Track Layouts of Graphs

Vida Dujmović

School of Computer Science
McGill University, Montréal

October 2003

A thesis submitted to McGill University in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Contents

Abstract

Graph drawing problems originate from diverse application domains. In some, such as software engineering and cartography, graphs are required to be visualized or drawn in ways that are easy to read and understand. In others, such as VLSI design, graphs are required to be laid out while satisfying some physical constraint. For example, when a drawing is to be displayed on a page or a computer screen, or is to be used for VLSI design, it is important to keep its area/volume small to avoid wasting space.

More often than not however, the idea of a good drawing, regardless of its purpose, coincides with having no edge crossings or having very few crossings. Unfortunately, whichever of the numerous drawing styles one considers, a problem requiring a crossing minimization of sorts will, almost certainly, be $\mathcal{NP}$-hard. The theory of fixed parameter tractability (FPT) provides a new and promising approach for coping with intractable problems. In the first part of this thesis we apply algorithmic techniques developed in this theory to well-known graph drawing problems. In particular, we contribute efficient FPT algorithms for crossing minimization and planarization problems concerning the *2-layer* drawing style.

In the second part of this thesis we introduce and comprehensively study so-called *track layouts* of graphs and their subdivisions. A relationship between this combinatorial structure and several well-known types of graph layouts is established, leading to a number of new results. For example, our study of track layouts of bounded treewidth graphs settles an open problem due to Ganley and Heath (2001) regarding *queue layouts* of such graphs. Moreover, the study also establishes that graphs of bounded treewidth have three-dimensional straight-line grid drawings with linear volume.

Through the study of track layouts of subdivisions, we determine that every graph with n vertices and m edges has a three-dimensional polyline grid drawing with the vertices on a rectangular prism, $\mathcal{O}(n + m \log n)$ volume and $\mathcal{O}(\log n)$ bends per edge.

Résumé

Les problèmes de dessin de graphes proviennent de domaines d'application divers. Dans certains, tels que le génie logiciel et la cartographie, des graphes doivent être visualisés ou dessinés d'une manière facile à lire et àcomprendre. Dans d'autres, tel que la conception de VLSI, des graphes doivent être dessinés tout en satisfaisant certaines contraintes physiques. Par exemple, au moment où un schéma doit être montré sur une page ou un écran d'ordinateur, ou être employé pour la conception de VLSI, il est important de maintenir une petite aire/volume du schéma pour éviter de gaspiller de l'espace.

Le plus souvent cependant, l'idée d'un bon schéma, indépendamment de son but, coïncide avec avoir aucun ou très peu de croisements d'arêtes. Malheureusement, quelque soit le style de dessin que l'on considère, un problème exigeant une minimisation de croisements sera, presque certainement, $\mathcal{NP}$-dur. La théorie de la tractabilité fixe de paramètres (FPT) fournit une approche nouvelle et prometteuse pour faire face à certains de ces problèmes insurmontables. Dans la première partie de cette thèse nous appliquons des techniques algorithmiques dérivées de cette théorie à des problèmes de dessin de graphe bien connus. En particulier, nous présentons des algorithmes efficaces de FPT pour des problèmes de minimisation de croisements et de planarisation du modèle de dessin *2-couche*.

Dans la deuxième partie de cette thèse nous présentons et étudions en dètails les agencements voies (track layouts) des graphes et de leurs subdivisions. Un rapport entre cette structure combinatoire et plusieurs types bien connus d'agencements de graphe est établi, menant à un certain nombre de résultats nouveaux. Par exemple, notre étude des track layouts des graphes de largeur arborescente (treewidth) bornée règle un problème non résolu dû à Ganley et Heath (2001) concernant des agencements-queues (queue layouts) de tels graphes. D'ailleurs, l'étude établit également que les graphes de treewidth bornée ont des dessins tridimensionnels de grille à ligne droite avec volume linéaire, qui représentent la plus grande classe connue de graphes avec de tels dessins. Par l'étude des track layouts des subdivisions, nous déterminons que chaque graphe avec n sommets et m arêtes a un dessin de grille tridimensionnelle de polyligne avec les sommets sur un prisme rectangulaire, un volume de $\mathcal{O}(n + m \log n)$ et $\mathcal{O}(\log n)$ coudes par arête.

Declaration

This thesis contains no material which has been accepted in whole, or in part, for any other degree or diploma. Except for results whose authors are cited where first mentioned, Chapters 3, 4, 5, 6, 7 and 8 of this thesis constitute an original contribution to knowledge.

Assistance has been received only as mentioned in the following.
I wish to gratefully acknowledge the scientific collaboration instrumental to this thesis.

- Chapter 3 is based on joint work with Sue Whitesides.
- Chapter 4 is based on joint work with Michael Fellows, Michael Hallett, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Fran Rosamond, Matthew Suderman, Sue Whitesides and David Wood. New ideas about the presentation of this work emerged in discussions about the original manuscript [50] with my thesis advisor Sue Whitesides. As a result, the exposition along with many of the proofs that appear in this chapter are new, and are joint work with my advisor.
- Sections 6.2 and 8.1 are based on joint work with Pat Morin and David Wood.
- Chapters 5, 6, 7 and 8 are based on joint work with David Wood.

Much of the material in this thesis has or will appear in print [50, 53–56, 58, 59] as annotated in the bibliographic notes at the end of each chapter.

Acknowledgements

First and foremost, I would like to thank my adviser Sue Whitesides. It is impossible to overstate her role in my scientific becomings. I would like to thank her for believing in my abilities, for convincing me to do a PhD — I never looked back — and making the experience more interesting and enjoyable than I had ever imagined. Our many hours of reading, writing and discussions spent in the local cafe, have been instrumental to this thesis. In addition, she provided me with many opportunities to learn from and work with different researchers from around the world by inviting me to her workshops and encouraging me to attend many conferences.

I would like to extend my sincere appreciation to Godfried Toussaint for inviting me to his workshops, for his contagious passion for all facets of research, and for his service above and beyond the call of duty.

David Wood, thank you for teaching me persistence, for the uncountable number of hours working together in real and cyber space, for sitting next to me and making me create that phd.tex file, and above all for your friendship.

During my studies, I was fortunate to have had the opportunity to collaborate and publish with many researchers, in addition to those mentioned in the declaration. Their influence has helped in numerous ways, and I am grateful to all of them.

I thank Sue Whitesides, Hazel Everett and Sylvain Lazard for the opportunity to work on geometry problems in computer graphics. It was a worthwhile and broadening experience to do research in an area not directly related to my thesis topic. In particular, I gratefully acknowledge funding from Project ISA-McGill for the opportunity to visit INRIA-Lorraine (LORIA) in Nancy, France. While in Nancy, Sylvain and Hazel welcomed me in their home. For that and far all those hours spent working on spherical cows, I thank you.

I have been fortunate enough to learn *Graph Drawing* and *Algorithmic Motion Planning* from Sue Whitesides, *Probabilistic Analysis of Algorithms* and *Data Structures* from Luc Devroye, *Robotics* from Gregory Dudek, *Computational Geometry* from Godfried Toussaint and most recently *Graph Minors* and *Approximation Algorithms* from Bruce Reed. I thank you all.

Working in the stimulating and positive atmosphere of the Computational Geometry

Lab has been an invaluable experience. I would like to extend my appreciation to my fellow students in the lab: Greg Aloupis, Erin McLeish, Ebrahim Mal-Alla, Tallman Nkgau, Mike Soss, Steve Robbins, and especially Matthew Suderman. Our many discussions about 1- and 2-layer planarization and crossing minimization, as well as your experimental research of these problems, has been very stimulating.

Without the lunch group, including Sylvain Bouix, Luc Devroye, Stefan Langerman, Pat Morin, Ralph Neininger, Bruce Reed, Godfried Toussaint, and Sue Whitesides, my time as a graduate student would neither be as (un)productive nor as much fun. Special thanks to Luc Devroye for colouring even the grayest of Montreal's winter days, and Bruce Reed for opening doors and making the geometry lab everything but a quiet place.

I am grateful for the three years of PhD scholarship I received from FCAR and for the financial support I received from the research grant of Sue Whitesides.

To my parents, koji su me učili da mislim svojom glavom, ništa od ovog ne bi bilo moguće bez vaše podrške.

