length one. In this section, we consider the general case, where edges may have arbitrary lengths. We say that an edge with length two or more is *long*. One approach would be to try to adapt the flow model from Section 10.3 to this more general case. By subdividing long edges, any level graph G can be transformed into a proper level graph G'. Observe that two edges in G' created by subdividing the same long edge must have the same slope in order to yield a fixed-slope drawing of G. In the context of our flow model, this means that the amount of flow across the corresponding slope arcs must be the same. Our problem then becomes an instance of the *integer equal flow problem*. In this problem, we are given a flow network along with disjoint sets $R_1, R_2, \ldots, R_t$ of arcs. The task is to find the maximum flow from s to t such that the amount of flow across arcs in the same set R_i is the same. This problem was introduced and shown to be NP-hard by Sahni [Sah74]. The problem remains NP-hard in special cases [EIS76, Sri+02] and the integrality gap of the fractional LP can be arbitrarily large [MS09].

In this section we show that λ-Drawability is NP-complete even for $\lambda = 2$, biconnected graphs where all edges have length one or two. To this end, we present a reduction from *rectilinear planar monotone 3-Sat* [dK12]. An instance of this problem consists of a set of variables X and a set of clauses C. A clause is *positive* (*negative*) when it consists of only positive (negative) literals. We say that the instance is *monotone* when each clause is either positive or negative. The corresponding *variable-clause graph* consists of the vertices $X \cup C$ and each undirected edge $\{x, c\}$ where $x \in X$ is a variable that appears in clause $c \in C$. The variable-clause graph admits a planar drawing such that (i) the variables are aligned along a virtual horizontal line ℓ_X, (ii) positive clauses are drawn as vertices above ℓ_X, and symmetrically (iii)negative

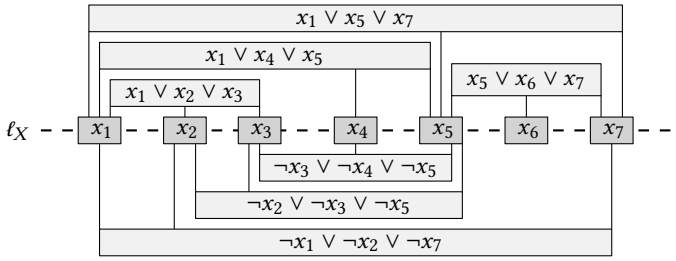


Figure 10.6: An instance of planar monotone 3-SAT.

clauses are drawn as vertices below ℓ_X. See Fig. 10.6.

Our reduction works by first replacing every vertex that corresponds to a variable by a variable gadget and every vertex that corresponds to a positive (negative) clause by a positive (negative) clause gadget. All three gadgets consist of *fixed* and *movable* parts. The fixed parts only admit one level-planar two-slope drawing, whereas the movable parts admit two or more drawings depending on the choice of slope for some edges. Second, the gadgets are connected by a common fixed frame. All fixed parts of the gadgets are connected to the common frame in order to provide a common point of reference. The movable parts of the gadgets then interact in such a way that any level-planar two-slope drawing induces a solution to the underlying planar monotone 3-SAT instance.

The variable gadget consists of a number of *connectors* arranged around a fixed horizontal line that connects all variable gadgets along the virtual line of variables ℓ_X. See Fig. 10.7, where the fixed structure is shaded in gray. Vertices drawn as squares are fixed, i.e., they cannot change their position relative to other vertices drawn as squares. Vertices drawn as circles are movable, i.e., they can change their position relative to vertices drawn as squares. The line of variables ℓ_X extends from the the square vertices on the left and right boundaries of the drawing. Every connector consists of two *pins*: the movable *assignment pin* and the fixed *reference pin*. The variable gadget in Fig. 10.7 features four connectors: two above the horizontal line and two below the horizontal line. Assignment pins are shaded in yellow and reference pins are shaded in gray. The relative position of the assignment pin and the reference pin of one connector encodes the truth assignment of the underlying variable. Moreover, the reference pin allows the fixed parts of the clause gadgets to be connected to the variable gadgets and thereby to the common frame. Comparing Fig. 10.7 (a) and (b), observe how the relative position of the two pins of each connector changes depending on the truth assignment of the underlying variable. The key structure of the variable gadget is that the position of the assignment pins of one variable

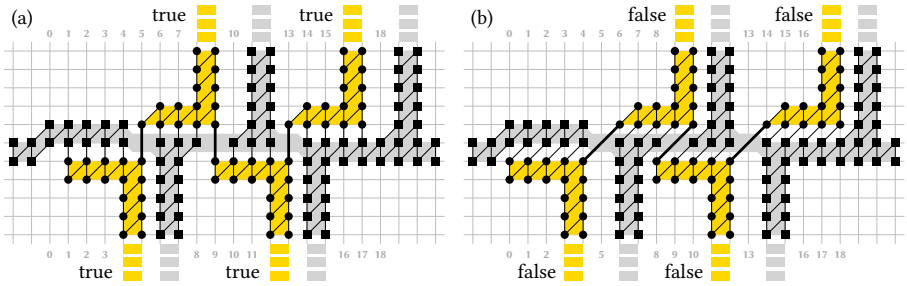


Figure 10.7: The variable gadget drawn in the "true" configuration (a) and the "false" configuration (b).

gadget are coordinated by long edges. In Fig. 10.7, long edges are drawn as thick lines. Changing the slope of these long edges moves all assignment pins above the horizontal line in one direction and all assignment pins below the horizontal line in the reverse direction. In this way, all connectors encode the same truth assignment of the underlying variable. Note that we can introduce as many connectors as needed for any one variable.

The positive clause gadget consists of a fixed boundary, a movable *wiggle* and three *assignment pin endings*. See Fig. 10.8, where the fixed boundary is shaded in gray, the wiggle is highlighted in blue and the assignment pin endings are highlighted in yellow. The fixed boundary is connected to the reference pin of the variable gadget that is rightmost among the connected variable gadgets. Because the variable gadgets are fixed to the common frame, the boundary of the positive clause gadget is also connected to the common frame. The assignment pin endings are connected to assignment pins of connectors of the corresponding variable gadgets. The idea of the positive clause gadget is that the wiggle has to wiggle through the space bounded by the assignment pin endings on the left and the fixed boundary on the right. Recall that the assignment pins change their horizontal position depending on the truth assignment of their underlying variables. The positive clause gadget is designed so that the wiggle can always be drawn, except for the case when all variables are assigned to false. See Fig. 10.8 (a)–(c), which shows the three possible situations when exactly one variable is assigned to true. In any case where at least one variable is assigned to true the wiggle can be drawn in one of the three ways shown. However, as shown in Fig. 10.8 (d), the wiggle cannot be drawn in the case where all variables are assigned to false. The reason for this is that the the assignment pin endings get so close to the fixed boundary that they leave too little space for the wiggle to be drawn. This means that some vertices must intersect, for example those shown in

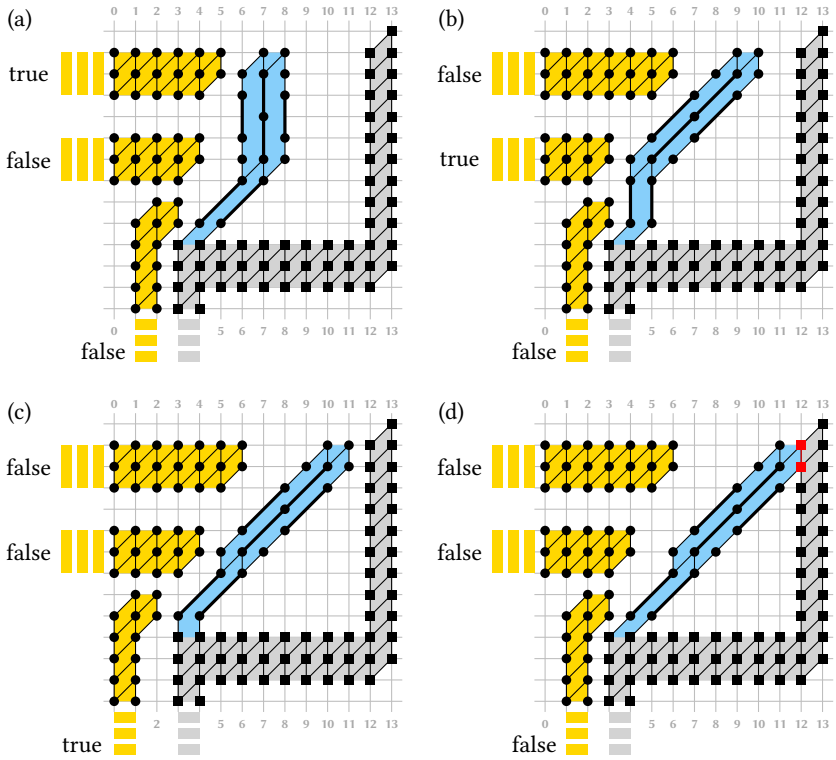


Figure 10.8: The positive clause gadget when the drawing when at least one variable is assigned to true (a–c). No planar drawing exists when all variables are assigned to false (d), because this leads to intersections, e.g., at the vertices marked in red.

red in Fig. 10.8 (d).

The negative clause gadgets works very similarly. It is drawn below the horizontal line of variables and it forces at least one of the incident variable gadgets to be configured as false. See Fig. 10.9 which shows admissible drawings (a–c) and that the case when all incident variables are configured as true cannot occur (d). Note that this uses the design of the variable gadget that the assignment and reference pins below the horizontal line are closer when the variable is assigned to true (i.e., the inverse situation compared to the situation above the horizontal line).

It is evident that any level-planar two-slope drawing of the resulting graph induces a solution of the underlying planar monotone 3-SAT problem and vice versa. Note

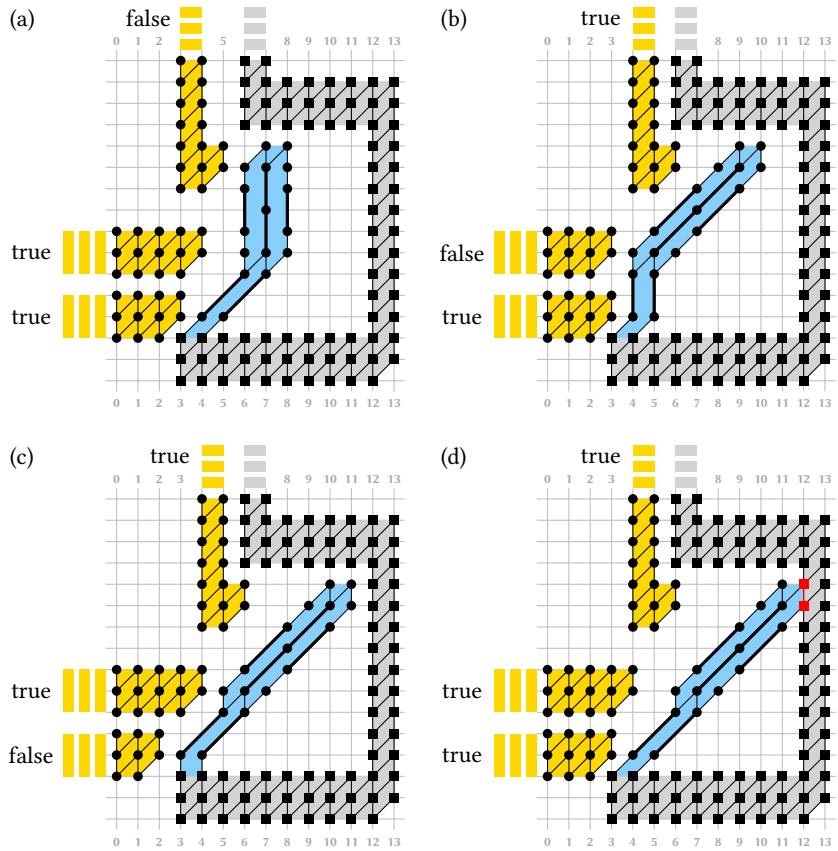


Figure 10.9: The negative clause gadget when at least one variable is assigned to false (a–c). No drawing exists when all variables are assigned to true (d), because this leads to intersections, e.g., at the vertices marked in red. Note the symmetry to the positive clause gadget in Fig. 10.8.

that the variable gadgets become biconnected when embedded into the common frame and that the clause gadgets are biconnected by design. Furthermore, all long edges have length two. We therefore conclude the following.

Theorem 33. *λ-Drawability is NP-complete even for $\lambda = 2$ and biconnected graphs where all edges have length one or two.*

10.7 Conclusion

In this chapter we studied λ-drawings, i.e., level-planar drawings with λ slopes. We model λ-drawings of proper level-planar graphs as integer flow networks. This lets us compute λ-drawings and extend connected partial λ-drawings in $O(n \log^3 n)$ time. We extend the duality between integer flows in a primal graph and shortest distances in its dual to obtain a more powerful distance model. This distance model lets us find λ-drawings in $O(n \log^2 n / \log \log n)$ time, extend not-necessarily-connected partial λ-drawings in $O(n^{4/3} \log n)$ time and find simultaneous λ-drawings in $O(\lambda n^{10/3} \log n)$ time.

In the non proper case, testing the existence of a 2-drawing becomes NP-hard, even for biconnected graphs with maximum edge length two. This leaves little room to extend our polynomial-time algorithms for more general classes of level-planar graphs.

11 Drawing Two Posets

We investigate the problem of drawing two posets of the same ground set so that one is drawn from left to right and the other one is drawn from the bottom up. The input to this problem is a directed graph $G = (V, E)$ and two sets X, Y with $X \cup Y = E$, each of which can be interpreted as a partial order of V. The task is to find a planar drawing of G such that each directed edge in X is drawn as an x-monotone edge, and each directed edge in Y is drawn as a y-monotone edge. Such a drawing is called an *xy-planar* drawing.

Testing whether a graph admits an xy-planar drawing is NP-complete in general. We consider the case that the planar embedding of G is fixed and the subgraph of G induced by the edges in Y is a connected spanning subgraph of G whose upward embedding is fixed. For this case we present a linear-time algorithm that determines whether G admits an xy-planar drawing and, if so, produces an xy-planar polyline drawing with at most three bends per edge.

This chapter extends work initiated as part of Vera Chekan's bachelor's thesis [Che19] and is based on joint work with Vera Chekan [BC20].

11.1 Introduction

A partial order $<$ over a set V can be interpreted as a directed graph G by interpreting the elements of V as the vertices of G, and interpreting the fact $u < v$ for $u, v \in V$ as a directed edge from u to v in G. An upward drawing of G is a bottom-up visualization

of the partial order $<$. If the drawing is also planar, then it is especially easy to understand for humans [Pur97]. Testing graphs for upward planarity is NP-hard in general [GT01], but feasible in linear time for graphs with a single source [BDMT98] and for graphs with a fixed underlying planar embedding [BDLM94].