. . . and Deep, now we can go to movies.

List of Figures

List of Tables

Chapter 1

Introduction

Graphs are used to model structural information arising from many fields, such as economics, engineering, social sciences, genetics, mathematics and computer science. In chemistry, the popular ball-and-stick model of a molecule is a graph. The nodes are atoms and the edges correspond to molecular bonds. In graph models of the World-Wide Web, nodes represent web pages and edges represent hyperlinks.

Graphs, as models of information, are often required to be visualized or drawn in ways that are easy to read and understand, or they are required to be *laid out* while satisfying some physical constraint. Graph drawing addresses the problems of characterizing the existence of such drawings and layouts, as well as developing algorithmic techniques for their automatic generation. Although graph drawing problems are attractive from a purely mathematical standpoint, they also arise in many application areas, including VLSI design, visualization, and DNA mapping.

There are infinitely many drawings of a graph. Producing a good drawing of a graph typically involves the optimization of several application-specific criteria. More often than not, the idea of a good drawing, regardless of its purpose, coincides with having few edge crossings. When a drawing is to be displayed on a page or a computer screen, or is to be used for VLSI design, it is important to keep the area/volume small to avoid wasting space. A bend on an edge increases the difficulty for the eye to follow the course of the edge. For this reason, both the total number of bends and the number of bends per edge should be kept small when the readability of a drawing is of concern. For most of these cases, it is hard to achieve the optimum. Garey and Johnson [87] showed that minimizing the number of crossings is $\mathcal{NP}$-complete. Kramer and van Leeuwen [130] proved that to test whether a graph can be embedded in a grid of prescribed size with vertices at grid points is $\mathcal{NP}$-complete. Garg and Tamassia [89] proved the $\mathcal{NP}$-completeness of determining the minimum number of bends for orthogonal drawings where edges consist of vertical and horizontal line segments.

Due to the seemingly inevitable combinatorial explosion of running time as a function of problem size, most of the algorithms that attempt to find exact solutions to $\mathcal{NP}$-complete problems are in general highly impractical. The theory of fixed parameter tractability (FPT) provides a new and promising approach for coping with intractable problems. The key idea behind FPT algorithms is to isolate some aspect(s) of the input as a parameter, and to confine the exponential part of the running time to that parameter, the benefit being that this parameter will often be much smaller in practice than the size of the whole input. Researchers in many fields are now developing fast and practical FPT algorithms for problems previously considered unsolvable. One such problem is the vertex cover problem.

This thesis and its contributions can be divided into two main parts. The first part is concerned with algorithmic graph drawing problems, in particular, $\mathcal{NP}$-hard optimization problems regarding *2-layer drawings*. In a 2-layer drawing, the vertices of a graph are placed on two parallel lines (*layers*), and the edges are drawn as straight line-segments between the layers. Such drawings have been studied extensively by the graph drawing community. We initiate the study of these problems from the FPT point of view and contribute efficient FPT algorithms for three well-known problems concerning 2-layer drawings (Chapters 3 and 4).

The second main contribution of this thesis is concerned with structural graph drawing problems, that is, with characterizing the existence of, and deriving bounds for, certain types of drawings and layouts. In particular, we introduce and study comprehensively the *track layouts* of graphs and their subdivisions. Similar structures, although less general, are implicit in several previous works [80, 110, 114, 165]. A *k-track layout* of a graph consists of a vertex k-colouring, and an ordering of vertices in each colour class, such that between each pair of colour classes no two edges cross. The *track-number* of a graph G is the minimum k such that G has a k-track layout. As an outcome of this study we derive several new results for well-known models of graph layouts: *queue-layouts*, *stack layouts* (more commonly called *book embeddings*), *3D straight-line grid drawings* and finally *3D polyline grid drawings* (Chapters 5, 6, 7 and 8).

The principal results of this thesis are outlined in more detail in Section 1.3. In addition, at the beginning of each chapter we state its contributions and put them into perspective with regard to the current state of the art.

1.1 Graph layouts (drawings)

In this section we introduce the topics of graph drawing that are in the scope of this thesis and also provide the relevant background.

1.1.1 Queue and stack layouts

A *queue layout* of a graph $G = (V, E)$ consists of an ordering $<$ on the vertices $V(G)$, and a partition of the edges $E(G)$ into *queues*, such that no two edges in the same queue are *nested* with respect to $<$: two edges vw and xy are nested with respect to $<$ if $v < x < y < w$. The minimum number of queues in a queue layout of G is called the *queue-number* of G, and is denoted by $\mathsf{qn}(G)$.

A similar concept is that of a *stack layout*, which consists of an ordering $<$ on $V(G)$, and a partition of $E(G)$ into *stacks* (or *pages*) such that no two edges in the same stack *cross* with respect to $<$. Two edges vw and xy cross with respect to $<$ if $v < x < w < y$. More detailed definitions of stack and queue layouts may be found in Chapter 2.

The minimum number of stacks in a stack layout of G is called the *stack-number* of G, and is denoted by $\mathsf{sn}(G)$. A queue (stack) layout with k queues (stacks) is called a *k-queue* (*k-stack*) *layout*, and a graph that admits a k-queue (k-stack) layout is called a *k-queue* (*k-stack*) *graph*. Examples of 3-stack and 3-queue layouts of K_6 are illustrated in Figure 1.1.

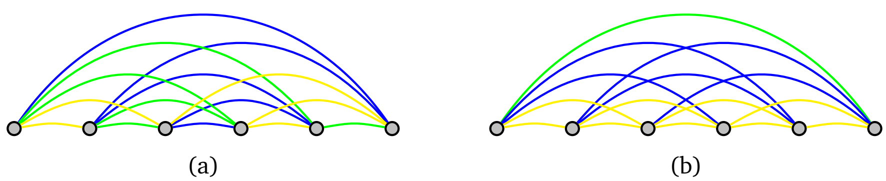


FIGURE 1.1: Layouts of K_6: (a) 3-stack, (b) 3-queue; the edges with the same colour form a stack in (a) and a queue in (b).

Stack layouts were independently introduced by Bernhart and Kainen [7] and by Cottafava and D'Antona [28]. Queue layouts were introduced by Heath *et al.* [110, 114]. Stack layouts are more commonly called *book embeddings*, and stack-number has been called *book-thickness*, *fixed outer-thickness*, and *page-number*.

Heath and Rosenberg [114] characterized 1-queue graphs as the 'arched leveled planar' graphs, and proved that it is $\mathcal{NP}$-complete to recognize such graphs. This result is in contrast to the situation for stack layouts; 1-stack graphs are precisely the outerplanar graphs [7], which can be recognized in polynomial time. However, 2-stack graphs are characterized as the subgraphs of planar Hamiltonian graphs [7], which implies that it is $\mathcal{NP}$-complete to test if the stack-number of a given graph is at most two [201]. Heath *et al.* [110] proved that 1-stack graphs are 2-queue graphs (rediscovered by Rengarajan and Veni Madhavan [165]), and that 1-queue graphs are 2-stack graphs.

While it is $\mathcal{NP}$-hard to minimize the number of stacks in a stack layout given a fixed vertex ordering [88, 189], Heath and Rosenberg [114] describe an $\mathcal{O}(m \log \log n)$ time algorithm for the analogous problem for queue layouts.

A tree is a 1-queue graph, since in a breadth-first vertex ordering of a tree no two edges are nested. Chung *et al.* [25] proved that in a depth-first vertex ordering of a tree no two edges cross. Thus trees are 1-stack graphs. Loosely speaking, *treewidth* measures how similar a graph is to a tree, and *band-width* is a measure of linearity of a graph (see Chapter 6 for the definitions). Rengarajan and Veni Madhavan [165] proved that graphs with treewidth at most two (the series-parallel graphs) are 2-stack and 3-queue graphs. Heath and Rosenberg [114] proved that for every graph G, the queue number is bounded by the bandwidth. Stack and/or queue layouts of bounded (that is, constant) treewidth graphs have been investigated in [25, 85, 165]. Ganley and Heath [85] proved that, for every graph G, the stack-number is bounded by the treewidth (using a depth-first traversal of a tree-decomposition), and asked whether queue-number is bounded by the treewidth. The principal result of Chapter 6 is to solve this question in the affirmative.

Yannakakis [207] showed that planar graphs are 4-stack graphs. The best known upper bound on the queue-number of a planar graph is $\mathcal{O}(\sqrt{n})$. Heath *et al.* [110, 114] asked whether every planar graph has $\mathcal{O}(1)$ queue-number.

Applications of stack and queue layouts include sorting permutations [78, 104, 119, 155, 162, 186], fault tolerant VLSI design [25, 171, 173, 174], parallel process scheduling [8], complexity theory [82, 83, 125], compact graph encodings [120, 148], compact routing tables [91], and graph drawing [9, 38, 202–204]. Stack and queue layouts of directed graphs [37, 105, 107, 111–113] and posets [2, 3, 134, 154, 185] have also been investigated.

Table 6.1 on page 78 summarizes some of the known bounds on the stack-number and queue-number of various classes of graphs, including the bound established in this thesis. Despite this wealth of research on stack and queue layouts, the following fundamental questions of Heath *et al.* [110] have remained unanswered.

Open Problem 1.1. [110] Is stack-number bounded by queue-number?

Open Problem 1.2. [110] Is queue-number bounded by stack-number?

Suppose that stack-number is bounded by queue-number, but queue-number is not bounded by stack-number. This would happen, for example, if there exists a constant s such that for every q there exists an s-stack graph with no q-queue layout. Then we would consider stacks to be more 'powerful' than queues, and vice versa. Note that for every sufficiently big integer n, there is an n-vertex graph G with $\mathsf{sn}(G) \geq 3^{\mathsf{qn}(G)}$ [110].

Depth-first search and breadth-first search can be thought of as the same algorithm, where depth-first search operates with a stack and breadth-first search operates with a queue. Thus stack and queue layouts of graphs are a means for measuring the relative power of depth-first search and breadth-first search. It is no coincidence that many algorithms for computing stack layouts use depth-first search [25, 85, 138], while breadth-first search is often used for computing queue layouts [110, 165]. These ideas are made particularly concrete in the case of trees (see Lemmas 5.16 and 5.17).

1.1.2 3D graph drawings

Graph drawing in the plane is well-studied (see [35, 126]). Motivated by experimental evidence suggesting that displaying a graph in three dimensions is better than in two [193, 194], and applications including information visualization [193], VLSI circuit design [135, 163, 172], and software engineering [195], there is a growing body of research in three-dimensional graph drawing.

In this thesis we study *three-dimensional straight-line grid drawings*, henceforth called *3D drawings*. In this model, vertices are positioned at distinct points in $\mathbb{Z}^3$ (called *grid-points*), and edges are drawn as straight line-segments with no crossings. (Two edges *cross* if they intersect at some point other than a common endpoint.) We focus on the problem of producing 3D drawings with small volume. This problem has been extensively studied [16, 19, 26, 36, 39, 40, 80, 106, 158, 161, 204]. Drawings of graphs in three dimensions with the vertices in $\mathbb{R}^3$ have also been studied [18, 23, 24, 32, 64, 90, 115–118, 144, 156]. Aesthetic criteria besides volume which have been considered include symmetry [115–118], aspect ratio [24, 90], angular resolution [24, 90], edge-separation [24, 90], and convexity [23, 24, 64, 180].

The classical result of de Fraysseix *et al.* [34] and Schnyder [176] states that every planar graph has an $\mathcal{O}(n^2)$ area 2D straight-line grid drawing. In contrast to the case in the plane, a folklore result states that every graph has a 3D drawing. Such a drawing can be constructed using the 'moment curve' algorithm in which vertex v_i, $1 \leq i \leq n$, is represented by the grid-point (i, i^2, i^3). It is easily seen — compare with Lemma 8.2 in Chapter 8 — that no two edges cross.

Since every graph has a 3D drawing, we are interested in optimizing certain measures of the quality of a drawing. If a 3D drawing is contained in an axis-aligned box with side lengths $X-1$, $Y-1$ and $Z-1$, then we speak of an $X \times Y \times Z$ drawing with *volume* $X \cdot Y \cdot Z$. That is, the volume of a 3D drawing is the number of gridpoints in the bounding box. This definition is formulated so that two-dimensional drawings have positive volume.

Observe that the drawings produced by the moment curve algorithm have $\mathcal{O}(n^6)$ volume. Cohen *et al.* [26] improved this bound, by proving that if p is a prime with $n < p \leq 2n$, and each vertex v_i is represented by the grid-point $(i, i^2 \bmod p, i^3 \bmod p)$, then there is still no crossing. This construction is a generalization of an analogous two-dimensional technique due to Erdös [76]. Furthermore, Cohen *et al.* [26] proved that the resulting $\mathcal{O}(n^3)$ volume bound is asymptotically optimal in the case of the complete graph K_n. It is therefore of interest to identify fixed graph parameters that allow for 3D drawings with small volume.

The first such parameter to be studied was the chromatic number [19, 158]. Calamoneri and Sterbini [19] proved that every 4-colourable graph has a 3D drawing with $\mathcal{O}(n^2)$ volume. Generalizing this result, Pach *et al.* [158] proved that graphs of bounded chromatic number have 3D drawings with $\mathcal{O}(n^2)$ volume, and that this bound is asymptotically optimal for the complete bipartite graph with equal sized bipartitions. If p is a suitably chosen prime, the main step of this algorithm represents the vertices in the ith colour class by gridpoints in the set $\{(i, t, it) : t \equiv i^2 \pmod{p}\}$. It follows that the volume bound is $\mathcal{O}(k^2n^2)$ for k-colourable graphs.

The first non-trivial $\mathcal{O}(n)$ volume bound was established by Felsner *et al.* [80] for outerplanar graphs. Their elegant algorithm "wraps" a two-dimensional drawing around a triangular prism. Poranen [161] proved that series-parallel digraphs have upward 3D drawings with $\mathcal{O}(n^3)$ volume, and that this bound can be improved to $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$ in certain special cases. Di Giacomo *et al.* [39] proved that series-parallel graphs with maximum degree three have 3D drawings with $\mathcal{O}(n)$ volume.

In a recent development, Bose *et al.* [16] proved that graphs admitting three-dimensional drawings with $\mathcal{O}(n)$ volume have $\mathcal{O}(n)$ edges. In particular, the maximum number of edges in an $X \times Y \times Z$ drawing is exactly $(2X-1)(2Y-1)(2Z-1) - XYZ$.

Straight-line drawings are a special case of polyline drawings. In particular, a *three-dimensional polyline grid drawing* of a graph, henceforth called a *3D polyline drawing*, represents the vertices by distinct points in $\mathbb{Z}^3$, and represents each edge as a polygonal chain with *bends* (if any) also at gridpoints, such that distinct edges do not cross. Here a point where a polygonal chain changes its direction is called a bend. Polyline drawings provide great flexibility as they can approximate drawings with curved edges. The number of bends, however, should be kept as small as possible, since bends typically reduce the readability of a drawing. A 3D polyline drawing with at most b bends per edge is called a *3D b-bend drawing*. Thus 0-bend drawings are 3D drawings. Of course, a 3D b-bend drawing of a graph G is precisely a 3D straight-line drawing of a subdivision of G with at most b division vertices per edge. This provides one of the motivations for our study of graph subdivisions in Chapter 7. The volume and number of bends in 3D polyline drawings where edges are restricted to be axis aligned have been previously studied; see [67, 68, 205] for example. This thesis initiates (in Chapter 8) the study of upper bounds on the volume and number of bends per edge in *arbitrary* 3D polyline drawings.

Table 8.2 on page 126 summarizes the best known upper bounds on the volume and bends per edge in 3D drawings and 3D polyline drawings, including those established in this thesis. In general, there is a trade-off between few bends and small volume in such drawings, which is evident in Table 8.2.

1.1.3 Layered (hierarchical) drawings

A common method for drawing directed graphs, which produces *layered drawings* or *hierarchical drawings* was introduced by Tomii *et al.* [188], Carpano [20] and Sugiyama *et al.* [183]. In this type of drawing, vertices are arranged on $h \geq 2$ *layers* (that is, on h parallel lines in the plane), and edges are drawn as straight line-segments between vertices on adjacent layers (as illustrated in Fig. 1.2). Layouts of this kind have applications in visualization [35], in DNA mapping [197] and in row-based VLSI layout [136]. Note that not all graphs have layered drawings, even if edge crossings are allowed. For example, a 3-cycle has no layered drawing.