More recent research has sought to extend this concept to two directions. The input to Bi-Monotonicity is an undirected graph whose vertices have fixed coordinates and the task is to draw each edge as a curve that is both x-monotone and y-monotone while maintaining planarity. This problem is NP-hard in general [KR19]. In Upward-Rightward Planarity, the question is whether there exists a planar drawing of a directed graph in which each edge is x-monotone or y-monotone. Every planar directed graph has an upward-rightward straight-line drawing in polynomial area that can be computed in linear time [Gia+14]. The input to HV-Rectilinear Planarity is an undirected graph G with vertex-degree at most four where each edge is labeled either as horizontal or as vertical. The task is to find a planar drawing of G where each edge labeled as horizontal (vertical) is drawn as a horizontal (vertical) line segment. This problem is NP-hard in general [DLP19]. Visibility representations of two planar st-graphs using L-shapes have been considered. [ELM16]. In such a representation, one graph has a horizontal visibility representation using the vertical parts of the L-shapes, and the other graph has a vertical visibility representation using the horizontal parts of the L-shapes. Finally, a windrose graph consists of an undirected graph G and for each vertex v of G a partition of its neighbors into four sets that correspond to the four quadrants around v. A windrose drawing of G is a drawing such that for each vertex v of G each neighbor lies in the correct quadrant and all edges are represented by curves that are monotone with respect to each axis. Testing graphs for windrose planarity is NP-hard in general, but there exists a polynomial-time algorithm for graphs with a fixed underlying planar embedding [Ang+18].

We investigate a new planarity variant called *xy-planarity*. Let $G = (V, E)$ be a directed graph together with two sets X, Y with $X \cup Y = E$. In an *xy-drawing* of G each directed edge in X is drawn as a strictly increasing x-monotone curve and each directed edge in Y is drawn as a strictly increasing y-monotone curve. Hence, an xy-drawing of G is a left-to-right visualization of the partial order defined by X and a bottom-up visualization of the partial order defined by Y, i.e., it is a drawing of two posets on the same ground set V. A planar xy-drawing is an *xy-planar* drawing. The study of xy-planarity has been proposed by Angelini et al. [Ang+18]. Because xy-planarity generalizes both upward planarity and windrose planarity, we immediately obtain the following.

Theorem 34. *Testing graphs for xy-planarity is NP-complete.*

We therefore consider the restricted case where the planar embedding of G is fixed, and the Y-induced subgraph of G is a connected spanning subgraph of G whose

upward embedding is fixed. For this case we present a linear-time algorithm that determines whether G admits an xy-planar drawing. Our algorithm uses several structural insights. First, in Section 11.3, we provide a new, simple combinatorial characterization of windrose planar embeddings. From each xy-planar drawing of G we can derive an embedded windrose planar graph G^+. Using our combinatorial characterization of windrose planar embeddings we can simplify G^+. In Section 11.4 we show that in this simplified graph, every edge of the original graph G corresponds to one of four windrose planar gadgets. To test G for xy-planarity, we show in Section 11.5 how to determine in linear time whether there exists a choice of one gadget for each edge of G that leads to a windrose planar embedding. In the positive case our algorithm outputs an xy-planar drawing on a polynomial-size grid where each edge has at most three bends. If every windrose planar graph has a straight-line drawing (an open question), then each edge has at most one bend, which we show to be optimal.

11.2 Preliminaries

We use standard terminology concerning (upward) planar graph drawings and embeddings. Let $G = (V, E)$ be a connected directed graph. An upward embedding induces an *underlying embedding* of G by concatenating the left-to-right order of incoming edges and the reversed left-to-right order of outgoing edges into one counter-clockwise cyclic order around each vertex. An embedding of G is *bimodal* if the incoming (outgoing) edges appear consecutively around each vertex. The underlying embedding of an upward embedding is bimodal. Recall that a vertex of G is a *sink* (*source*) if it is incident only to incoming (outgoing) edges. A vertex that is neither a source nor a sink of G is called an *inner vertex*. Consider an upward planar drawing of G and its underlying planar embedding $\mathcal{E}$ of G. Recall that a sink/source assignment $\psi : v \mapsto (e, e')$ maps each source and sink v to two edges e, e' incident to v so that e immediately precedes e' in counter-clockwise cyclic order of edges incident to v defined by $\mathcal{E}$. This corresponds to a unique face f of $\mathcal{E}$ such that e immediately precedes e' on the facial walk of f. Thus, we say that ψ assigns v to f. The assignment is *upward consistent* if ψ assigns $n_f + 1$ vertices to one face, and $n_f - 1$ vertices to all other faces. The face to which $n_f + 1$ vertices are assigned is the outer face. From an embedding $\mathcal{E}$ and a sink/source assignment ψ an upward embedding of G can be obtained by splitting for each sink (source) v the counter-clockwise cyclic order of edges incident to v defined by $\mathcal{E}$ between the two edges e, e' with $\psi(v) = (e, e')$ to obtain the right-to-left (left-to-right) order of incoming (outgoing) edges. In the reverse direction, from an upward embedding of G a sink/source assignment ψ can be obtained as follows. Assign each sink v to (e, e') where e and e' are the rightmost

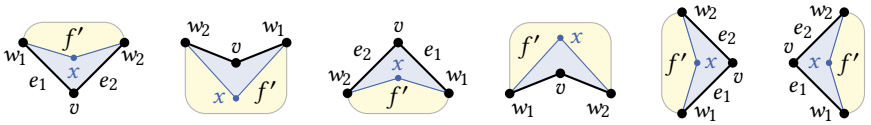


Figure 11.1: Reducing the size of the block-cutvertex tree by augmenting the graph given a cut-vertex v if v is (a) a sink, (b) a source, or (c) an inner vertex.

and leftmost incoming edge incident to v, respectively. Assign each source v to (e, e') where e and e' are the leftmost and rightmost outgoing edge incident to v, respectively. Thus, upward embeddings are equivalent to planar embeddings together with a sink/source assignment. The following was observed by Bertolazzi et al. [BDLM94] for biconnected graphs, we provide a straight-forward extension to simply-connected graphs.

Lemma 74. *Let G be a connected directed acyclic graph together with an embedding $\mathcal{E}$. There exists an upward planar embedding of G whose underlying embedding is $\mathcal{E}$ if and only if $\mathcal{E}$ is planar and bimodal, and it admits an upward consistent assignment.*

Proof. Bertolazzi et al. have proven the statement for biconnected graphs [BDLM94]. We extend it to simply connected graphs by induction over the number of maximal biconnected components of G. If there is one such component the result of Bertolazzi et al. applies. If G consists of a single vertex or a single edge, the statement holds trivially. Let v be a cutvertex of G. Then there exist two edges $e_1 = \{v, w_1\}$ and $e_2 = \{v, w_2\}$ incident to v such that e_1 immediately follows e_2 in the counter-clockwise order of edges around v and w_1 and w_2 belong to different maximal biconnected components of G.

Let Γ be an upward planar drawing of G with underlying embedding $\mathcal{E}$. Let f be a face such that e_1 and e_2 appear consecutively on its facial walk. Moving closely along e_1 and e_2 in f, insert a new vertex x and connect it to w_1 and w_2 as shown in Figure 11.1. Note that this case distinction can be made on the basis of Γ, although it could not be made solely on the basis of $\mathcal{E}$. This separates f into one face of size four (shaded in blue), and another face f' (shaded in yellow). Let G' denote the graph obtained in this way, and let Γ' denote the drawing obtained in this way. In G' the vertices w_1 and w_2 belong to the same maximal biconnected component. Thus, G' has at least one less maximal biconnected component than G. By induction, the underlying planar embedding $\mathcal{E}'$ of Γ' then admits an upward consistent assignment ψ'. By construction of the gadgets the number of face sources and face sinks of f equals the number of face sources and face sinks of f', respectively. To

obtain an upward consistent assignment ψ of $\mathcal{E}$ define $\psi : v \mapsto (e_2, e_1)$ if and only if $\psi' : v \mapsto (\{x, w_2\}, \{x, w_1\})$.

Now assume that $\mathcal{E}$ admits an upward consistent assignment ψ. Repeat a similar argument as above. This time, the case distinction is made not based on Γ, but based on ψ, obtaining an upward consistent assignment ψ' of $\mathcal{E}'$. By induction there exists an upward planar embedding $\mathcal{U}'$ of G' whose underlying embedding is $\mathcal{E}'$. Removing x from $\mathcal{U}'$ gives an upward planar embedding $\mathcal{U}$ of G whose underlying embedding is $\mathcal{E}$. □

11.3 Combinatorial View of Windrose Planarity

A *windrose graph* is a directed graph G whose edges are labeled as either north-west (NW) or north-east (NE). A *windrose drawing* of G is an upward drawing of G where all NW (NE) edges decrease (increase) monotonically along the x-axis. In this way, the neighbors of each vertex are partitioned into the four quadrants of the plane around v. An upward planar embedding of G is *windrose planar* if it is induced by a windrose planar drawing of G. Let $\mathcal{U}$ be an upward planar embedding of G and let v be a vertex of G. We say $\mathcal{U}$ is *windrose consistent at* v if (i) the NW edges precede the NE edges in the left-to-right order of outgoing edges incident to v, and (ii) the NE edges precede the NW edges in the left-to-right order of incoming edges incident to v. We say that $\mathcal{U}$ is *windrose consistent* if it is windrose consistent at all vertices of G and show the following.

Lemma 75. *Let G be a directed graph together with an upward planar embedding $\mathcal{U}$. Then $\mathcal{U}$ is a windrose planar embedding of G if and only if $\mathcal{U}$ is windrose consistent.*

Proof. If $\mathcal{U}$ is a windrose planar embedding then $\mathcal{U}$ is windrose consistent. For the reverse direction, assume that $\mathcal{U}$ is windrose consistent. We show that $\mathcal{U}$ is windrose planar by induction over the number of NW edges in G. Every upward planar graph admits a straight-line drawing [DT88]. If G contains no NW edge, first vertically stretch such a straight-line drawing so that all slopes lie in the interval $(\pi/4, 3\pi/4)$, then rotate the stretched drawing so that all slopes lie in the interval $(0, \pi/2)$ to obtain a windrose planar drawing of G. For the inductive step, the idea is to find an NW edge in G that can be relabeled as NE, where the result is a graph G' such that $\mathcal{U}$ is a windrose consistent embedding of G'. To this end, construct an *edge dependency graph* $D = (E, A)$ as follows. The edges of G are the nodes of D. Construct the directed arcs of D as follows. For each vertex v of G, consider the outgoing edges $e_1^+, e_2^+, \ldots, e_m^+$ in the left-to-right order prescribed by $\mathcal{U}$. For $1 \leq i < m$ add the arc (e_i^+, e_{i+1}^+) to D. Moreover, consider the incoming edges $e_1^-, e_2^-, \ldots, e_n^-$ in the left-to-right order prescribed by $\mathcal{U}$. For $1 \leq i < n$ add the arc (e_{i+1}^-, e_i^-) to D. Consider an upward

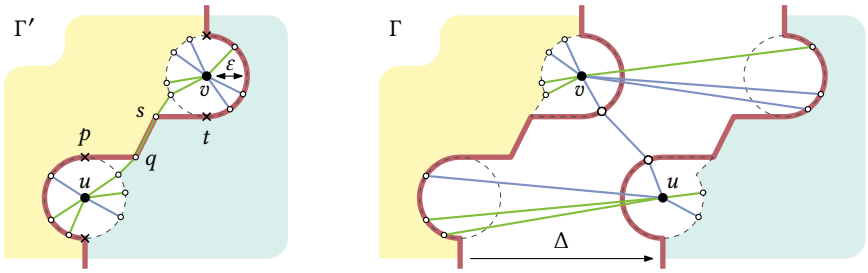


Figure 11.2: Transforming a windrose planar drawing Γ' of G' into a windrose planar drawing Γ of G.

planar straight-line drawing Γ of G. An arc (e, f) of the dependency graph D can be interpreted as the edge e having a greater slope than the edge f in Γ. This directly implies that D is acyclic. The edge dependency graph D contains no arc from an NE edge to an NW edge by construction. Thus, if G contains an NW edge, there exists an NW edge e whose predecessors are all NW edges and whose successors in D are all NE edges. Let G' denote the graph obtained from G by relabeling e as NE. Observe that $\mathcal{U}$ is a windrose consistent upward planar embedding of G'. We show how to transform a windrose planar drawing Γ' of G' whose upward embedding is $\mathcal{U}$ into a windrose planar drawing Γ of G whose upward embedding is $\mathcal{U}$. We may assume that the vertices are in general position. Otherwise, this can be achieved by slightly perturbing the vertices in the drawing. See Figure 11.2. By convention, we draw NW edges in blue and NE edges in green.

For $\varepsilon > 0$ construct a y-monotone curve c in Γ' as follows; see the red curve in Figure 11.2. The curve c extends vertically down infinitely from the point below u at distance ε. It contains the left half-circle of radius ε centered at u. It contains the horizontal segment from the highest point p of the half-circle up to the point q where it meets e. Symmetrically, c extends vertically up infinitely from the point above v at distance ε. It contains the right half-circle of radius ε centered at v. It contains the horizontal segment from the lowest point of the half-circle t up to the point s where it meets e. Finally, c follows e from q to s. There exists an $\varepsilon > 0$ so that (i) c intersects no vertex, (ii) the circles of radius ε around u and v intersect only line segments that have that vertex as an endpoint, and (iii) the segments from p to q and from s to t do not intersect any edge, except for e. For property (i), note that the vertices have been assumed to be in general position in Γ'. For property (ii), recall that edges are drawn as finite polygonal chains. For property (iii), observe that all predecessors of e in D are NW edges.

Shift the area to the right of c further right by an offset Δ so that u lies to the

right of v. Then u and v can be connected by an NW edge. The segments between u and the intersection points on the circle of radius ε centered at u can be drawn as straight-line segments, preserving their monotonicity with respect to the x-axis. The same holds true for the segments whose endpoint is v. Finally, c may intersect edges on the vertical segments below u and above v. Because of property (i) shifting creates intermediate horizontal segments which can be made non-horizontal by slightly perturbing their endpoints. □

11.4 From xy-Drawings to Windrose Drawings

Let $G = (V, E)$ be a directed graph together with sets X, Y such that $X \cup Y = E$. Define $G|_Y$ as the subgraph of G induced by the edges in Y. Further, let Γ denote an xy-drawing of G. Recall that edges are drawn as finite polygonal chains. Define G^+ as the graph obtained from G by subdividing the edges of G at each bend in Γ such that each directed edge (u, v) of G^+ corresponds to an upward straight-line segment from u to v (u is below v) in Γ. Label (u, v) as NW (NE) if the corresponding segment decreases (increases) along the x-axis. Then G^+ is a windrose graph together with a windrose planar drawing Γ^+. Let $\mathcal{E}^+$ denote the windrose planar embedding induced by Γ^+.