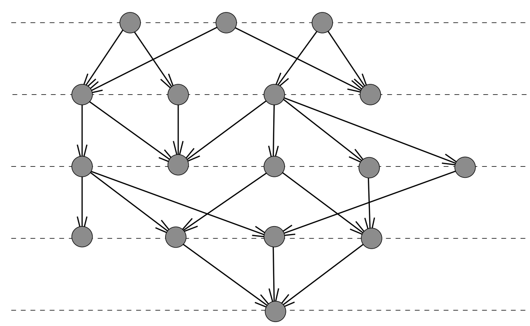


FIGURE 1.2: A layered drawing.

As is the case with other styles of drawings, the quality and readability of layered drawings depends heavily on the number of edge crossings. If the vertices of a planar graph G are not preassigned to layers, testing whether G has a layered drawing without edge crossings is an $\mathcal{NP}$-complete problem. Heath and Rosenberg [114] derived this result as part of their proof that recognizing 1-queue graphs is $\mathcal{NP}$-complete. However, if the vertices of G are preassigned to h layers as part of the input, Jünger *et al.* [121] demonstrated that there is a linear-time algorithm to test if G has a layered drawing without edge crossings subject to the vertex assignment.

Crossing minimization. The number of edge crossings in a layered drawing depends only on the ordering of the vertices within each layer, rather than on the precise coordinates of the vertices. Although this simplifies the problem in some sense (that is, the problem becomes discrete), choosing vertex orderings that minimize the number of edge crossings in layered drawings is in fact an $\mathcal{NP}$-complete problem even if there are only two layers [70]. The two layer problem was proposed by Harary [102], Harary and Schwenk [103] and Watkins [198]. They gave the first structural results for the problem. Two-layer drawings are of fundamental importance in the *layer-by-layer sweep* method introduced by Sugiyama *et al.* [183]. Most techniques for producing layered drawings first assign vertices to $h \geq 2$

layers (sometimes this vertex assignment is determined by the context), and then do a layer-by-layer sweep. An ordering π_1 for the vertices in the top layer L_1 is chosen and fixed. Then for each succeeding layer L_i, an ordering π_i is sought that minimizes the number of edge crossings among the edges between L_{i-1} and L_i. This process is repeated until some stopping criterion is satisfied.

A key step in this method is to minimize crossings between two adjacent layers when the ordering in one layer is fixed. This problem is called the 1-SIDED CROSSING MINIMIZATION problem. Unfortunately, this basic problem is also $\mathcal{NP}$-complete [70]. The problem is $\mathcal{NP}$-complete even for graphs with only degree-1 vertices in the fixed layer and vertices of degree at most 4 in the other layer [147], that is, for a forest of 4-stars. This problem is the focus of Chapter 3, where it is studied from the fixed parameter tractability point of view.

The 1-SIDED CROSSING MINIMIZATION problem has been studied extensively by the graph drawing community. Much effort has gone into the design of efficient heuristics (e.g. [21, 48, 65, 70, 183, 191, 196]). In terms of exact algorithms, Jünger and Mutzel [122] succeeded in employing integer linear programming (ILP) methods in order to find an exact solution to the 1-SIDED CROSSING MINIMIZATION problem. They first transform the problem to a linear ordering problem that is subsequently solved via the branch and cut method. In the same work the authors also surveyed numerous heuristics that have been proposed and experimentally compared their performances with the optimal solutions generated by their own method. They reported that the *iterated barycentre* method of Sugiyama *et al.* [183] performed best in practice. However, from a theoretical point of view, the *median heuristic* of Eades and Wormald [70] is a linear-time 3-approximation algorithm, whereas the barycentre heuristic is a $\Theta(\sqrt{n})$-approximation algorithm [137]. Recently, Nagamochi [152] devised a 1.47-approximation for the problem.

Planarization. Instead of minimizing the number of edge crossings, one can seek to remove the minimum number of edges such that the remaining graph has an h-layer drawing without edge crossings. A graph is *biplanar* if it has a 2-layer drawing without edge crossings. Consider a 2-layer drawing of a bipartite graph G produced by first drawing a maximum biplanar subgraph of G and then drawing all the remaining edges. Although such a drawing is unlikely to minimize the number of crossings, there is some experimental evidence to suggest that 2-layer drawings in which all the crossings occur in few edges are more readable than drawings with fewer total crossings [149].

The 2-LAYER PLANARIZATION problem asks for a minimum set of edges to be deleted from a given graph G so that the remaining graph is biplanar. When the input graph is bipartite and the ordering of one bipartition is given as part of the input, we talk about the 1-LAYER PLANARIZATION problem (or the 1-SIDED PLANARIZATION problem). The 1- and

2-Layer Planarization problems are the focus of Chapter 4, where we study both of the problems from the fixed parameter tractability point of view.

Despite the practical significance of the problems, 1- and 2-Layer Planarization have received less attention in the graph drawing literature than their crossing minimization counterparts. The 2-Layer Planarization problem is $\mathcal{NP}$-complete [69, 188], even for planar biconnected bipartite graphs with vertices in respective bipartitions having degree two and three [69]. Eades and Whitesides [69] show that the 1-Layer Planarization problem is also $\mathcal{NP}$-complete, even for graphs with only degree-1 vertices in the fixed layer and vertices of degree at most 2 in the other layer; that is, for collections of 1- and 2-paths. With the order of the vertices in both layers fixed the problem can be solved in polynomial time [69, 151].

Integer linear programming algorithms have been presented for 1- and 2-Layer Planarization [149, 151]. Shahrokhi *et al.* [179] present an $\mathcal{O}(n)$ time dynamic programming algorithm for 2-Layer Planarization of weighted trees, for which the objective is to minimize the total weight of deleted edges. Although Tomii *et al.* [188] claim an $\mathcal{O}(n^3)$ time algorithm for the 2-Layer Planarization problem on trees, Mutzel [149] demonstrates a tree for which their algorithm is not optimal.

Integer linear programming algorithms have also been developed to produce layered drawings on more than two layers. In particular, ILP techniques have been applied to both planarization and crossing minimization problems with vertices preassigned to $h > 2$ layers [108, 109, 133]. Crossing free layered drawings of trees have been investigated in [181].

1.2 Fixed parameter tractability

One of the outcomes of the last thirty years of complexity theory is the realization that most interesting computational problems are essentially intractable, being $\mathcal{NP}$-complete or worse. It has been pointed out in Garey and Johnson [86] that parameters associated with different parts of the input can interact in a wide variety of ways in producing non-polynomial complexity. Downey and Fellows [47] initiated a systematic analysis of the complexity of *parameterized decision problems*. Specifically, one of the principle ideas of parameterized complexity is to look more deeply into the structure of the input with the aim of identifying the parts (*parameters*) that contribute to intractability. Many intractable computational problems have natural parameters. The number of edge crossings in a drawing is one such parameter. Some problem parameters may be less obvious. Examples include treewidth, bandwidth, and pathwidth of an input graph. When the maximum number k of allowed edge crossings is small, an algorithm for crossing minimization whose running time is exponential in k but polynomial in the size of the graph may be useful.

The theory of *parameterized complexity* [47] addresses complexity issues of this nature, in which a problem is specified in terms of one or more parameters. This complexity theory can be viewed not only as a potential means of dealing with intractability but perhaps also as a general framework for problem analysis and algorithm design, including the design of heuristics and approximation algorithms.

Just as *polynomial time*, $\mathcal{P}$, is the basis of traditional complexity theory, the basic concept of the parameterized complexity framework is that of *fixed parameter tractability*. A problem with input size n and parameter size k is *fixed parameter tractable*, or in the *class* $\mathcal{FPT}$, if there is an algorithm to solve the problem in $f(k) \cdot n^{\alpha}$ time, where α is a constant independent of k and n, and f is an arbitrary function dependent only on parameter k. A problem in $\mathcal{FPT}$ is thus solvable in polynomial time for a fixed k. The classical example is the FPT algorithm that solves the vertex cover problem in time $\mathcal{O}(kn + 1.29^k \cdot k^2)$ [22, 46]. So the problem is well solved for input graphs of any size so long as k is no more than around 100. Yet it is not surprising that many parameterized problems appear not to be in $\mathcal{FPT}$. For instance, just as the traveling salesman problem is not likely to be in $\mathcal{P}$ according to traditional complexity theory, so is the independent set problem not likely to be in $\mathcal{FPT}$ according to parameterized complexity theory. Downey and Fellows [47] defined a whole hierarchy of parameterized decision problem classes, $\mathcal{FPT} \subseteq \mathcal{W}[1] \subseteq \mathcal{W}[2] \subseteq \ldots$ and appropriate reducibility and completeness notations. The independent set problem, for example, is $\mathcal{W}[1]$-complete.