11.4.1 Simplifying Windrose Planar Embeddings

Each edge (u, v) of G in Y corresponds to a path $(u = y_1, y_2, \ldots, y_n = v)$ in G^+. For $1 \leq i < n$ the edge connecting y_i and y_{i+1} is oriented from y_i to y_{i+1} and is either an NE edge or an NW edge. We can simplify $G^+$and $\mathcal{E}^+$. Similarly-labeled edges $(y_i, y_{i+1}), (y_j, y_{j+1})$ with $1 \leq i < j < n$ can be replaced by one edge (y_i, y_{j+1}) with the same label. We argue that the arising embedded graph is still windrose planar. First, the order of labels around vertices y_i and y_{j+1} has not been changed, and the remaining vertices have not been affected, as a result the embedding is still windrose consistent. Second, we show that the embedding is still upward planar. This is directly implied by two facts. First, vertices $y_{i+1}, \ldots, y_j$ are neither sources nor sinks. And second, the sink/source assignment of y_i and y_{j+1} has not been changed. Thereby the number of face sources and face sinks and the number of sources and sinks assigned to every face also stay the same. Since $\mathcal{E}^+$ is upward consistent, then the simplified embedding is upward consistent, too. By Lemma 75, the simplified embedding admits a windrose planar drawing (it is also an xy-planar drawing of G). We can repeat this, until every edge $(u, v) \in Y$ is represented with either a single edge, or with two edges with different labels. In the following, we want that every gadget consists of exactly two edges. For this purpose, if the path from u to v consists

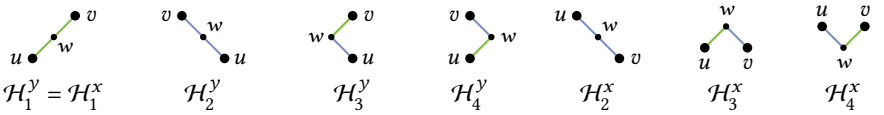


Figure 11.3: The gadgets $\mathcal{H}_1^x, \ldots, \mathcal{H}_4^x$ and $\mathcal{H}_1^y, \ldots, \mathcal{H}_4^y$ that are used to represent an x-monotone edge and an y-monotone edge (u, v), respectively. NW (NE) edges are drawn in blue (green).

of a single edge, we subdivide it into two edges with the same label. See Figure 11.3 for the four possible gadgets $\mathcal{H}_1^y, \ldots, \mathcal{H}_4^y$.

Each edge (u, v) of G in X corresponds to a path $(u = x_1, x_2, \ldots, x_n = v)$ in G^+. For $1 \leq i < n$ the edge connecting x_i and x_{i+1} in G^+ is either a NE edge oriented from x_i to x_{i+1}, or an NW edge oriented from x_{i+1} to x_i. Again, we can simplify $\mathcal{E}^+$. For $1 < i < n$ if the edges connecting x_{i-1} and x_i, and x_i and x_{i+1} are oriented in the same direction they also are similarly labeled and the same argument as above can be used to replace x_i and its incident edges by an edge connecting x_{i-1} and x_{i+1}. Now consider the case that for $1 \leq i \leq n-3$ the edges $e_1 = (x_i, x_{i+1})$ and $e_3 = (x_{i+2}, x_{i+3})$ are labeled as NE and the edge $e_2 = (x_{i+1}, x_{i+2})$ is labeled as NW. Because Γ is an xy-drawing, the sink/source assignment induced by $\mathcal{E}^+$ assigns x_{i+1} to (e_2, e_1) and x_{i+2} to (e_2, e_3) (in terms of upward planarity as defined in Section 11.2). Replacing the vertices x_{i+1}, x_{i+2} and their incident edges by a single edge (x_i, x_{i+3}) labeled as NE reduces the number of face sinks by one and it reduces the number of face sources of both incident faces by one. The number of assigned face sinks and face sources is also reduced by one. Thereby, the sink/source assignment is still consistent and the embedding is upward planar. Together with the arguments for edges of G in Y this shows that the simplified embedding remains a windrose planar embedding. See Figure 11.3 for the four possible gadgets $\mathcal{H}_1^x, \ldots, \mathcal{H}_4^x$.

Let G^* and $\mathcal{E}^*$ denote the simplified windrose graph and windrose planar embedding. Every windrose planar drawing of G^* with embedding $\mathcal{E}^*$ induces an xy-planar drawing Γ' of G such that Γ and Γ' induce the same planar embedding of G and the same upward planar embedding of $G|_Y$. Note that G^* is obtained from G by replacing (i) each edge of G in $X \setminus Y$ with a gadget in $\mathcal{H}^x$, (ii) each edge of G in $Y \setminus X$ with a gadget in $\mathcal{H}^y$, and (iii) each edge of G in $X \cap Y$ with the (unique) gadget in $\mathcal{H}^x \cap \mathcal{H}^y$ (where $\mathcal{H}^x = \{\mathcal{H}_1^x, \mathcal{H}_2^x, \mathcal{H}_3^x, \mathcal{H}_4^x\}$ and $\mathcal{H}^y = \{\mathcal{H}_1^y, \mathcal{H}_2^y, \mathcal{H}_3^y, \mathcal{H}_4^y\}$). We say that a windrose graph obtained from G by such a gadget replacement is *derived* from G. A windrose planar embedding of a graph derived from G induces a planar embedding of G and an upward planar embedding of $G|_Y$. We have shown the following.

Lemma 76. *Let $G = (V, E)$ be a directed graph with sets X, Y such that $X \cup Y = E$.*

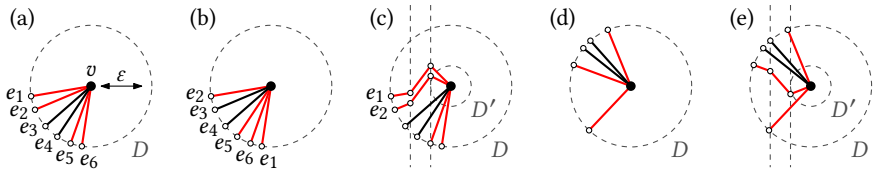


Figure 11.4: Two xy-drawings that have the same cyclic order of edges around the boundary of D and assignment of line segments to quadrants, but distinct upward planar embeddings (a, b). Edges in X are drawn in red, edges in Y are drawn in black. The xy-drawing (a) can be locally modified to obtain a special xy-drawing (c) that admits no such ambiguities. Modifying a drawing where Property (2) does not hold true works symmetrically (d, e).

This graph admits an xy-planar drawing with planar embedding $\mathcal{E}$ of G and upward planar embedding $\mathcal{U}$ of $G|_Y$ if and only if there exists a derived graph G^* of G with a windrose planar embedding that induces $\mathcal{E}$ and $\mathcal{U}$.

11.4.2 Special Windrose Planar Embeddings

Inspired by Lemma 76, part of the approach to test G for xy-planarity will be to use Lemma 74 to test for every edge $e \in X$ and each gadget $\mathcal{H}_i^x \in \mathcal{H}^x$ whether replacing e with $\mathcal{H}_i^x$ leads to an upward planar embedding of $G|_Y + e$. If all edges incident to e lie in the same quadrant, it is not right away possible to derive the upward planar embedding of $G|_Y + e$ just from the upward planar embedding of $G|_Y$ and the gadget choice $\mathcal{H}_i^x$. For an example, consider Figure 11.4 (a, b). Even though e_1 might be replaced by the same gadget the sink assignment of $G|_Y + e_1$ in (a) is $\psi : v \mapsto (e_4, e_1)$, whereas in (b) it is $\psi : v \mapsto (e_1, e_3)$. To prevent such ambiguities, we introduce the notion of special xy-drawings.

In any xy-planar drawing Γ of G there exists some $\varepsilon > 0$ so that the disk D of radius ε centered at v does not contain any vertex other than v, intersects only edges incident to v, and does not contain any point where an edge bends. A counterclockwise traversal of the boundary of D gives four (possibly empty) linear orders of the edges in each quadrant. We say that Γ is *special* if for each vertex v of G the following four properties hold true. (1) If v has only incoming edges in X and incoming edges in Y, then the first edge in the southwestern quadrant is in Y. (2) If v has only incoming edges in X and outgoing edges in Y, then the last edge in the northwestern quadrant is in Y. (3) If v has only outgoing edges in X and incoming edges in Y, then the last edge in the southeastern quadrant is in Y. (4) If v has only outgoing edges in X and outgoing edges in Y, then the first edge in the northeastern

quadrant is in Y.

If the drawing Γ is not special, then we can modify it locally around each vertex where one of the four properties does not hold to obtain a special drawing. Consider the case that v is a vertex of G that has only incoming edges in X and Y but the first edge in the southwestern quadrant is not in Y, i.e., Property (1) does not hold true for v. See Figure 11.4 (a). Introduce two new bends on each edge $e_1, e_2, \ldots, e_n$ in X that precede the first edge in Y in the southwestern quadrant so that the line segments incident to v lie in the northwestern quadrant. This preserves the x-monotonicity of the drawing and ensures Property (1). See Figure 11.4 (c) for the modified drawing, where Property (1) holds true (consider the smaller disk D'). The other three properties can be ensured symmetrically; see Figure 11.4 (d, e) for an example of how to ensure Property (2).

Note how in the special windrose drawings in Figure 11.4 (c, e) for each red edge e the upward planar embedding of $G|_Y + e$ can be derived just from the upward planar embedding of $G|_Y$ and the gadget choice $\mathcal{H}_i^x$ for e. A windrose planar embedding is *special* if it is derived from a special xy-planar drawing. Observe that simplifying the windrose planar embedding as explained in the previous section does not alter edges incident to non-subdivision vertices. Therefore, Lemma 76 can be strengthened to special windrose planar embeddings as follows.

Lemma 77. *Let $G = (V, E)$ be a directed graph with sets X, Y such that $X \cup Y = E$. This graph admits an xy-planar drawing with planar embedding $\mathcal{E}$ of G and upward planar embedding $\mathcal{U}$ of $G|_Y$ if and only if there exists a derived graph G^* of G together with a special windrose planar embedding that induces $\mathcal{E}$ and $\mathcal{U}$.*

Because every windrose planar graph admits a polyline drawing in polynomial area with at most one bend per edge [Ang+18] we immediately obtain the following.

Corollary 10. *Every xy-planar graph admits a polyline drawing in polynomial area with at most three bends per edge.*

If every embedded windrose planar graph admitted a straight-line drawing, then every xy-planar graph would admit a polyline drawing with at most one bend per edge. Not every xy-planar graph admits an xy-planar straight-line drawing; see Figure 11.5. Lemma 77 also motivates our approach of testing G for xy-planarity by testing whether there exists a replacement of the edges of G with gadgets that respects the given planar and upward planar embeddings and yields a special windrose planar embedding.

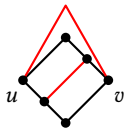


Figure 11.5: An xy-planar graph that does not admit an xy-planar straight-line drawing.

11.5 An xy-Planarity Testing Algorithm

Let G be a directed graph together with edge sets X, Y and let e be an edge in X. Define the *gadget candidate set* $\mathcal{H}(e)$ as the subset of $\mathcal{H}^x = \{\mathcal{H}_1^x, \mathcal{H}_2^x, \mathcal{H}_3^x, \mathcal{H}_4^x\}$ that contains each $\mathcal{H}_i^x \in \mathcal{H}^x$ so that the embedding $\mathcal{U}+\mathcal{H}_i^x$ is an upward planar embedding of $G|_Y+e$. Recall that Lemma 77 justifies that we can limit our considerations to windrose planar embeddings that are special, which lets us unambiguously derive the upward planar embedding $\mathcal{U} + \mathcal{H}_i^x$ of $G|_Y + e$ from the fixed upward planar embedding $\mathcal{U}$ of $G|_Y$ and the gadget choice $\mathcal{H}_i^x$ for e. This is needed to test for upward planarity using Lemma 74. For $e \in X$ the gadget candidate set $\mathcal{H}(e)$ can be computed by tentatively replacing e with each $\mathcal{H}_i^x$ and then running the upward planarity test for fixed upward embeddings. In fact, this can even be done in overall linear time for all edges in X. To this end, choose for each face f of $\mathcal{U}$ some vertex v and let $v_1, v_2, \ldots, v_k$ denote the facial walk of f. Traverse the facial walk of f once to compute for each $1 \leq i \leq k$ the number of face sources on the path $v_1, v_2, \ldots, v_i$, and the number of sources and sinks of G on the path $v_1, v_2, \ldots, v_i$ assigned to f. This is possible in linear time for all faces of $\mathcal{U}$. Now consider the insertion of some edge $(v_a, v_b) \in X$ into f, splitting f into two faces f_1, f_2. Using the previously calculated values, it can be checked in constant time whether f_1 and f_2 are upward consistent in the sense of Lemma 74. Together with the previous argument we conclude the following.

Lemma 78. *The gadget candidate sets for all edges in X can be computed in linear time.*

11.5.1 Finding a Windrose Planar Derived Graph

For each edge e of G add the variables e^{NE}, e^{NW}, e^{SW}, e^{SE}, and add the clause $\neg x \vee \neg y$ for each pair x, y of distinct such variables. This means that every edge of G^* is assigned to at most one quadrant. Make sure that every edge of G^* is assigned to at least one quadrant as follows. Let (u, v) be an edge of G and let w denote the vertex of the gadget in G^* that replaces (u, v). If $(u, v) \in X$, then add the clauses $(u, w)^{\mathrm{NE}} \vee (u, w)^{\mathrm{SE}}$ so that (u, v) exits u in the east, and $(w, v)^{\mathrm{NW}} \vee (w, v)^{\mathrm{SW}}$ so that (u, v) enters v from

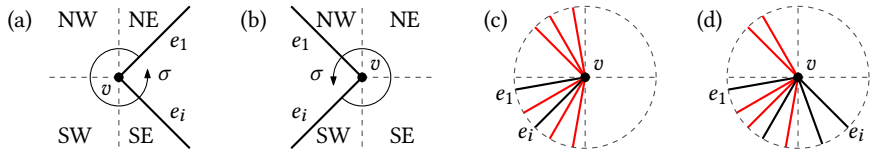


Figure 11.6: The situation around a vertex v that has both an outgoing edge e_1 and an incoming edge e_i in Y (a, b). An example of a vertex v that has only incoming edges in X and Y (c, d). Here e_1 and e_i are the leftmost and rightmost edges with respect to the fixed upward embedding.

the west. Next, if $(u, v) \in Y$, then add the clauses $(u, w)^{\text{NE}} \vee (u, w)^{\text{NW}}$ so that (u, v) exits u in the north, and $(w, v)^{\text{SW}} \vee (w, v)^{\text{SE}}$ so that (u, v) enters v from the south.

Placing the edges of G^* into quadrants induces a unique gadget that replaces each edge of G. Let $e = (u, v)$ be an edge of G. Each gadget $\mathcal{H}_i^x \notin \mathcal{H}(e)$ places the edge (u, w) in quadrant p and the edge (w, v) in quadrant q. Moreover, include the clause $\neg((u, w)^p \wedge (w, v)^q) = \neg(u, w)^p \vee \neg(w, v)^q$, this would prevent the gadget $\mathcal{H}_i^x$ from being induced. Because no other gadget places the edge (u, w) in quadrant p and (w, v) in q adding this clause does not prevent any other gadget from being induced. Add such a clause for each gadget $\mathcal{H}_i^x \notin \mathcal{H}(e)$.

The last step is to ensure windrose consistency at each vertex v of G^*. Use the combinatorial criterion from Lemma 75. Since we consider gadgets with a prescribed assignment at w, we implicitly ensure that the embedding is windrose consistent around such vertices (see Figure 11.3). Let v be a non-subdivision vertex of G^*, i.e., v is also a vertex of G. Consider the case that v has incoming and outgoing edges in Y. Let $\sigma = e_1, e_2, \ldots, e_i, \ldots, e_n, e_1$ denote the counter-clockwise cyclic order of edges incident to v in G^* such that e_1 is a subdivision edge of an outgoing edge in Y and e_i is a subdivision edge of an incoming edge in Y. To achieve windrose consistency the edges must be labeled as NE, then NW, then SW and finally SE in the sequence $e_1, \ldots, e_i$. This can be ensured with the constraints

$$e_j^{\text{NW}} \Rightarrow \neg e_{j+1}^{\text{NE}}$$
$$e_j^{\text{SW}} \Rightarrow \neg e_{j+1}^{\text{NW}} \quad \text{and} \quad e_j^{\text{SW}} \Rightarrow \neg e_{j+1}^{\text{NE}}$$
$$e_j^{\text{SE}} \Rightarrow \neg e_{j+1}^{\text{SW}} \quad \text{and} \quad e_j^{\text{SW}} \Rightarrow \neg e_{j+1}^{\text{NW}} \quad \text{and} \quad e_j^{\text{SW}} \Rightarrow \neg e_{j+1}^{\text{NE}}$$

for $1 \leq j < i$; see Figure 11.6 (a). Similarly, in $e_i, \ldots, e_n, e_1$ edges must be labeled as SW, then SE, then NE and finally NW; see Figure 11.6 (b). A symmetric argument holds for vertices of G that have both incoming and outgoing edges in X.

The remaining case consists of four subcases where v has only incoming or only outgoing edges in Y, and only incoming or only outgoing edges in X. Consider the

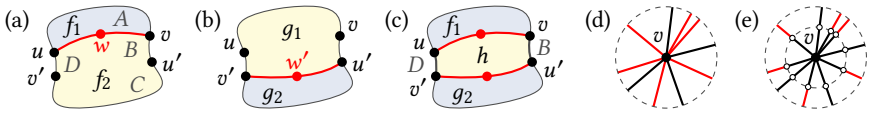


Figure 11.7: Proof of Lemma 79. If f_1, f_2, f_1', f_2' are upward planar, then so is h (a–c). The case of adjacent edges reduces to the case of independent edges (d, e).

case that v has only incoming edges in Y and only incoming edges in X (the other cases are symmetric); see Figure 11.6 (c, d). Let $\sigma = e_1, e_2, \ldots, e_i, \ldots, e_n, e_1$ be the counter-clockwise cyclic order of edges incident to v in G^* such that e_1 and e_i are subdivision edges of the leftmost and rightmost incoming edges in Y in the fixed upward planar embedding of $G|_Y$. Add the constraints $e_j^{\text{SE}} \vee e_j^{\text{SW}}$ and $e_j^{\text{SE}} \Rightarrow e_{j+1}^{\text{SE}}$ for $1 \leq j < i$, the constraint $e_i^{\text{SE}} \Rightarrow e_{i+1}^{\text{NW}}$, and, because we seek special embeddings, the constraints $e_j^{\text{NW}} \Rightarrow e_{j+1}^{\text{NW}}$ for $i < j < n$.

11.5.2 Correctness

Every windrose graph derived from G induces a solution of the 2-SAT instance. Every solution of the 2-SAT instance induces a windrose graph G^* derived from G together with a windrose planar embedding. For each edge $e \in X$ a solution to the 2-SAT instance induces a replacement gadget $\mathcal{H}_i^x$ so that $\mathcal{U} + \mathcal{H}_i^x$ is an upward planar embedding of $G|_Y + e$. The final component to our xy-planarity test is to show that even though we tested the gadget candidates individually, the fact that the 2-SAT instance is satisfiable implies that inserting for each edge in X its replacement gadget leads to an upward planar embedding of G^*.

Lemma 79. *Let $G = (V, E)$ be a directed graph with sets X, Y such that $X \cup Y = E$. Let $\mathcal{E}$ be an embedding of G and let $\mathcal{U}$ be an upward planar embedding of $G|_Y$. If the corresponding 2-SAT instance is satisfiable, then the embedding $\mathcal{U}^*$ of the graph G^* induced by a solution of this instance is upward planar.*

Proof. We show that $\mathcal{U}^*$ is upward planar inductively by showing that inserting the gadgets one by one preserves upward planarity. Edges that are inserted into different faces of $\mathcal{U}$ can be treated separately. First, consider two independent edges $e = \{u, v\}, e' = \{u', v'\}$ (directed appropriately) in X that are inserted into the same face f of $\mathcal{U}$; see Figure 11.7. Because $\mathcal{E}$ is planar the endpoints of e and e' do not alternate. Let u, A, v, B, u', C, v', D denote the facial walk of f, where A, B, C, D are sets of vertices. The solution of the 2-SAT instance prescribes gadgets $\mathcal{H}_i^x, \mathcal{H}_j^x$ that are inserted for e, e', respectively. Let w and w' denote the subdivision vertex of $\mathcal{H}_i^x$ and $\mathcal{H}_j^x$, respectively. Inserting $\mathcal{H}_i^x$ for e splits f into two faces f_1, f_2 such that

the facial walk of f_1 is u, A, v, w and the facial walk of f_2 is u, w, v, B, u', C, v', D; see Figure 11.7 (a). Similarly, inserting $\mathcal{H}_j^x$ for e' splits f into two faces g_1, g_2 such that the facial walk of g_1 is $u', w', v', D, u, A, v, B$ and the facial of g_2 is u', C, v', w'; see Figure 11.7 (b). Finally, inserting both $\mathcal{H}_i^x$ and $\mathcal{H}_j^x$ splits f into three faces, namely f_1, g_2 and a face h whose facial walk is $u, w, v, B, u', w', v', D$; see Figure 11.7 (c). From the construction of the gadget candidate sets we know that f, f_1, f_2, g_1, g_2 are all upward consistent. We are left to show that h is upward consistent as well. To this end, we use Lemma 74.

Let z be some face of an upward embedding and let Z be a set of vertices on the facial walk of z. For the scope of this proof, let $\psi(Z, z)$ denote the number of sources and sinks in Z assigned to z. We have the following.

$$\psi(f) = \psi(B \cup D, f) + \psi(A, f) + \psi(C, f) + \psi(\{u, v\}, f) + \psi(\{u', v'\}, f) \tag{11.1}$$

$$\begin{aligned}\psi(f_2) &= \psi(B \cup D, f_2) + \psi(C, f_2) + \psi(\{u, v, w\}, f_2) + \psi(\{u', v'\}, f_2)\\ &= \psi(B \cup D, f) + \psi(C, f) + \psi(\{u, v, w\}, f_2) + \psi(\{u', v'\}, f)\end{aligned} \tag{11.2}$$

$$\begin{aligned}\psi(g_1) &= \psi(B \cup D, g_1) + \psi(A, g_1) + \psi(\{u, v\}, g_1) + \psi(\{u', v', w'\}, g_1)\\ &= \psi(B \cup D, f) + \psi(A, f) + \psi(\{u, v\}, f) + \psi(\{u', v', w'\}, g_1)\end{aligned} \tag{11.3}$$

$$\begin{aligned}\psi(h) &= \psi(B \cup D, h) + \psi(\{u, v, w\}, h) + \psi(\{u', v', w'\}, h)\\ &= \psi(B \cup D, f) + \psi(\{u, v, w\}, f_2) + \psi(\{u', v', w'\}, g_1)\end{aligned} \tag{11.4}$$

Observe that equations (11.2)–(11.4) only hold because the upward embedding $\mathcal{U}$ of $G|_Y$ is fixed and the edges e, e' are independent. Adding (11.2) and (11.3) and then subtracting (11.1) gives (11.4) (gray terms remain, terms of the same color cancel out each other). This shows $\psi(h) = \psi(f_2) + \psi(g_1) - \psi(f)$.

Again, let z be some face of an upward embedding and let Z be a set of vertices on the facial walk of z. For the scope of this proof, let $n(Z, z)$ denote the number of face

sources of z in Z. We have the following.

$$n_f = n(B \cup D, f) + n(A, f) + n(C, f) + n(\{u, v\}, f) + n(\{u', v'\}, f) \tag{11.5}$$

$$\begin{aligned} n_{f_2} &= n(B \cup D, f_2) + n(C, f_2) + n(\{u, v, w\}, f_2) + n(\{u', v'\}, f_2) \\ &= n(B \cup D, f) + n(C, f) + n(\{u, v, w\}, f_2) + n(\{u', v'\}, f) \end{aligned} \tag{11.6}$$

$$\begin{aligned} n_{g_1} &= n(B \cup D, g_1) + n(A, g_1) + n(\{u, v\}, g_1) + n(\{u', v', w'\}, g_1) \\ &= n(B \cup D, f) + n(A, f) + n(\{u, v\}, f) + n(\{u', v', w'\}, g_1) \end{aligned} \tag{11.7}$$

$$\begin{aligned} n_h &= n(B \cup D, h) + n(\{u, v, w\}, h) + n(\{u', v', w'\}, h) \\ &= n(B \cup D, f) + n(\{u, v, w\}, f_2) + n(\{u', v', w'\}, g_1) \end{aligned} \tag{11.8}$$

Similarly, adding (11.6) and (11.7) and then subtracting (11.5) gives (11.8), which shows that $n_h = n_{f_2} + n_{g_1} - n_f$. Let $k = 1$ if f is the outer face and let $k = -1$ if f is an inner face. Lemma 74 gives

$$\psi(f) = n_f + k \tag{11.9}$$

$$\psi(f_2) = n_{f_2} + k \tag{11.10}$$

$$\psi(g_1) = n_{g_1} + k. \tag{11.11}$$

Adding (11.10) and (11.11) and then subtracting (11.9) gives

$$\psi(f_2) + \psi(g_1) - \psi(f) = n_{f_2} + n_{g_1} - n_f + k,$$

which, together with $\psi(h) = \psi(f_2) + \psi(g_1) - \psi(f)$ and $n_h = n_{f_2} + n_{g_1} - n_f$ as shown above, gives $\psi(h) = n_h + k$. This shows that h is upward consistent in the sense of Lemma 74.

The same idea extends to the case of faces that are bounded by gadgets corresponding to more than two independent edges. The case of adjacent edges reduces to the case of independent edges by subdividing for each edge $(u, v) \in X$ the edges of the gadget that replace (u, v) and treating the subdivision edges incident to u and v as edges in Y; see Figure 11.7 (d, e). The choice of each gadget specifies the direction of the edges incident to u and v, and, possibly, the assignment of u and v to a face. □

Lem. 79 gives that $\mathcal{U}^*$ is upward planar. The clauses of the 2-SAT instance are designed such that (i) $\mathcal{U}^*$ is windrose consistent by Lem. 75, i.e., $\mathcal{U}^*$ is windrose planar, and (ii) $\mathcal{U}^*$ is special. Lem. 77 gives that G is xy-planar if and only if the 2-SAT instance is satisfiable. To compute an xy-planar drawing of G, use the linear-time windrose-planar drawing algorithm of Angelini et al. [Ang+18] to compute a windrose planar drawing of G^*, which induces an xy-planar drawing of G.

Theorem 35. *Let $G = (V, E)$ be a directed graph together with subsets X, Y of its edges E such that $X \cup Y = E$, a planar embedding $\mathcal{E}$ of G and an upward planar embedding $\mathcal{U}$ of $G|_Y$. It can be tested in linear time whether there exists an xy-planar drawing of G whose underlying planar embedding is $\mathcal{E}$ and whose underlying upward planar embedding restricted to $G|_Y$ is $\mathcal{U}$. In the positive case, such a drawing can be computed in linear time as well.*

11.6 Conclusion

We introduced and studied the concept of xy-planarity which is particularly suitable to draw two posets on the same ground set, one from bottom to top and the other from left to right. Every xy-planar drawing of a graph G induces a derived windrose planar graph G^*, which implies that every xy-planar graph admits a polyline drawing in polynomial area with at most three bends per edge. Because xy-planarity generalizes both upward planarity and windrose planarity, xy-planarity testing is NP-complete in general. We considered the case that the upward part $G|_Y$ is a connected spanning subgraph of G whose upward embedding $\mathcal{U}$ is fixed, and that the planar embedding $\mathcal{E}$ of G is fixed as well. For this case, we have given a linear-time xy-planarity testing algorithm. It uses the connection to derived windrose planar graphs, a novel combinatorial view of windrose planarity and a careful analysis of upward planar embeddings and windrose planar embeddings.

12 Conclusion and Open Problems

In this thesis, we contributed to the study of planarity variants for directed graphs. The level planarity testing algorithms in Part I clear up some of the problems of the existing approaches and handle the case when the embedding of the input graph is fixed. In Part II we developed data structures that represent level planar and upward planar embeddings and use them to efficiently solve constrained embedding problems. Finally, in Part III, we introduced new drawing styles for directed planar graphs that go beyond the well-established notions of level planarity and upward planarity and developed algorithms to compute such drawings.