In recent years a variety of methods useful for proving fixed parameter tractability have emerged. One of them is the celebrated *graph minors theorem* by Robertson and Seymour [166, 167]. The theorem is considered by many to be the most important result in graph theory. Its power may be illustrated through the *linking number* problem, which can be viewed as a graph drawing problem. Informally, the linking number problem asks if a graph can be embedded in $\mathbb{R}^3$ such that the maximum size of a collection of linked disjoint cycles is bounded by k. A graph is *linkless* if it has an embedding such that no pair of disjoint cycles is linked. Up until the proof of the graph minors theorem it was unknown if the problem is decidable even for $k = 0$, that is, for recognizing linkless graphs. Remarkably, by the graphs minors theorem, there is an $\mathcal{O}(n^3)$ algorithm to decide if a given graph is linkless [168]. For fixed k, the theorem implies that the linking problem is in $\mathcal{FPT}$.

Unfortunately, not only do the algorithms based on the graph minors theorem have astronomical hidden constants, but the non-constructive nature of some parts of this work make it at times difficult to conceive any kind of algorithm. Specifically, while the theorem may imply the existence of a polynomial-time algorithm for a problem, sometimes no algorithm is known. One such problem is deciding whether a graph is knotless, that is, embeddable in $\mathbb{R}^3$ without a knot.

Another major outgrowth from the work of Robertson and Seymour on graph minors, has been the exploration of a new graph parameter, *treewidth*, associated with *tree decompositions* of a graph. As already noted, the treewidth measures how similar a graph is to a tree. The fundamental idea was that many results and techniques applicable to trees should carry over to graphs that are "tree-like". Indeed, from the stand point of parameterized complexity, tree-decompositions have turned out to be a very powerful tool for deriving FPT algorithms. If a graph has bounded treewidth, then many intractable problems become tractable by dynamic programming on tree-decompositions, and by automata techniques [30]. These FPT methods, relying on tree-decompositions of graphs, are currently "approaching" practicality. The best algorithm for computing a tree-decomposition is due to Bodlaender [13]. The algorithm runs in $\mathcal{O}(2^{32k^3}|G|)$ time, where G is an input graph and k is an upper bound on its treewidth.

There is a steadily growing list of examples of FPT algorithms with more practical costs in the parameter, such as 2^k. Almost all the practical FPT algorithms, including those for the vertex cover problem mentioned above, have been derived by the important elementary methods, *bounded search tree* and *kernelization*. While very simple algorithmic strategies, they are in some sense new, as typically they have been overlooked previously because they have exponential costs in the parameter.

The idea behind the *kernelization* method (that is, the method of reduction to a *problem kernel*) is to define operations that transform (in polynomial time) a given problem instance P of size n to an "equivalent" problem instance P', where the size of P' is a function solely of the parameter k. Then instance P' is exhaustively searched for a solution. Since the size of the space that is exhaustively searched is bounded by k, the exponential part of the running time will also be a function of k only.

As its name suggests, the idea behind the *bounded search tree* method is to build a search tree for a problem. Associated with the root node of a search tree is a given problem instance. Each node of the tree branches into some number of subproblems, which typically have a smaller parameter than their "parent" instance. The critical observation for many parameterized problems in that the size of the tree is bounded by the parameter only, thus giving rise to fixed parameter tractable algorithms.

We apply the bounded search tree and kernelization methods in Chapters 3 and 4. We study 3D drawings of graphs that have bounded treewidth/pathwidth in Chapter 6.

1.2.1 Fixed parameter tractability and graph drawing

Applications of FPT techniques to hard graph drawing problems has only just begun. Thanks to the graph minors theorem, in addition to the examples mentioned in the previous section, we now know that testing if a given graph can be embedded in a surface of genus γ is in $\mathcal{FPT}$. Moreover, for fixed genus, Mohar [143] gave a linear-time algorithm for the problem.

Although conjectured [47] to be fixed parameter *intractable*, the general crossing minimization problem, where vertices are not restricted to lie on parallel lines nor are edges restricted to be straight, has been shown recently to be in $\mathcal{FPT}$. Specifically, Grohe [94] gave an $f(k) \cdot n^2$ algorithm for recognizing graphs that can be drawn in the plane with at most k crossings. A very similar approach would work for deleting k edges to leave a graph planar. Since the approach relies on deep structure theorems from the Robertson-Seymour graph minors project, the FPT result of Grohe does not yield a practical algorithm.

Dujmović *et al.* [49] have studied layered drawing problems from the fixed parameter tractability point of view. In particular, the h-LAYER CROSSING MINIMIZATION problem as well as the related h-LAYER PLANARIZATION problem are considered. Here the number of layers h is also considered to be a parameter of the problem. The h-LAYER CROSSING MINIMIZATION problem asks if a given graph can be drawn on h layers with at most k crossings. The h-LAYER PLANARIZATION problem asks if k edges can be *removed* from a given graph such that the remaining graph can be drawing on h layers without edge crossings. (The vertices of a given graph are not preassigned to layers.) It has been proved in [49], using bounded pathwidth techniques, that both these general problems (which include 1-SIDED CROSSING MINIMIZATION, and 1- and 2-LAYER PLANARIZATION) are in the class $\mathcal{FPT}$. The algorithms use a path-decomposition (a structure related to tree-decompositions) as their basis. As pointed out in the previous section, a pathwidth-based approach is only of theoretical interest, since the running time of the algorithms is dominated by the cost of finding the path-decomposition which is $\mathcal{O}(2^{32(h+2k)^3} n)$.

Although not practical, these results suggest, at least for restricted versions of the problems, that there might be more practical algorithms, that is, algorithms with more reasonable parameter functions. This provides the motivation for our study in Chapters 3 and 4.

1.3 Contributions, organization, and guidelines for the reader

Chapter 2 introduces definitions and notation used throughout the thesis.

We now describe the contributions of each chapter.

Chapter 3 We give an $\mathcal{O}(\phi^k \cdot n^2)$ fixed parameter tractable algorithm for the 1-SIDED CROSSING MINIMIZATION problem. The constant ϕ in the running time is the golden ratio $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$. The parameter k is the number of allowed edge crossings.

Chapter 4 We give an $\mathcal{O}(k \cdot 6^k + |G|)$ fixed parameter tractable algorithm for the 2-LAYER

PLANARIZATION problem and an $\mathcal{O}(3^k \cdot |G|)$ algorithm for the 1-LAYER PLANARIZATION problem. The parameter k is the number of allowed edge deletions.

Chapter 5 We study basic properties of track layouts. For instance we provide a lower bound on the number of tracks for any graph. Furthermore, several results describing how to convert one type of layout of a graph G into another type of layout of G are presented. These manipulations are critical for a number of results in the subsequent chapters.

Chapter 6 We prove that the track-number is bounded by treewidth and consequently that the queue-number is bounded by treewidth, thus resolving an open problem due to Ganley and Heath [85] (2001), and disproving a conjecture of Pemmaraju [160] (1992). This result provides renewed hope for the positive resolution of Open Problem 1.2, on page 4.

Chapter 7 We improve the best known upper bound on the number of division vertices per edge in a 3-stack subdivision of an n-vertex graph G from $\mathcal{O}(\log n)$ to $\mathcal{O}(\log \min\{\mathsf{sn}(G), \mathsf{qn}(G)\})$. Moreover, this result reduces Open Problem 1.2, whether queue-number is bounded by stack-number, to whether 3-stack graphs have bounded queue-number.