Part I. We started with Chapter 3, where we found that level planarity testing with fixed embedding is feasible in linear time. A key tool to achieve this is our untangling lemma, which relies on fixed y-coordinates of the vertices. In contrast, radial upward and upward planarity with fixed embedding (i.e., the cyclic order of edges around each vertex is fixed) can only be solved in $O(n \log^3 n)$ and $O(n^{3/2})$ time, respectively. It is an open question whether these problems can be solved in linear time. One attempt could be to see whether it is possible to make the faces of a directed graph with fixed planar embedding small by inserting new edges without affecting its upward planarity with respect to the fixed planar embedding. This would require a more precise understanding of the interactions between planarity and upward planarity.

In Chapter 4, we have shown equivalence of the correctness of a 2-SAT formulation of level planarity and a Hanani-Tutte-style topological characterization of level planarity. This constitutes a complete correctness proof separate from the one due to Randerath et al., whose completeness has been doubted by some authors [FPSŠ13].

Moreover, using a Hanani-Tutte-style topological characterization of radial level planarity, we have shown that this equivalence carries over to radial level planarity. This yields a novel very simple 2-SAT formulation of radial level planarity. Unfortunately, both algorithms are simply tests, so they do not output (radial) level planar embeddings. Can they be extended to output such an embedding? We note that the disputed proof due to Randerath et al. works precisely in this way, namely by translating a satisfying assignment of the 2-SAT into a level planar embedding.

In Chapter 5, we presented a linear-time (radial) level-planarity testing algorithm. It follows the same well-established general approach as previous research, but differs in some key parts. In particular, we discard the PQ-tree in favor of the more modern PC-tree, which we then augment to keep track of apices instead of just spaces. In our opinion, our algorithm is the first one that comes with a convincing proof of correctness. Indeed, we treat an important case which we believe Bachmaier et al. [BBF05] have missed. This would make our algorithm the first correct linear-time algorithm for radial level planarity testing and radial level planar embedding. Although our algorithm is more simple than the previously existing ones, it is still far from trivial. It would be interesting to see a working implementation of our algorithm as a witness to its practicality.

Part II. In this part, we approached constrained embedding problems from two angles. Our first approach was to augment PC-trees to be able to handle partial orders of their leaves, which lets us find partial and constrained level-planar embeddings. This approach is limited to single-source level graphs. It would be interesting to consider whether this approach can be extended to the general case with multiple sources. We have shown that the general multi-source case is NP-hard, so a polynomial-time algorithm is unlikely to exist. However, the problem might be fixed-parameter tractable, e.g., with respect to the number of sources and sinks of the input graph. One idea for such an algorithm would be to combine our partial level planarity algorithm with our linear-time level planarity testing algorithm. For example, it could be feasible to keep track of several apex candidates between each pair of leaves $x \triangleright y$ of the PC-tree, and testing multiple such candidates during one merge operation.

Our second approach to handle constrained embedding problems was to develop SPQR-tree-like embedding representations for biconnected single-source level-planar graphs and upward-planar graphs. These representations offer two major advantages over PC-trees. One, they enable a global view on all level planar or upward planar embeddings of a graph. Two, they can be used almost as drop-in replacements to translate existing constrained embedding algorithms from the planar setting to the level planar and upward planar settings with only very few modifications. We have provided multiple examples of such straight-forward translations. It would

be interesting to see whether SPQR-tree-like embedding representations also exist for other planarity variants. For instance, it seems possible that the UP-tree could be restricted to an SPQR-tree-like embedding representation for windrose planarity using our characterization of windrose planarity from Chapter 11.

Part III. In the third part, we introduced and studied three new drawing styles for directed graphs. We have shown that testing for multilevel planarity, a generalization of level planarity where vertices are not constrained to exactly one level but rather to sets of levels, is NP-hard for many cases. The hardness reductions strongly rely on the fact that our definition of multilevel planarity requires each vertex to be placed at integer y-coordinates. An alternative definition of multilevel planarity could instead constrain the y-coordinate of each vertex to lie within a real interval. Would such a definition allow for efficient algorithms for more general cases?

In Chapter 10, we have studied level planar drawings with few slopes. For two slopes, this is a loose level-planar equivalent of the popular orthogonal drawing style for planar graphs. Finally, in Chapter 11 we have generalized the concept of upward planarity from one direction to two directions under the name xy-planarity. For this planarity variant, we have presented an efficient algorithm that covers the case when rightward edges are inserted into a graph with fixed upward planar embedding. It would be interesting to see whether the requirement that the upward planar embedding be fixed could be weakened to only fixing the planar embedding. This would once more require a better understanding of the interplay between planarity and upward planarity.

Bibliography

[AB19] Patrizio Angelini and Michael A. Bekos. **Hierarchical Partial Planarity**. In *Algorithmica* volume 81:6, pages 2196–2221, 2019. DOI: 10.1007/s00453-018-0530-6.
Cited on page 132.

[ABBG11] Christopher Auer, Christian Bachmaier, Franz-Josef Brandenburg, and Andreas Gleißner. **Classification of Planar Upward Embedding**. In *Graph Drawing - 19th International Symposium, GD 2011, Eindhoven, The Netherlands, September 21-23, 2011, Revised Selected Papers*. Ed. by Marc J. van Kreveld and Bettina Speckmann. Volume 7034 of Lecture Notes in Computer Science, pages 415–426. Springer, 2011. DOI: 10.1007/978-3-642-25878-7_39.
Cited on page 24.

[ABR14] Patrizio Angelini, Thomas Bläsius, and Ignaz Rutter. **Testing Mutual duality of Planar graphs**. In *Int. J. Comput. Geom. Appl.* volume 24:4, pages 325–346, 2014. DOI: 10.1142/S0218195914600103.
Cited on page 11.

[ADDF17] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. **Strip Planarity Testing for Embedded Planar Graphs**. In *Algorithmica* volume 77:4, pages 1022–1059, 2017. DOI: 10.1007/s00453-016-0128-9.
Cited on pages 20, 26.

[ADP11] Patrizio Angelini, Giuseppe Di Battista, and Maurizio Patrignani. **Finding a Minimum-depth Embedding of a Planar Graph in $O(n^4)$ Time**. In *Algorithmica* volume 60:4, pages 890–937, 2011. DOI: 10.1007/s00453-009-9380-6.
Cited on pages 132, 158.

[Ang+12] Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. **Testing the simultaneous embeddability of two graphs whose intersection is a biconnected or a connected graph**. In *J. Discrete Algorithms* volume 14, pages 150–172, 2012. DOI: 10.1016/j.jda.2011.12.015.
Cited on pages 108, 151, 153.

[Ang+15a] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. **Testing Cyclic Level and Simultaneous Level Planarity**. In *CoRR* volume abs/1510.08274, 2015. arXiv: 1510.08274. URL: http://arxiv.org/abs/1510.08274.
Cited on pages 107, 108.

[Ang+15b] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Vincenzo Roselli. **The importance of being proper: (In clustered-level planarity and T-level planarity)**. In *Theor. Comput. Sci.* volume 571, pages 1–9, 2015. DOI: 10.1016/j.tcs.2014.12.019.
Cited on pages 107, 178.

[Ang+15c] Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. **Testing Planarity of Partially Embedded Graphs**. In *ACM Trans. Algorithms* volume 11:4, pages 32:1–32:42, 2015. DOI: 10.1145/2629341.
Cited on pages 1, 107, 132, 151, 152, 158, 172, 177.

[Ang+16] Patrizio Angelini, Steven Chaplick, Sabine Cornelsen, Giordano Da Lozzo, Giuseppe Di Battista, Peter Eades, Philipp Kindermann, Jan Kratochvíl, Fabian Lipp, and Ignaz Rutter. **Simultaneous Orthogonal Planarity**. In *Graph Drawing and Network Visualization - 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers*. Ed. by Yifan Hu and Martin Nöllenburg. Volume 9801 of Lecture Notes in Computer Science, pages 532–545. Springer, 2016. DOI: 10.1007/978-3-319-50106-2_41.
Cited on pages 132, 203.

[Ang+18] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Valentino Di Donato, Philipp Kindermann, Günter Rote, and Ignaz Rutter. **Windrose Planarity: Embedding Graphs with Direction-Constrained Edges**. In *ACM Trans. Algorithms* volume 14:4, pages 54:1–54:24, 2018. DOI: 10.1145/3239561.
Cited on pages 222, 230, 235.

[Ang+20] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. **Beyond level planarity: Cyclic, torus, and simultaneous level planarity**. In *Theor. Comput. Sci.* volume 804, pages 161–170, 2020. DOI: 10.1016/j.tcs.2019.11.024.
Cited on pages 59, 178, 204.

[Ark+04] Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Martin L. Demaine, Joseph S. B. Mitchell, Saurabh Sethia, and Steven Skiena. **When can you fold a map?** In *Comput. Geom.* volume 29:1, pages 23–46, 2004. DOI: 10.1016/j.comgeo.2004.03.012.
Cited on page 26.

[Bac04] Christian Bachmaier. **Circle Planarity of Level Graphs**. PhD thesis. Universität Passau, 2004.
Cited on page 62.

[Ban+19] Michael J. Bannister, William E. Devanny, Vida Dujmovic, David Eppstein, and David R. Wood. **Track Layouts, Layered Path Decompositions, and Leveled Planarity**. In *Algorithmica* volume 81:4, pages 1561–1583, 2019. DOI: 10.1007/s00453-018-0487-5.
Cited on page 178.

[BBF05] Christian Bachmaier, Franz-Josef Brandenburg, and Michael Forster. **Radial Level Planarity Testing and Embedding in Linear Time**. In *J. Graph Algorithms Appl.* volume 9:1, pages 53–97, 2005. DOI: 10.7155/jgaa.00100.
Cited on pages v, 2, 4, 21, 34, 62, 63, 107, 178, 238.

[BBJR19] Lukas Barth, Guido Brückner, Paul Jungeblut, and Marcel Radermacher. **Multilevel Planarity**. In *WALCOM: Algorithms and Computation - 13th International Conference, WALCOM 2019, Guwahati, India, February 27 - March 2, 2019, Proceedings*. Ed. by Gautam K. Das, Partha Sarathi Mandal, Krishnendu Mukhopadhyaya, and Shin-Ichi Nakano. Volume 11355 of Lecture Notes in Computer Science, pages 219–231. Springer, 2019. DOI: 10.1007/978-3-030-10564-8_18.
Cited on page 177.

[BBJR21] Lukas Barth, Guido Brückner, Paul Jungeblut, and Marcel Radermacher. **Multilevel Planarity**. In *J. Graph Algorithms Appl.* volume 25:1, pages 151–170, 2021. DOI: 10.7155/jgaa.00554. URL: https://doi.org/10.7155/jgaa.00554.
Cited on page 177.

[BC20] Guido Brückner and Vera Chekan. **Drawing Two Posets**. In *CoRR* volume abs/2010.12928, 2020. arXiv: 2010.12928.
Cited on pages 21, 221.

[BDLM94] Paola Bertolazzi, Giuseppe Di Battista, Giuseppe Liotta, and Carlo Mannino. **Upward Drawings of Triconnected Digraphs**. In *Algorithmica* volume 12:6, pages 476–497, 1994. DOI: 10.1007/BF01188716.
Cited on pages 10, 21, 24, 30, 31, 157, 178, 185, 199, 222, 224.

[BDMT98] Paola Bertolazzi, Giuseppe Di Battista, Carlo Mannino, and Roberto Tamassia. **Optimal Upward Planarity Testing of Single-Source Digraphs**. In *SIAM J. Comput.* volume 27:1, pages 132–169, 1998. DOI: 10.1137/S0097539794279626.
Cited on pages 157, 162, 166, 167, 169, 171, 178, 182, 183, 185, 186, 188, 199, 222.

[Bek+18] Michael A. Bekos, Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. **Universal Slope Sets for Upward Planar Drawings**. In *Graph Drawing and Network Visualization - 26th International Symposium, GD 2018, Barcelona, Spain, September 26-28, 2018, Proceedings*. Ed. by Therese C. Biedl and Andreas Kerren. Volume 11282 of Lecture Notes in Computer Science, pages 77–91. Springer, 2018. DOI: 10.1007/978-3-030-04414-5_6.
Cited on page 203.

[BHR19] Guido Brückner, Markus Himmel, and Ignaz Rutter. **An SPQR-Tree-Like Embedding Representation for Upward Planarity**. In *Graph Drawing and Network Visualization - 27th International Symposium, GD 2019, Prague, Czech Republic, September 17-20, 2019, Proceedings*. Ed. by Daniel Archambault and Csaba D. Tóth. Volume 11904 of Lecture Notes in Computer Science, pages 517–531. Springer, 2019. DOI: 10.1007/978-3-030-35802-0_39.
Cited on pages 108, 132, 157.

[BKM19] Guido Brückner, Nadine Davina Krisam, and Tamara Mchedlidze. **Level-Planar Drawings with Few Slopes**. In *Graph Drawing and Network Visualization - 27th International Symposium, GD 2019, Prague, Czech Republic, September 17-20, 2019, Proceedings*. Ed. by Daniel Archambault and Csaba D. Tóth. Volume 11904 of Lecture Notes in Computer Science, pages 559–572. Springer, 2019. DOI: 10.1007/978-3-030-35802-0_42.
Cited on page 201.

[BKR13] Thomas Bläsius, Stephen G. Kobourov, and Ignaz Rutter. **Simultaneous Embedding of Planar Graphs**. In *Handbook on Graph Drawing and Visualization*. Ed. by Roberto Tamassia. Chapman and Hall/CRC, 2013, pages 349–381.
Cited on pages 1, 108, 132, 158, 203.

[BKR18] Thomas Bläsius, Annette Karrer, and Ignaz Rutter. **Simultaneous Embedding: Edge Orderings, Relative Positions, Cutvertices**. In *Algorithmica* volume 80:4, pages 1214–1277, 2018. DOI: 10.1007/s00453-017-0301-9.
Cited on pages 108, 153, 170.

[BL76] Kellogg S. Booth and George S. Lueker. **Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms**. In *J. Comput. Syst. Sci.* volume 13:3, pages 335–379, 1976. DOI: 10.1016/S0022-0000(76)80045-1.
Cited on pages 15, 113.

[BLR16] Thomas Bläsius, Sebastian Lehmann, and Ignaz Rutter. **Orthogonal graph drawing with inflexible edges**. In *Comput. Geom.* volume 55, pages 26–40, 2016. DOI: 10.1016/j.comgeo.2016.03.001.
Cited on page 158.

[BM04] John M. Boyer and Wendy J. Myrvold. **On the Cutting Edge: Simplified O(n) Planarity by Edge Addition**. In *J. Graph Algorithms Appl.* volume 8:3, pages 241–273, 2004. DOI: 10.7155/jgaa.00091.
Cited on page 1.

[BM90] Daniel Bienstock and Clyde L. Monma. **On the Complexity of Embedding Planar Graphs To Minimize Certain Distance Measures**. In *Algorithmica* volume 5:1, pages 93–109, 1990. DOI: 10.1007/BF01840379.
Cited on page 132.

[BMW06] János Barát, Jirí Matousek, and David R. Wood. **Bounded-Degree Graphs have Arbitrarily Large Geometric Thickness**. In *Electron. J. Comb.* volume 13:1, 2006. DOI: 10.37236/1029.
Cited on page 203.

[Bor+17] Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. **Multiple-Source Multiple-Sink Maximum Flow in Directed Planar Graphs in Near-Linear Time**. In *SIAM J. Comput.* volume 46:4, pages 1280–1303, 2017. DOI: 10.1137/15M1042929.
Cited on pages 25, 207, 208.

[BR15] Thomas Bläsius and Ignaz Rutter. **Disconnectivity and relative positions in simultaneous embeddings**. In *Comput. Geom.* volume 48:6, pages 459–478, 2015. DOI: 10.1016/j.comgeo.2015.02.002.
Cited on page 108.

[BR16] Thomas Bläsius and Ignaz Rutter. **Simultaneous PQ-Ordering with Applications to Constrained Embedding Problems**. In *ACM Trans. Algorithms* volume 12:2, pages 16:1–16:46, 2016. DOI: 10.1145/2738054.
Cited on page 108.

[BR17] Guido Brückner and Ignaz Rutter. **Partial and Constrained Level Planarity**. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. Ed. by Philip N. Klein, pages 2000–2011. SIAM, 2017. DOI: 10.1137/1.9781611974782.130.
Cited on pages 34, 105, 132, 151, 152, 153, 158, 178, 203.

[BR20] Guido Brückner and Ignaz Rutter. **An SPQR-Tree-Like Embedding Representation for Level Planarity**. In *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*. Ed. by Yixin Cao, Siu-Wing Cheng, and Minming Li. Volume 181 of LIPIcs, pages 8:1–8:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. DOI: 10.4230/LIPIcs.ISAAC.2020.8.
Cited on page 131.

[BR21] Guido Brückner and Ignaz Rutter. **Radial Level Planarity with Fixed Embedding**. In *J. Graph Algorithms Appl.* volume 25:1, pages 353–366, 2021. DOI: 10.7155/jgaa.00561.
Cited on page 19.

[Bra14] Franz-Josef Brandenburg. **Upward planar drawings on the standing and the rolling cylinders**. In *Comput. Geom.* volume 47:1, pages 25–41, 2014. DOI: 10.1016/j.comgeo.2013.08.003.
Cited on page 107.

[BRS18] Guido Brückner, Ignaz Rutter, and Peter Stumpf. **Level Planarity: Transitivity vs. Even Crossings**. In *Graph Drawing and Network Visualization - 26th International Symposium, GD 2018, Barcelona, Spain, September 26-28, 2018, Proceedings*. Ed. by Therese C. Biedl and Andreas Kerren. Volume 11282 of Lecture Notes in Computer Science, pages 39–52. Springer, 2018. DOI: 10.1007/978-3-030-04414-5_3.
Cited on pages 33, 63, 107, 157.

[Brü16] Guido Brückner. **Extending Partial Planar Drawings of Level Graphs**. MA thesis. Karlsruhe Institute of Technology, 2016.
Cited on page 105.

[BRW16] Thomas Bläsius, Ignaz Rutter, and Dorothea Wagner. **Optimal Orthogonal Graph Drawing with Convex Bend Costs**. In *ACM Trans. Algorithms* volume 12:3, pages 33:1–33:32, 2016. DOI: 10.1145/2838736.
Cited on pages 1, 131, 158.

[CD95] M. Chandramouli and A. A. Diwan. **Upward Numbering Testing for Triconnected Graphs**. In *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20-22, 1995, Proceedings*. Ed. by Franz-Josef Brandenburg. Volume 1027 of Lecture Notes in Computer Science, pages 140–151. Springer, 1995. DOI: 10.1007/BFb0021798.
Cited on page 20.

[CFK19] Steven Chaplick, Radoslav Fulek, and Pavel Klavík. **Extending partial representations of circle graphs**. In *J. Graph Theory* volume 91:4, pages 365–394, 2019. DOI: 10.1002/jgt.22436.
Cited on page 108.

[CGMW09] Markus Chimani, Carsten Gutwenger, Petra Mutzel, and Christian Wolf. **Inserting a vertex into a planar graph**. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 375–383. SIAM, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496812.
Cited on page 132.

[CH16] Markus Chimani and Petr Hlinený. **Inserting Multiple Edges into a Planar Graph**. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*. Ed. by Sándor P. Fekete and Anna Lubiw. Volume 51 of LIPIcs, pages 30:1–30:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. DOI: 10.4230/LIPIcs.SoCG.2016.30.
Cited on page 132.

[Che19] Vera Chekan. **Upward-Rightward-Prescribed Planarity**. Karlsruhe Institute of Technology, 2019.
Cited on page 221.

[Chi+13] Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. **The Open Graph Drawing Framework (OGDF)**. In *Handbook on Graph Drawing and Visualization*. Ed. by Roberto Tamassia. Chapman and Hall/CRC, 2013, pages 543–569.
Cited on page 61.

[Cho34] Chaim Chojnacki. **Über wesentlich unplättbare Kurven im dreidimensionalen Raume**. In *Fundamenta Mathematicae* volume 23:1, pages 135–142, 1934.
Cited on pages 1, 34.

[CNAO85] Norishige Chiba, Takao Nishizeki, Shigenobu Abe, and Takao Ozawa. **A Linear Algorithm for Embedding Planar Graphs Using PQ-Trees**. In *J. Comput. Syst. Sci.* volume 30:1, pages 54–76, 1985. DOI: 10.1016/0022-0000(85)90004-2.
Cited on pages 101, 135.

[DD11] Will Durant and Ariel Durant. **The Age of Louis XIV: The Story of Civilization**. Simon and Schuster, 2011.
Cited on page 178.

[DDF20] Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. **Extending upward planar graph drawings**. In *Comput. Geom.* volume 91, page 101668, 2020. DOI: 10.1016/j.comgeo.2020.101668.
Cited on pages 108, 158.

[DESW07] Vida Dujmovic, David Eppstein, Matthew Suderman, and David R. Wood. **Drawings of planar graphs with few slopes and segments**. In *Comput. Geom.* volume 38:3, pages 194–212, 2007. DOI: 10.1016/j.comgeo.2006.09.002.
Cited on pages 1, 203.

[DETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. **Graph Drawing: Algorithms for the Visualization of Graphs**. Prentice-Hall, 1999. ISBN: 0-13-301615-3.
Cited on page 199.

[DF09] Giuseppe Di Battista and Fabrizio Frati. **Efficient C-Planarity Testing for Embedded Flat Clustered Graphs with Small Faces**. In *J. Graph Algorithms Appl.* volume 13:3, pages 349–378, 2009. DOI: 10.7155/jgaa.00191.
Cited on page 177.

[DJKR14] Giordano Da Lozzo, Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter. **Planar Embeddings with Small and Uniform Faces**. In *Algorithms and Computation - 25th International Symposium, ISAAC 2014, Jeonju, Korea, December 15-17, 2014, Proceedings*. Ed. by Hee-Kap Ahn and Chan-Su Shin. Volume 8889 of Lecture Notes in Computer Science, pages 633–645. Springer, 2014. DOI: 10.1007/978-3-319-13075-0_50.
Cited on page 158.

[dK12] Mark de Berg and Amirali Khosravi. **Optimal Binary Space Partitions for Segments in the Plane**. In *Int. J. Comput. Geom. Appl.* volume 22:3, pages 187–206, 2012. DOI: 10.1142/S0218195912500045.
Cited on pages 121, 126, 195, 215.

[DLP18] Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. **Bend-Minimum Orthogonal Drawings in Quadratic Time**. In *Graph Drawing and Network Visualization - 26th International Symposium, GD 2018, Barcelona, Spain, September 26-28, 2018, Proceedings*. Ed. by Therese C. Biedl and Andreas Kerren. Volume 11282 of Lecture Notes in Computer Science, pages 481–494. Springer, 2018. DOI: 10.1007/978-3-030-04414-5_34.
Cited on page 158.

[DLP19] Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. **HV-planarity: Algorithms and complexity**. In *J. Comput. Syst. Sci.* volume 99, pages 72–90, 2019. DOI: 10.1016/j.jcss.2018.08.003.
Cited on page 222.

[dM12] Hubert de Fraysseix and Patrice Ossona de Mendez. **Trémaux trees and planarity**. In *Eur. J. Comb.* volume 33:3, pages 279–293, 2012. DOI: 10.1016/j.ejc.2011.09.012.
Cited on page 1.

[DN88] Giuseppe Di Battista and Enrico Nardelli. **Hierarchies and planarity theory**. In *IEEE Trans. Syst. Man Cybern.* volume 18:6, pages 1035–1046, 1988. DOI: 10.1109/21.23105.
Cited on pages 2, 62, 101, 107, 108, 109, 110, 113, 142, 150.

[DR18] Giordano Da Lozzo and Ignaz Rutter. **Approximation Algorithms for Facial Cycles in Planar Embeddings**. In *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*. Ed. by Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao. Volume 123 of LIPIcs, pages 41:1–41:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. DOI: 10.4230/LIPIcs.ISAAC.2018.41.
Cited on page 158.

[DSW07] Vida Dujmovic, Matthew Suderman, and David R. Wood. **Graph drawings with few slopes**. In *Comput. Geom.* volume 38:3, pages 181–193, 2007. DOI: 10.1016/j.comgeo.2006.08.002.
Cited on page 203.

[DT88] Giuseppe Di Battista and Roberto Tamassia. **Algorithms for Plane Representations of Acyclic Digraphs**. In *Theor. Comput. Sci.* volume 61, pages 175–198, 1988. DOI: 10.1016/0304-3975(88)90123-5.
Cited on pages 135, 182, 187, 225.

[DT89] Giuseppe Di Battista and Roberto Tamassia. **Incremental Planarity Testing (Extended Abstract)**. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 436–441. IEEE Computer Society, 1989. DOI: 10.1109/SFCS.1989.63515.
Cited on pages 1, 132, 158.

[DT90] Giuseppe Di Battista and Roberto Tamassia. **On-Line Graph Algorithms with SPQR-Trees**. In *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*. Ed. by Mike Paterson. Volume 443 of Lecture Notes in Computer Science, pages 598–611. Springer, 1990. DOI: 10.1007/BFb0032061.
Cited on pages 1, 132, 158.

[DT96] Giuseppe Di Battista and Roberto Tamassia. **On-Line Maintenance of Triconnected Components with SPQR-Trees**. In *Algorithmica* volume 15:4, pages 302–318, 1996. DOI: 10.1007/BF01961541.
Cited on pages 1, 132, 158.

[EFK09] Alejandro Estrella-Balderrama, J. Joseph Fowler, and Stephen G. Kobourov. **On the Characterization of Level Planar Trees by Minimal Patterns**. In *Graph Drawing, 17th International Symposium, GD 2009, Chicago, IL, USA, September 22-25, 2009. Revised Papers*. Ed. by David Eppstein and Emden R. Gansner. Volume 5849 of Lecture Notes in Computer Science, pages 69–80. Springer, 2009. DOI: 10.1007/978-3-642-11805-0_9.
Cited on page 2.

[EIS76] Shimon Even, Alon Itai, and Adi Shamir. **On the Complexity of Timetable and Multicommodity Flow Problems**. In *SIAM J. Comput.* volume 5:4, pages 691–703, 1976. DOI: 10.1137/0205048.
Cited on page 215.

[ELM16] William S. Evans, Giuseppe Liotta, and Fabrizio Montecchiani. **Simultaneous visibility representations of plane st-graphs using L-shapes**. In *Theor. Comput. Sci.* volume 645, pages 100–111, 2016. DOI: 10.1016/j.tcs.2016.06.045.
Cited on page 222.

[Est+07] Alejandro Estrella-Balderrama, Elisabeth Gassner, Michael Jünger, Merijam Percan, Marcus Schaefer, and Michael Schulz. **Simultaneous Geometric Graph Embeddings**. In *Graph Drawing, 15th International Symposium, GD 2007, Sydney, Australia, September 24-26, 2007. Revised Papers*. Ed. by Seok-Hee Hong, Takao Nishizeki, and Wu Quan. Volume 4875 of Lecture Notes in Computer Science, pages 280–290. Springer, 2007. DOI: 10.1007/978-3-540-77537-9_28.
Cited on page 108.

[FB04] Michael Forster and Christian Bachmaier. **Clustered Level Planarity**. In *SOFSEM 2004: Theory and Practice of Computer Science, 30th Conference on Current Trends in Theory and Practice of Computer Science, Merin, Czech Republic, January 24-30, 2004*. Ed. by Peter van Emde Boas, Jaroslav Pokorný, Mária Bieliková, and Julius Stuller. Volume 2932 of Lecture Notes in Computer Science, pages 218–228. Springer, 2004. DOI: 10.1007/978-3-540-24618-3_18.
Cited on page 178.

[FCE95] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. **Planarity for Clustered Graphs**. In *Algorithms - ESA '95, Third Annual European Symposium, Corfu, Greece, September 25-27, 1995, Proceedings*. Ed. by Paul G. Spirakis. Volume 979 of Lecture Notes in Computer Science, pages 213–226. Springer, 1995. DOI: 10.1007/3-540-60313-1_145.
Cited on page 158.

[FK07] J. Joseph Fowler and Stephen G. Kobourov. **Minimum Level Nonplanar Patterns for Trees**. In *Graph Drawing, 15th International Symposium, GD 2007, Sydney, Australia, September 24-26, 2007. Revised Papers*. Ed. by Seok-Hee Hong, Takao Nishizeki, and Wu Quan. Volume 4875 of Lecture Notes in Computer Science, pages 69–75. Springer, 2007. DOI: 10.1007/978-3-540-77537-9_10.
Cited on page 2.

[FPS16] Radoslav Fulek, Michael J. Pelsmajer, and Marcus Schaefer. **Hanani-Tutte for Radial Planarity II**. In *Graph Drawing and Network Visualization - 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers*. Ed. by Yifan Hu and Martin Nöllenburg. Volume 9801 of Lecture Notes in Computer Science, pages 468–481. Springer, 2016. DOI: 10.1007/978-3-319-50106-2_36.
Cited on pages iv, 4, 33, 35, 59, 63.