Furthermore, we prove that every graph has a 2-queue subdivision, a 4-track subdivision, and a mixed 1-stack 1-queue subdivision. All these values are optimal for every non-planar graph. In addition, the number of division vertices per edge in these subdivisions, namely $\mathcal{O}(\log \mathsf{qn}(G))$, is optimal to within a constant factor, for every graph G. The main results of this chapter are summarized in Table 7.1 on page 95.

Chapter 8 We prove that every n-vertex graph with track-number t has a 3D drawing with $\mathcal{O}(t^2 n)$ volume. Consequently, given the bounds on the track-number derived in Chapters 6 and 7, we infer that the graphs of bounded treewidth have 3D drawings with $\mathcal{O}(n)$ volume. Furthermore, every graph has a 3D polyline drawing with $\mathcal{O}(m \log n)$ volume and $\mathcal{O}(\log n)$ division vertices per edge.

We conclude and give a list of open problems in **Chapter 9**.

A reader interested in a particular subject may find the following division helpful. Although the individual chapters of this thesis are not fully self-contained, they can be arranged into topical units that *are*, as follows.

The FPT algorithm for 1-SIDED CROSSING MINIMIZATION:

Section 2.2.4 and Chapter 3

The FPT algorithms for 1- and 2-LAYER PLANARIZATION:

Section 2.2.4 and Chapter 4

Basics on track layouts:

Chapters 2 and 5

Track and queue layouts of bounded treewidth/pathwidth graphs:

Chapters 2, 5 and 6

Layouts of subdivisions:

Chapters 2, 5 and 7

3D straight-line and polyline drawings:

Chapters 2, 5, 6, and 8

Chapter 2

Preliminaries

In this chapter we introduce definitions and the notation used throughout the thesis. Undefined terms from graph theory can be found in Diestel [41].

2.1 Graphs

Throughout this thesis $G = (V, E)$ is a graph with vertex set $V(G)$ and edge set $E(G)$. When the graph is clear from the context, we will sometimes denote the vertex set by V and the edge set by E. We assume G is finite, simple and undirected unless explicitly stated otherwise.

The number of vertices and edges of G are respectively denoted by $n = |V(G)|$ and $m = |E(G)|$. The subgraph of G induced by a set of vertices $S \subseteq V(G)$ is denoted by $G[S]$. For all $A, B \subseteq V(G)$, we denote by $G[A, B]$ the bipartite subgraph of G with vertex set $A \cup B$ and edge set $\{vw \in E(G) : v \in A, w \in B\}$. The spanning subgraph of G induced by a set of edges $S \subseteq E(G)$ is denoted by $G[S]$. For a set of vertices $S \subseteq V(G)$, $G \setminus S$ denotes $G[V(G) \setminus S]$, and $G \setminus v$ denotes $G \setminus \{v\}$ for all vertices v. Similarly, for a set of edges $S \subseteq E(G)$, $G \setminus S$ denotes $G[E(G) \setminus S]$, and $G \setminus vw$ denotes $G \setminus \{vw\}$ for all edges vw.

A *subdivision* of a graph G is a graph obtained from G by replacing each edge $vw \in E(G)$ by a path $v, x_1, x_2, \dots, x_p, w$ where $p \geq 0$. Vertices on this path distinct from v and w (that is, $x_1, x_2, \dots, x_p$) are called *division* vertices.

A *clique* in a graph is a set of pairwise adjacent vertices. A *(proper) vertex t-colouring* of a graph G is a partition $\{V_i : 1 \leq i \leq t\}$ of $V(G)$ such that for every edge $vw \in E(G)$, if $v \in V_i$ and $w \in V_j$ then $i \neq j$. A set V_i, $1 \leq i \leq t$, in a vertex t-colouring of G is a *colour class*. The minimum t such that G is vertex t-colourable is the *chromatic number* of G, denoted by $\chi(G)$. A *star colouring* of G is a vertex colouring with no bichromatic 4-vertex path; that is, each bichromatic subgraph is a forest of stars. The *star chromatic number* of G, denoted by $\chi_{\text{st}}(G)$, is the minimum number of colours in a star colouring of G.

A graph H is a *minor* of a graph G if H is isomorphic to a graph obtained from a subgraph of G by contracting edges. A family of graphs closed under taking minors is *proper* if it is not the class of all graphs. The following general bound on the star chromatic number is due to Nešetřil and Ossona de Mendez [153].

Lemma 2.1. [153] *The star chromatic number of a graph G is at most a quadratic function of the maximum chromatic number of a minor of G. Hence, every proper minor-closed graph family has bounded star chromatic number.*

2.1.1 Rooted trees

Let T be a rooted tree. The vertices of T are called *nodes*, and we assume that the edges are oriented away from its root node r. This assumption on rooted trees will stand for the remainder of this thesis. A node in T with no outgoing edge is a *leaf* in T. As is standard, when referring to the edge of a directed graph, xy means an edge oriented from x to y. The *depth* of a node $x \in V(T)$ is the distance from r to x in T, and is denoted by depth(x). The *height* of T is the maximum depth of a node in T. Let deg(x), deg$^{-}(x)$, and deg$^{+}(x)$ denote the degree, indegree, and outdegree of each node $x \in V(T)$. We denote by $\rho(x)$ the parent node of each non-root node $x \in V(T)$.

2.1.2 Graph parameters

A *graph parameter* is a function α that assigns to every graph G a non-negative integer $\alpha(G)$. Let $\mathcal{G}$ be a class of graphs. By $\alpha(\mathcal{G})$ we denote the function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum of $\alpha(G)$, taken over all n-vertex graphs $G \in \mathcal{G}$. We say $\mathcal{G}$ has *bounded* α if $\alpha(\mathcal{G}) \in \mathcal{O}(1)$. A graph parameter α is *bounded by* a graph parameter β (for some class $\mathcal{G}$), if there exists a *binding* function g such that $\alpha(G) \leq g(\beta(G))$ for every graph G (in $\mathcal{G}$). If α is bounded by β (in $\mathcal{G}$) and β is bounded by α (in $\mathcal{G}$) then α and β are *tied* (in $\mathcal{G}$). Clearly, if α and β are tied then a graph family $\mathcal{G}$ has bounded α if and only if $\mathcal{G}$ has bounded β. These notions were introduced by Gyárfás [99] in relation to near-perfect graph families for which the chromatic number is bounded by the clique-number.

2.1.3 Vertex ordering

A binary relation $<$ over a set S defines a *partial order* on S if it is transitive, antisymmetric and reflexive. An ordered pair $P = (S, <)$, where $<$ is the partial order on set S is *partially ordered set* (*POSET*). A partial order is a *total order* if it also satisfies $x < y$ or $y < x$, for all distinct $x, y \in S$.

A *vertex ordering* of an n-vertex graph G is a bijection $\sigma : V(G) \to \{1, 2, \ldots, n\}$. We write $v <_\sigma w$ to mean that $\sigma(v) < \sigma(w)$. Thus $<_\sigma$ is a total order on $V(G)$. We say G (or $V(G)$)

is *ordered by* $<_\sigma$. At times, it will be convenient to express σ by the list $(v_1, v_2, \ldots, v_n)$, where $\sigma(v_i) = i$. These notions extend to subsets of vertices in the natural way. Suppose that $V_1, V_2, \ldots, V_k$ are disjoint sets of vertices, such that each V_i is ordered by $<_i$. Then $(V_1, V_2, \ldots, V_k)$ denotes the vertex ordering σ such that $v <_\sigma w$ whenever $v \in V_i$ and $w \in V_j$ with $i < j$, or $v \in V_i$, $w \in V_i$, and $v <_i w$. We write $V_1 <_\sigma V_2 <_\sigma \cdots <_\sigma V_k$. In a vertex ordering σ of a graph G, let $L(e)$ and $R(e)$ denote the endpoints of each edge $e \in E(G)$ such that $L(e) <_\sigma R(e)$.

For a set of vertices S ordered by σ, let $\overleftarrow{S}$ denote the set S ordered by the reverse vertex ordering π, where for each pair of vertices $v, w \in S$, $v <_\pi w$ if and only if $w <_\sigma v$.

A vertex ordering σ of a directed acyclic graph (DAG) G is *topological* if $v <_\sigma w$ for all edges $vw \in E(G)$.