[FPS17] Radoslav Fulek, Michael J. Pelsmajer, and Marcus Schaefer. **Hanani-Tutte for Radial Planarity**. In *J. Graph Algorithms Appl.* volume 21:1, pages 135–154, 2017. DOI: 10.7155/jgaa.00408.
Cited on pages iv, 2, 4, 33, 35, 63, 107.

[FPSS11] Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Stefankovic. **Hanani-Tutte and Monotone Drawings**. In *Graph-Theoretic Concepts in Computer Science - 37th International Workshop, WG 2011, Teplá Monastery, Czech Republic, June 21-24, 2011. Revised Papers*. Ed. by Petr Kolman and Jan Kratochvíl. Volume 6986 of Lecture Notes in Computer Science, pages 283–294. Springer, 2011. DOI: 10.1007/978-3-642-25870-1_26.
Cited on page 132.

[FPSŠ13] Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. **Hanani–Tutte, Monotone Drawings, and Level-Planarity**. In *Thirty Essays on Geometric Graph Theory*. Ed. by János Pach, pages 263–287. New York, NY: Springer New York, 2013. ISBN: 978-1-4614-0110-0. DOI: 10.1007/978-1-4614-0110-0_14.
Cited on pages iv, 2, 4, 33, 35, 36, 63, 107, 157, 237.

[FR06] Jittat Fakcharoenphol and Satish Rao. **Planar graphs, negative weight edges, shortest paths, and near linear time**. In *J. Comput. Syst. Sci.* volume 72:5, pages 868–889, 2006. DOI: 10.1016/j.jcss.2005.05.007.
Cited on page 213.

[FT20] Radoslav Fulek and Csaba D. Tóth. **Atomic Embeddability, Clustered Planarity, and Thickenability**. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*. Ed. by Shuchi Chawla, pages 2876–2895. SIAM, 2020. DOI: 10.1137/1.9781611975994.175.
Cited on page 108.

[Gab83] Harold N. Gabow. **An Efficient Reduction Technique for Degree-Constrained Subgraph and Bidirected Network Flow Problems**. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*. Ed. by David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, pages 448–456. ACM, 1983. DOI: 10.1145/800061.808776.
Cited on pages 21, 22.

[GHKR14] Luca Grilli, Seok-Hee Hong, Jan Kratochvíl, and Ignaz Rutter. **Drawing Simultaneously Embedded Graphs with Few Bends**. In *Graph Drawing - 22nd International Symposium, GD 2014, Würzburg, Germany, September 24-26, 2014, Revised Selected Papers*. Ed. by Christian A. Duncan and Antonios Symvonis. Volume 8871 of Lecture Notes in Computer Science, pages 40–51. Springer, 2014. DOI: 10.1007/978-3-662-45803-7_4.
Cited on page 108.

[Gia+14] Emilio Di Giacomo, Walter Didimo, Michael Kaufmann, Giuseppe Liotta, and Fabrizio Montecchiani. **Upward-rightward planar drawings**. In *5th International Conference on Information, Intelligence, Systems and Applications, IISA 2014, Chania, Crete, Greece, July 7-9, 2014*. Ed. by Nikolaos G. Bourbakis, George A. Tsihrintzis, and Maria Virvou, pages 145–150. IEEE, 2014. DOI: 10.1109/IISA.2014.6878792.
Cited on page 222.

[GJ75] M. R. Garey and David S. Johnson. **Complexity Results for Multiprocessor Scheduling under Resource Constraints**. In *SIAM J. Comput.* volume 4:4, pages 397–411, 1975. DOI: 10.1137/0204035.
Cited on page 122.

[GJ77] M. R. Garey and David S. Johnson. **Two-Processor Scheduling with Start-Times and Deadlines**. In *SIAM J. Comput.* volume 6:3, pages 416–426, 1977. DOI: 10.1137/0206029.
Cited on page 191.

[GLM15] Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. **Drawing Outer 1-planar Graphs with Few Slopes**. In *J. Graph Algorithms Appl.* volume 19:2, pages 707–741, 2015. DOI: 10.7155/jgaa.00376.
Cited on page 203.

[GLM18] Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. **Drawing subcubic planar graphs with four slopes and optimal angular resolution**. In *Theor. Comput. Sci.* volume 714, pages 51–73, 2018. DOI: 10.1016/j.tcs.2017.12.004.
Cited on page 203.

[GLM20] Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. **1-bend upward planar slope number of SP-digraphs**. In *Comput. Geom.* volume 90, page 101628, 2020. DOI: 10.1016/j.comgeo.2020.101628.
Cited on page 203.

[GM00] Carsten Gutwenger and Petra Mutzel. **A Linear Time Implementation of SPQR-Trees**. In *Graph Drawing, 8th International Symposium, GD 2000, Colonial Williamsburg, VA, USA, September 20-23, 2000, Proceedings*. Ed. by Joe Marks. Volume 1984 of Lecture Notes in Computer Science, pages 77–90. Springer, 2000. DOI: 10.1007/3-540-44541-2_8.
Cited on pages 150, 170.

[GMW05] Carsten Gutwenger, Petra Mutzel, and René Weiskircher. **Inserting an Edge into a Planar Graph**. In *Algorithmica* volume 41:4, pages 289–308, 2005. DOI: 10.1007/s00453-004-1128-8.
Cited on pages 1, 131, 132.

[GT01] Ashim Garg and Roberto Tamassia. **On the Computational Complexity of Upward and Rectilinear Planarity Testing**. In *SIAM Journal on Computing* volume 31:2, pages 601–625, 2001. DOI: 10.1137/S0097539794277123.
Cited on pages 21, 34, 178, 181, 191, 222.

[GT85] Harold N. Gabow and Robert Endre Tarjan. **A Linear-Time Algorithm for a Special Case of Disjoint Set Union**. In *J. Comput. Syst. Sci.* volume 30:2, pages 209–221, 1985. DOI: 10.1016/0022-0000(85)90014-5.
Cited on pages 69, 101, 153, 171.

[Has01] S. Mehdi Hashemi. **Digraph embedding**. In *Discret. Math.* volume 233:1-3, pages 321–328, 2001. DOI: 10.1016/S0012-365X(00)00249-1.
Cited on page 24.

[Has81] Refael Hassin. **Maximum Flow in (s,t) Planar Networks**. In *Inf. Process. Lett.* volume 13:3, page 107, 1981. DOI: 10.1016/0020-0190(81)90120-4.
Cited on page 208.

[HH07] Martin Harrigan and Patrick Healy. **Practical Level Planarity Testing and Layout with Embedding Constraints**. In *Graph Drawing, 15th International Symposium, GD 2007, Sydney, Australia, September 24-26, 2007. Revised Papers*. Ed. by Seok-Hee Hong, Takao Nishizeki, and Wu Quan. Volume 4875 of Lecture Notes in Computer Science, pages 62–68. Springer, 2007. DOI: 10.1007/978-3-540-77537-9_9.
Cited on pages 2, 34, 107, 178.

[HJL13] Bernhard Haeupler, Krishnam Raju Jampani, and Anna Lubiw. **Testing Simultaneous Planarity when the Common Graph is 2-Connected**. In *J. Graph Algorithms Appl.* volume 17:3, pages 147–171, 2013. DOI: 10.7155/jgaa.00289.
Cited on page 155.

[HK04] Patrick Healy and Ago Kuusik. **Algorithms for multi-level graph planarity testing and layout**. In *Theor. Comput. Sci.* volume 320:2-3, pages 331–344, 2004. DOI: 10.1016/j.tcs.2004.02.033.
Cited on page 2.

[HKL04] Patrick Healy, Ago Kuusik, and Sebastian Leipert. **A characterization of level planar graphs**. In *Discret. Math.* volume 280:1-3, pages 51–63, 2004. DOI: 10.1016/j.disc.2003.02.001.
Cited on pages 2, 190.

[HKRS97] Monika Rauch Henzinger, Philip N. Klein, Satish Rao, and Sairam Subramanian. **Faster Shortest-Path Algorithms for Planar Graphs**. In *J. Comput. Syst. Sci.* volume 55:1, pages 3–23, 1997. DOI: 10.1006/jcss.1997.1493.
Cited on page 213.

[HL96] Michael D. Hutton and Anna Lubiw. **Upward Planar Drawing of Single-Source Acyclic Digraphs**. In *SIAM J. Comput.* volume 25:2, pages 291–311, 1996. DOI: 10.1137/S0097539792235906.
Cited on pages 132, 157, 159, 160, 161, 178.

[HM03] Wen-Lian Hsu and Ross M. McConnell. **PC trees and circular-ones arrangements**. In *Theor. Comput. Sci.* volume 296:1, pages 99–116, 2003. DOI: 10.1016/S0304-3975(02)00435-8.
Cited on pages iv, 4, 14, 15, 99.

[HME06] Seok-Hee Hong, Brendan D. McKay, and Peter Eades. **A Linear Time Algorithm for Constructing Maximally Symmetric Straight Line Drawings of Triconnected Planar Graphs**. In *Discret. Comput. Geom.* volume 36:2, pages 283–311, 2006. DOI: 10.1007/s00454-006-1231-5.
Cited on pages 1, 131.

[HN13] Patrick Healy and Nikola S. Nikolov. **Hierarchical Drawing Algorithms**. In *Handbook on Graph Drawing and Visualization*. Ed. by Roberto Tamassia. Chapman and Hall/CRC, 2013, pages 409–453.
Cited on pages 106, 202.

[Hof17] Udo Hoffmann. **On the Complexity of the Planar Slope Number Problem**. In *J. Graph Algorithms Appl.* volume 21:2, pages 183–193, 2017. DOI: 10.7155/jgaa.00411.
Cited on page 203.

[HP95] Lenwood S. Heath and Sriram V. Pemmaraju. **Recognizing Leveled-Planar Dags in Linear Time**. In *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20-22, 1995, Proceedings*. Ed. by Franz-Josef Brandenburg. Volume 1027 of Lecture Notes in Computer Science, pages 300–311. Springer, 1995. DOI: 10.1007/BFb0021813.
Cited on pages 2, 62.

[HP99] Lenwood S. Heath and Sriram V. Pemmaraju. **Stack and Queue Layouts of Directed Acyclic Graphs: Part II**. In *SIAM J. Comput.* volume 28:5, pages 1588–1626, 1999. DOI: 10.1137/S0097539795291550.
Cited on page 2.

[HR92] Lenwood S. Heath and Arnold L. Rosenberg. **Laying out Graphs Using Queues**. In *SIAM J. Comput.* volume 21:5, pages 927–958, 1992. DOI: 10.1137/0221055.
Cited on page 178.

[HRK97] S. Mehdi Hashemi, Ivan Rival, and Andrzej Kisielewicz. **The Complexity of Upward Drawings on Spheres**. In *Order* volume 14, pages 327–363, 1997. DOI: 10.1023/A:1006095702164.
Cited on page 21.

[HT08] Bernhard Haeupler and Robert Endre Tarjan. **Planarity Algorithms via PQ-Trees (Extended Abstract)**. In *Electron. Notes Discret. Math.* volume 31, pages 143–149, 2008. DOI: 10.1016/j.endm.2008.06.029. URL: https://doi.org/10.1016/j.endm.2008.06.029.
Cited on page 1.

[HT73] John E. Hopcroft and Robert Endre Tarjan. **Dividing a Graph into Triconnected Components**. In *SIAM J. Comput.* volume 2:3, pages 135–158, 1973. DOI: 10.1137/0202012.
Cited on pages 1, 132, 150, 158, 170.

[HT74] John E. Hopcroft and Robert Endre Tarjan. **Efficient Planarity Testing**. In *J. ACM* volume 21:4, pages 549–568, 1974. DOI: 10.1145/321850.321852.
Cited on pages 1, 21, 61.

[HT84] Dov Harel and Robert Endre Tarjan. **Fast Algorithms for Finding Nearest Common Ancestors**. In *SIAM J. Comput.* volume 13:2, pages 338–355, 1984. DOI: 10.1137/0213024.
Cited on page 153.

[Hu69] Te Chiang Hu. **Integer Programming and Network Flows**. Reading, MA: Addison-Wesley, 1969, Reading, MA.
Cited on page 208.

[IS79] Alon Itai and Yossi Shiloach. **Maximum Flow in Planar Networks**. In *SIAM J. Comput.* volume 8:2, pages 135–150, 1979. DOI: 10.1137/0208012.
Cited on page 208.

[Jän+15] Stefan Jänicke, Annette Geßner, Greta Franzini, Melissa Terras, Simon Mahony, and Gerik Scheuermann. **TRAViz: A Visualization for Variant Graphs**. In *Digit. Scholarsh. Humanit.* volume 30:Suppl-1, pages i83–i99, 2015. DOI: 10.1093/llc/fqv049.
Cited on page 201.

[Jel+09] Eva Jelínková, Jan Kára, Jan Kratochvíl, Martin Pergel, Ondrej Suchý, and Tomás Vyskocil. **Clustered Planarity: Small Clusters in Cycles and Eulerian Graphs**. In *J. Graph Algorithms Appl.* volume 13:3, pages 379–422, 2009. DOI: 10.7155/jgaa.00192.
Cited on page 177.

[JKR13] Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter. **A Kuratowski-type theorem for planarity of partially embedded graphs**. In *Comput. Geom.* volume 46:4, pages 466–492, 2013. DOI: 10.1016/j.comgeo.2012.07.005.
Cited on pages 107, 158, 177.

[JL02] Michael Jünger and Sebastian Leipert. **Level Planar Embedding in Linear Time**. In *J. Graph Algorithms Appl.* volume 6:1, pages 67–113, 2002. DOI: 10.7155/jgaa.00045.
Cited on pages 2, 62, 63, 107, 132, 157, 178, 199.

[JL10] Krishnam Raju Jampani and Anna Lubiw. **Simultaneous Interval Graphs**. In *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*. Ed. by Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park. Volume 6506 of Lecture Notes in Computer Science, pages 206–217. Springer, 2010. DOI: 10.1007/978-3-642-17517-6_20.
Cited on page 108.

[JL12] Krishnam Raju Jampani and Anna Lubiw. **The Simultaneous Representation Problem for Chordal, Comparability and Permutation Graphs**. In *J. Graph Algorithms Appl.* volume 16:2, pages 283–315, 2012. DOI: 10.7155/jgaa.00259.
Cited on page 108.