2.2 Graph layouts

2.2.1 Stack and queue layouts

In a vertex ordering σ of a graph G, consider two edges $e, f \in E(G)$ with no common endpoint and with $L(e) <_\sigma R(e)$.

- e and f *cross*: $L(e) <_\sigma L(f) <_\sigma R(e) <_\sigma R(f)$.
- e and f *nest* and f is *nested inside* e: $L(e) <_\sigma L(f) <_\sigma R(f) <_\sigma R(e)$.

A *stack* (respectively, *queue*) in σ is a set of edges $F \subseteq E(G)$ such that no two edges in F are crossing (nested) in σ. A queue E' has a total order $\preceq$, called the *queue order*, such that

$$\forall e, f \in E', \quad e \preceq f \iff L(e) \leq_\sigma L(f) \text{ and } R(e) \leq_\sigma R(f) \ . \tag{2.1}$$

A k-*stack* (*queue*) *layout* of G is a pair $(\sigma, \{E_1, E_2, \ldots, E_k\})$ where σ is a vertex ordering of G, and $\{E_1, E_2, \ldots, E_k\}$ is a partition of $E(G)$ such that each E_i is a *stack* (*queue*) in σ. At times we write $\mathsf{stack}(e) = \ell$ (or $\mathsf{queue}(e) = \ell$) if $e \in E_\ell$.

Consider the problem of assigning the edges of a graph G to the minimum number of stacks given a fixed vertex ordering σ of G. As illustrated in Figure 2.1(a), a *twist* in σ is a matching $\{v_i w_i \in E(G) : 1 \leq i \leq k\}$ such that

$$v_1 <_\sigma v_2 <_\sigma \cdots <_\sigma v_k <_\sigma w_1 <_\sigma w_2 <_\sigma \cdots <_\sigma w_k \ .$$

A vertex ordering with a k-edge twist needs at least k stacks, since each edge of a twist must be in a distinct stack. Unfortunately the converse is not true. There exist vertex orderings with no $(k+1)$-edge twist that require $\Omega(k \log k)$ stacks [128]. Moreover, as noted in the introduction, it is $\mathcal{NP}$-complete to test if a fixed vertex ordering of a graph admits a k-stack layout [88][1]. On the other hand, Kostochka [129] proved that a vertex ordering with no 3-edge twist admits a 5-stack layout, and Ageev [1] proved that 5-stacks are sometimes necessary in this case. In general, Kostochka and Kratochvíl [128] proved that a vertex ordering with no $(k+1)$-edge twist admits a 2^{k+6}-stack layout[2], thus improving on previous bounds by Gyárfás [97, 98]. Hence the stack-number of a graph G is bounded by the minimum, taken over all vertex orderings σ of G, of the maximum number of edges in a twist in σ.

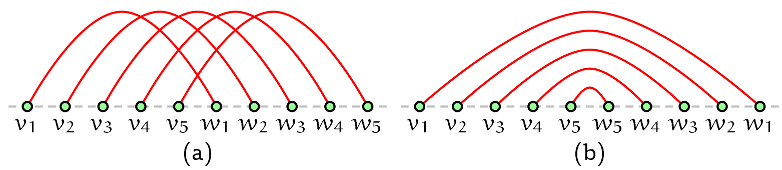


FIGURE 2.1: (a) 5-edge twist, (b) 5-edge rainbow.

Now consider the analogous problem for queue layouts: assign the edges of a graph G to the minimum number of queues given a fixed vertex ordering σ of G. As illustrated in Figure 2.1(b), a *rainbow* in σ is a matching $\{v_i w_i \in E(G) : 1 \leq i \leq k\}$ such that

$$v_1 <_\sigma v_2 <_\sigma \cdots <_\sigma v_k <_\sigma w_k <_\sigma w_{k-1} <_\sigma \cdots <_\sigma w_1 \quad .$$

The rainbow $\{v_i w_i : 2 \leq i \leq k\}$ is said to be *inside* $v_1 w_1$. We now give a new and simple proof of a result by Heath and Rosenberg [114].

Lemma 2.2. [114] *A vertex ordering of a graph G admits a k-queue layout of G if and only if it has no $(k+1)$-edge rainbow.*

Proof. A k-queue layout has no $(k+1)$-edge rainbow since each edge of a rainbow must be in a distinct queue. Conversely, suppose we have a vertex ordering with no $(k+1)$-edge rainbow. For every edge $vw \in E(G)$, let $\mathsf{queue}(vw)$ be the maximum number of edges in a rainbow inside vw plus one. If vw is nested inside xy then $\mathsf{queue}(vw) < \mathsf{queue}(xy)$. Hence we have a valid queue assignment. The number of queues is at most k. □

Thus determining $\mathsf{qn}(G)$ can be viewed as the following vertex ordering problem.

Lemma 2.3. [114] *The queue-number $\mathsf{qn}(G)$ of a graph G is the minimum, taken over all vertex orderings σ of G, of the maximum size of a rainbow in σ.* □

[1] Unger [189, 190] claimed that it is $\mathcal{NP}$-complete to determine whether a given vertex ordering of a graph G admits a 4-stack layout, and that there is an $\mathcal{O}(n \log n)$ time algorithm in the case of 3-stack layouts. Crucial details are missing from these papers.

[2] Unger [189] claimed without proof that a vertex ordering with no $(k+1)$-edge twist admits a $2k$-stack layout. This claim is refuted by Ageev [1] in the case of $k = 2$.

Heath and Rosenberg [114] presented a $\mathcal{O}(m \log \log n)$ time algorithm for computing the k-queue layout in Lemma 2.2.

2.2.2 Mixed layouts

Stack and queue layouts are generalized through the notion of a *mixed* layout. Here each edge of a graph is assigned to a stack or to a queue, defined with respect to a common vertex ordering. We speak of an *s-stack q-queue mixed layout* and an *s-stack q-queue graph*. Part of the motivation for studying mixed stack and queue layouts is that they model the double-ended queue (dequeue) data structure, since a dequeue may be simulated by two stacks and one queue.

2.2.3 Track layouts

Let $\{V_i : 1 \leq i \leq t\}$ be the colour classes in a (vertex) t-colouring of a graph G. Suppose that $<_i$ is a vertex ordering on each colour class V_i. Then each pair $(V_i, <_i)$ is a *track*, and $\{(V_i, <_i) : 1 \leq i \leq t\}$ is a *t-track assignment* of G. We say $\mathsf{track}(v) = i$ when $v \in V_i$. To ease the notation we denote track assignments by $\{V_i : 1 \leq i \leq t\}$ when the ordering on each colour class is implicit. The *span* of an edge vw in a track assignment $\{V_i : 1 \leq i \leq t\}$ is $|i - j|$ where $v \in V_i$ and $w \in V_j$. That there is a fixed ordering of the tracks in a track assignment is implicit in the definition of span. Let $\{V_{i,j} : i \geq 0, 1 \leq j \leq b_i\}$ be a track assignment of a graph G. Define the *partial span* of an edge $vw \in E(G)$ with $v \in V_{i_1,j_1}$ and $w \in V_{i_2,j_2}$ to be $|i_1 - i_2|$.

As illustrated in Figure 2.2, an *X-crossing* in a track assignment consists of two edges vw and xy such that $v <_i x$ and $y <_j w$, for distinct colours i and j.

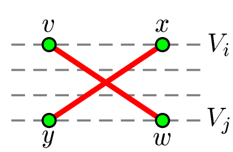


FIGURE 2.2: An X-crossing in a track assignment.

An *edge k-colouring* of G is simply a partition $\{E_i : 1 \leq i \leq k\}$ of $E(G)$. An edge $vw \in E_i$ is said to be *coloured* i, written $\mathsf{col}(vw) = i$. A (k, t)*-track layout* of G consists of a t-track assignment of G and an edge k-colouring of G with no monochromatic X-crossing. A graph admitting a (k, t)-track layout is called a (k, t)*-track graph*. The minimum t such that a graph G is a (k, t)-track graph is denoted by $\mathsf{tn}_k(G)$.

$(1, t)$-track layouts (that is, with no X-crossing) will be of particular interest due to applications in three-dimensional graph drawing (see Chapter 8). We often refer to such track layouts as *monochromatic track layouts*. A $(1, t)$-track layout is called a *t-track layout*. A graph admitting a t-track layout is called a *t-track graph*. The *track-number* of G is $\mathsf{tn}_1(G)$, simply denoted by $\mathsf{tn}(G)$.