[JLM97] Michael Jünger, Sebastian Leipert, and Petra Mutzel. **Pitfalls of Using PQ-Trees in Automatic Graph Drawing**. In *Graph Drawing, 5th International Symposium, GD '97, Rome, Italy, September 18-20, 1997, Proceedings*. Ed. by Giuseppe Di Battista. Volume 1353 of Lecture Notes in Computer Science, pages 193–204. Springer, 1997. DOI: 10.1007/3-540-63938-1_62.
Cited on pages 2, 62.

[JLM98] Michael Jünger, Sebastian Leipert, and Petra Mutzel. **Level Planarity Testing in Linear Time**. In *Graph Drawing, 6th International Symposium, GD'98, Montréal, Canada, August 1998, Proceedings*. Ed. by Sue Whitesides. Volume 1547 of Lecture Notes in Computer Science, pages 224–237. Springer, 1998. DOI: 10.1007/3-540-37623-2_17.
Cited on pages 2, 21, 34, 62, 107, 120, 132.

[Jun17] Paul Jungeblut. **On Interval Planar Graphs**. Karlsruhe Institute of Technology, 2017.
Cited on page 177.

[Kam01] Henry Kamen. **Philip V of Spain: The King Who Reigned Twice**. Yale University Press, 2001.
Cited on page 178.

[KKKW12] Pavel Klavík, Jan Kratochvíl, Tomasz Krawczyk, and Bartosz Walczak. **Extending Partial Representations of Function Graphs and Permutation Graphs**. In *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*. Ed. by Leah Epstein and Paolo Ferragina. Volume 7501 of Lecture Notes in Computer Science, pages 671–682. Springer, 2012. DOI: 10.1007/978-3-642-33090-2_58.
Cited on page 108.

[KKOS15] Pavel Klavík, Jan Kratochvíl, Yota Otachi, and Toshiki Saitoh. **Extending partial representations of subclasses of chordal graphs**. In *Theor. Comput. Sci.* volume 576, pages 85–101, 2015. DOI: 10.1016/j.tcs.2015.02.007.
Cited on page 108.

[Kla+17a] Pavel Klavík, Jan Kratochvíl, Yota Otachi, Ignaz Rutter, Toshiki Saitoh, Maria Saumell, and Tomás Vyskocil. **Extending Partial Representations of Proper and Unit Interval Graphs**. In *Algorithmica* volume 77:4, pages 1071–1104, 2017. DOI: 10.1007/s00453-016-0133-z.
Cited on page 108.

[Kla+17b] Pavel Klavík, Jan Kratochvíl, Yota Otachi, Toshiki Saitoh, and Tomás Vyskocil. **Extending Partial Representations of Interval Graphs**. In *Algorithmica* volume 78:3, pages 945–967, 2017. DOI: 10.1007/s00453-016-0186-z.
Cited on pages 108, 114.

[KMS18] Philipp Kindermann, Wouter Meulemans, and André Schulz. **Experimental Analysis of the Accessibility of Drawings with Few Segments**. In *J. Graph Algorithms Appl.* volume 22:3, pages 501–518, 2018. DOI: 10.7155/jgaa.00474.
Cited on page 203.

[KMW14] Kolja B. Knauer, Piotr Micek, and Bartosz Walczak. **Outerplanar graph drawings with few slopes**. In *Comput. Geom.* volume 47:5, pages 614–624, 2014. DOI: 10.1016/j.comgeo.2014.01.003.
Cited on page 203.

[KPP13] Balázs Keszegh, János Pach, and Dömötör Pálvölgyi. **Drawing Planar Graphs of Bounded Degree with Few Slopes**. In *SIAM J. Discret. Math.* volume 27:2, pages 1171–1183, 2013. DOI: 10.1137/100815001.
Cited on page 203.

[KPPT08] Balázs Keszegh, János Pach, Dömötör Pálvölgyi, and Géza Tóth. **Drawing cubic graphs with at most five slopes**. In *Comput. Geom.* volume 40:2, pages 138–147, 2008. DOI: 10.1016/j.comgeo.2007.05.003.
Cited on page 203.

[KR19] Boris Klemz and Günter Rote. **Ordered Level Planarity and Its Relationship to Geodesic Planarity, Bi-Monotonicity, and Variations of Level Planarity**. In *ACM Trans. Algorithms* volume 15:4, pages 53:1–53:25, 2019. DOI: 10.1145/3359587.
Cited on pages 34, 107, 110, 178, 222.

[Kri18] Nadine Davina Krisam. **Drawing of Level Planar Graphs with Fixed Slopes**. Karlsruhe Institute of Technology, 2018.
Cited on page 201.

[KS81] Richard M. Karp and Michael Sipser. **Maximum Matchings in Sparse Random Graphs**. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 364–375. IEEE Computer Society, 1981. DOI: 10.1109/SFCS.1981.21.
Cited on page 31.

[KT06] Jon M. Kleinberg and Éva Tardos. **Algorithm design**. Addison-Wesley, 2006. ISBN: 978-0-321-37291-8.
Cited on page 207.

[Kur30] Casimir Kuratowski. **Sur le problème des courbes gauches en Topologie**. In *Fundamenta Mathematicae* volume 15, pages 271–283, 1930. DOI: 10.4064/fm-15-1-271-283.
Cited on page 1.

[KW17] Tomasz Krawczyk and Bartosz Walczak. **Extending Partial Representations of Trapezoid Graphs**. In *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*. Ed. by Hans L. Bodlaender and Gerhard J. Woeginger. Volume 10520 of Lecture Notes in Computer Science, pages 358–371. Springer, 2017. DOI: 10.1007/978-3-319-68705-6_27.
Cited on page 108.

[Lei98] Sebastian Leipert. **Level Planarity Testing and Embedding in Linear Time**. PhD thesis. Universität zu Köln, 1998.
Cited on pages 62, 101, 182, 187.

[LLMN13] William Lenhart, Giuseppe Liotta, Debajyoti Mondal, and Rahnuma Islam Nishat. **Planar and Plane Slope Number of Partial 2-Trees**. In *Graph Drawing - 21st International Symposium, GD 2013, Bordeaux, France, September 23-25, 2013, Revised Selected Papers*. Ed. by Stephen K. Wismath and Alexander Wolff. Volume 8242 of Lecture Notes in Computer Science, pages 412–423. Springer, 2013. DOI: 10.1007/978-3-319-03841-4_36.
Cited on page 203.

[Mac37] Saunders Mac Lane. **A Structural Characterization of Planar Combinatorial Graphs**. In *Duke Mathematical Journal* volume 3:3, pages 460–472, 1937. DOI: 10.1215/S0012-7094-37-00336-3.
Cited on pages 1, 132, 158.

[MNR16] Tamara Mchedlidze, Martin Nöllenburg, and Ignaz Rutter. **Extending Convex Partial Drawings of Graphs**. In *Algorithmica* volume 76:1, pages 47–67, 2016. DOI: 10.1007/s00453-015-0018-6.
Cited on pages 108, 203.

[MS09] Carol A. Meyers and Andreas S. Schulz. **Integer equal flows**. In *Oper. Res. Lett.* volume 37:4, pages 245–249, 2009. DOI: 10.1016/j.orl.2009.03.006.
Cited on page 215.

[MW10] Shay Mozes and Christian Wulff-Nilsen. **Shortest Paths in Planar Graphs with Real Lengths in $O(n \log^2 n/\log\log n)$ Time**. In *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part II*. Ed. by Mark de Berg and Ulrich Meyer. Volume 6347 of Lecture Notes in Computer Science, pages 206–217. Springer, 2010. DOI: 10.1007/978-3-642-15781-3_18.
Cited on page 212.

[NN20] Soeren Nickel and Martin Nöllenburg. **Towards Data-Driven Multilinear Metro Maps**. In *Diagrammatic Representation and Inference - 11th International Conference, Diagrams 2020, Tallinn, Estonia, August 24-28, 2020, Proceedings*. Ed. by Ahti-Veikko Pietarinen, Peter Chapman, Leonie Bosveld-de Smet, Valeria Giardino, James E. Corter, and Sven Linker. Volume 12169 of Lecture Notes in Computer Science, pages 153–161. Springer, 2020. DOI: 10.1007/978-3-030-54249-8_12.
Cited on page 203.

[Nöl14] Martin Nöllenburg. **A Survey on Automated Metro Map Layout Methods**. In *Schematic Mapping Workshop*. Essex, UK, Apr. 2014.
Cited on page 203.

[Pap94] Achilleas Papakostas. **Upward Planarity Testing of Outerplanar Dags**. In *Graph Drawing, DIMACS International Workshop, GD '94, Princeton, New Jersey, USA, October 10-12, 1994, Proceedings*. Ed. by Roberto Tamassia and Ioannis G. Tollis. Volume 894 of Lecture Notes in Computer Science, pages 298–306. Springer, 1994. DOI: 10.1007/3-540-58950-3_385.
Cited on page 178.

[Pat06] Maurizio Patrignani. **On Extending a Partial Straight-line Drawing**. In *Int. J. Found. Comput. Sci.* volume 17:5, pages 1061–1070, 2006. DOI: 10.1142/S0129054106004261.
Cited on pages 108, 203.

[Pat13] Maurizio Patrignani. **Planarity Testing and Embedding**. In *Handbook on Graph Drawing and Visualization*. Ed. by Roberto Tamassia. Chapman and Hall/CRC, 2013, pages 1–42.
Cited on page 61.

[PCJ97] Helen C. Purchase, Robert F. Cohen, and Murray I. James. **An Experimental Study of the Basis for Graph Drawing Algorithms**. In *ACM J. Exp. Algorithmics* volume 2, page 4, 1997. DOI: 10.1145/264216.264222.
Cited on page 106.

[Pla76] C. R. Platt. **Planar lattices and planar graphs**. In *J. Comb. Theory, Ser. B* volume 21:1, pages 30–39, 1976. DOI: 10.1016/0095-8956(76)90024-1.
Cited on page 157.

[PP06] János Pach and Dömötör Pálvölgyi. **Bounded-Degree Graphs can have Arbitrarily Large Slope Numbers**. In *Electron. J. Comb.* volume 13:1, 2006. DOI: 10.37236/1139.
Cited on page 203.

[PT04] János Pach and Géza Tóth. **Monotone drawings of planar graphs**. In *J. Graph Theory* volume 46:1, pages 39–47, 2004. DOI: 10.1002/jgt.10168.
Cited on page 34.

[PT11] János Pach and Géza Tóth. **Monotone drawings of planar graphs**. In *CoRR* volume abs/1101.0967, 2011. arXiv: 1101.0967.
Cited on page 34.

[Pur02] Helen C. Purchase. **Metrics for Graph Drawing Aesthetics**. In *J. Vis. Lang. Comput.* volume 13:5, pages 501–516, 2002. DOI: 10.1006/jvlc.2002.0232.
Cited on page 203.

[Pur97] Helen C. Purchase. **Which Aesthetic has the Greatest Effect on Human Understanding?** In *Graph Drawing, 5th International Symposium, GD '97, Rome, Italy, September 18-20, 1997, Proceedings*. Ed. by Giuseppe Di Battista. Volume 1353 of Lecture Notes in Computer Science, pages 248–261. Springer, 1997. DOI: 10.1007/3-540-63938-1_67.
Cited on page 222.

[Ran+01] Bert Randerath, Ewald Speckenmeyer, Endre Boros, Peter L. Hammer, Alexander Kogan, Kazuhisa Makino, Bruno Simeone, and Ondrej Cepek. **A Satisfiability Formulation of Problems on Level Graphs**. In *Electron. Notes Discret. Math.* volume 9, pages 269–277, 2001. DOI: 10.1016/S1571-0653(04)00327-0.
Cited on pages iv, 2, 3, 4, 33, 34, 35, 41, 59, 63, 107, 132, 157.

[Sah74] Sartaj Sahni. **Computationally Related Problems**. In *SIAM J. Comput.* volume 3:4, pages 262–279, 1974. DOI: 10.1137/0203021.
Cited on page 215.

[Sch13] Marcus Schaefer. **Toward a Theory of Planarity: Hanani-Tutte and Planarity Variants**. In *J. Graph Algorithms Appl.* volume 17:4, pages 367–440, 2013. DOI: 10.7155/jgaa.00298.
Cited on pages 107, 109.

[SH99] Wei-Kuan Shih and Wen-Lian Hsu. **A New Planarity Test**. In *Theor. Comput. Sci.* volume 223:1-2, pages 179–191, 1999. DOI: 10.1016/S0304-3975(98)00120-0.
Cited on pages 1, 15.

[Sri+02] K. Srinathan, Pranava R. Goundan, M. V. N. Ashwin Kumar, R. Nandakumar, and C. Pandu Rangan. **Theory of Equal-Flows in Networks**. In *Computing and Combinatorics, 8th Annual International Conference, COCOON 2002, Singapore, August 15-17, 2002, Proceedings*. Ed. by Oscar H. Ibarra and Louxin Zhang. Volume 2387 of Lecture Notes in Computer Science, pages 514–524. Springer, 2002. DOI: 10.1007/3-540-45655-4_55.
Cited on page 215.

[STT81] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. **Methods for Visual Understanding of Hierarchical System Structures**. In *IEEE Trans. Syst. Man Cybern.* volume 11:2, pages 109–125, 1981. DOI: 10.1109/TSMC.1981.4308636.
Cited on page 106.

[Tam87] Roberto Tamassia. **On Embedding a Graph in the Grid with the Minimum Number of Bends**. In *SIAM J. Comput.* volume 16:3, pages 421–444, 1987. DOI: 10.1137/0216030.
Cited on pages 1, 131.

[Tut66] William Thomas Tutte. **Connectivity in Graphs**. University of Toronto Press, 1966.
Cited on pages 1, 132, 158.

[Tut70] William T. Tutte. **Toward a Theory of Crossing Numbers**. In *Journal of Combinatorial Theory* volume 8:1, pages 45–53, 1970.
Cited on pages 1, 34.

[Ver90] Yves Colin de Verdière. **Sur un nouvel invariant des graphes et un critère de planarité**. In *J. Comb. Theory, Ser. B* volume 50:1, pages 11–21, 1990. DOI: 10.1016/0095-8956(90)90093-F.
Cited on page 1.

[Wag37] Klaus Wagner. **Über eine Eigenschaft der ebenen Komplexe**. In *Mathematische Annalen* volume 114, pages 570–590, 1937. DOI: 10.1007/BF01594196.
Cited on page 1.

[War62] Stephen Warshall. **A Theorem on Boolean Matrices**. In *J. ACM* volume 9:1, pages 11–12, 1962. DOI: 10.1145/321105.321107.
Cited on page 114.

[Wik19] Wikipedia contributors. **War of the Spanish Succession family tree template — Wikipedia**. Online, accessed February 20th, 2019. 2019.
URL: https://w.wiki/qpx.
Cited on page 178.