Note: A track layout that allows edges between consecutive vertices in a track is called an *improper track layout*. This concept, in the case of three tracks, is implicit in the work of Felsner *et al.* [80], who proved that every outerplanar graph has an improper 3-track layout. The following observation gives a compelling reason to only consider proper track layouts.[3].

Observation 2.1. *If a graph G has an improper t-track layout, then G has a $2t$-track layout.*

Proof. For each track $(V_i, <_i)$ of an improper t-track layout of G, move every second vertex from track $(V_i, <_i)$ to a new track $(V_i', <_i)$. Clearly there are no X-crossings and no edges within the tracks. Thus we obtain a $2t$-track layout of G. □

Hence the track-number of a graph is at most twice its 'improper track-number'. For this reason, in this thesis we choose not to consider improper track layouts.

2.2.4 2-Layer drawings

In a *2-layer drawing* of a bipartite graph $G = (A, B; E)$, the vertices in A are positioned on a line in the plane, which is parallel to a different line containing the vertices in B, and the edges are drawn as straight line-segments. A *biplanar graph* is a bipartite graph that admits a 2-layer drawing with no edge crossings. Note that by the definition, being biplanar and having a 2-track layout are equivalent notions.

Biplanar graphs are easily characterized as shown first by Harary and Schwenk [103]. A *caterpillar* is a tree such that deleting the leaves gives a (possibly empty) path.

Lemma 2.4. [103] *Let G be a graph. The following are equivalent:*

(a) G is biplanar.

(b) G has a 2-track layout.

(c) G is a forest of caterpillars (as illustrated in Figure 2.3).

[3] In [53, 204] we called a track layout an *ordered layering with no X-crossing and no intra-layer edges*, and an improper track layout was called an *ordered layering with no X-crossing*.

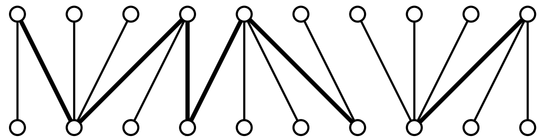


FIGURE 2.3: Forest of caterpillars.

2.2.5 3D straight-line and polyline drawings

A *three-dimensional straight-line grid drawing* of a graph, henceforth called a *3D drawing*, represents the vertices by distinct points in $\mathbb{Z}^3$ (called *grid-points*), and represents each edge as a line-segment between its endpoints, such that edges only intersect at common endpoints, and an edge only intersects a vertex that is an endpoint of that edge.

Straight-line drawings are a special case of polyline drawings. In particular, a *three-dimensional polyline grid drawing* of a graph, henceforth called a *3D polyline drawing*, represents the vertices by distinct gridpoints, and represents each edge as a polygonal chain between its endpoints with bends (if any) also at gridpoints, such that distinct edges only intersect at common endpoints, and each edge only intersects a vertex that is an endpoint of that edge. Here a point where a polygonal chain changes its direction is called a *bend*. A 3D polyline drawing with at most b bends per edge is called a *3D b-bend drawing*. Thus 0-bend drawings are 3D drawings. Of course, a 3D b-bend drawing of a graph G is precisely a 3D straight-line drawing of a subdivision of G with at most b division vertices per edge.

The *bounding box* of a 3D (polyline) drawing is the minimum axis-aligned box containing the drawing. If the bounding box has side lengths $X-1$, $Y-1$ and $Z-1$, then we speak of an $X \times Y \times Z$ (polyline) drawing with *volume* $X \cdot Y \cdot Z$. That is, the volume of a 3D (polyline) drawing is the number of gridpoints in the bounding box. This definition is formulated so that two-dimensional drawings have positive volume.

2.3 Bibliographic notes

Observation 2.1 first appeared in [52]. The proof of Lemma 2.2 is a part of [55].

Chapter 3

Crossing Minimization

In this chapter we study a parameterized analogue of the 1-SIDED CROSSING MINIMIZATION problem, where the parameter k is the number of allowed edge crossings. In particular, given a bipartite graph $G = (A, B, E)$, an integer k and a vertex ordering π_A of A, the 1-SIDED CROSSING MINIMIZATION problem asks if there is a 2-layer drawing of G that respects π_A and that has at most k crossings. We give an FPT algorithm for the problem that runs in $\mathcal{O}(\phi^k|B|^2 + |A||B|)$ time, where the constant ϕ is the golden ratio. The algorithm is based on the bounded search tree method.

As noted in the introduction, the more general version of this problem, the h-LAYER CROSSING MINIMIZATION problem, has been shown [49] to be in the class $\mathcal{FPT}$ by the bounded pathwidth method. However, the running time obtained for the 1-SIDED CROSSING MINIMIZATION problem is $\mathcal{O}(2^{256k^3}n)$, so the algorithm is impractical even for $k = 1$.

For small values of k, our algorithm should find an optimal solution for the 1-SIDED CROSSING MINIMIZATION problem in a reasonable amount of time. Of course, for dense graphs both approaches are highly impractical. We may argue however, that to some extent dense graphs are of little interest. From the practical point of view, an instance with a high number of crossings in its optimal drawing, is hardly worthwhile optimizing since the resulting drawing will be unreadable anyway. From the theoretical point of view, not only is the problem still $\mathcal{NP}$-complete for very sparse graphs [147], but in addition Eades and Wormald [70] proved that the ratio of the number of crossings in an arbitrary 2-layer drawing to the number of crossings in an optimal 2-layer drawing approaches 1 if graphs become dense.

The remainder of the chapter is organized as follows. After definitions and preliminary results in Section 3.1, we study properties of optimal drawings in Section 3.2. Our algorithm for the 1-SIDED CROSSING MINIMIZATION problem is then given in Section 3.3. Two common generalizations are addressed in Section 3.4. Final remarks are presented in Section 3.5. Section 3.6 constitutes the appendix to the chapter.

3.1 Preliminaries

In this section we introduce notation, formalize the problem statement, and recall some well-known facts about the problem.

Let $G = (A, B, E)$ denote a bipartite graph with vertex set $V(G) = A \cup B$ and edge set $E(G) \subseteq A \times B$. We study the following problem.

Problem:1-SIDED CROSSING MINIMIZATION
Instance: a bipartite graph $G = (A, B; E)$, an integer k, and a fixed vertex ordering π_A of A on the top layer.
Question: Is there a 2-layer drawing of G that respects π_A and has at most k crossings?

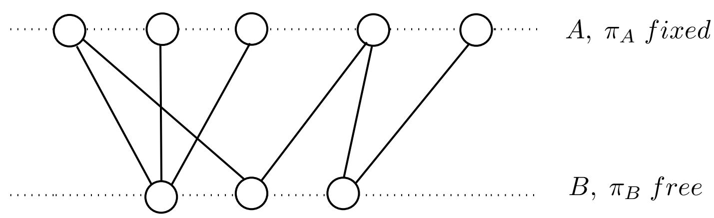


FIGURE 3.1: A 2-layer drawing. The vertex ordering π_A of A is *fixed*. Vertices of B are *free*.

From now on, we assume that input graphs are bipartite, with minimum degree at least 1, and that a vertex ordering π_A has been specified for the top layer. In addition to denoting a bipartition of G, let A also denote the top, *fixed* layer, whose vertex ordering π_A is fixed. Similarly, let B denote the bottom, *free* layer, whose vertices are free to be permuted. Figure 3.1 illustrates this terminology. We do not consider multiple edges, although these are easy to handle. Section 3.4.1 discusses this.

Let $\langle G, \pi_A, k\rangle$ denote an instance of the 1-SIDED CROSSING MINIMIZATION problem, and let (G, π_A, π_B) denote a combinatorial representation of a 2-layer drawing of G, with π_A and π_B giving the vertex orderings for the vertices on layers A and B, respectively. Let the number of crossings in the drawing (G, π_A, π_B) be denoted by $\mathsf{cr}(G, \pi_A, \pi_B)$, and let the minimum possible number of crossings subject to the vertices of A being ordered by π_A be denoted by $\mathsf{cr}(G, \pi_A, \pi_{opt})$. Note that $\mathsf{cr}(G, \pi_A, \pi_{opt}) = \min_{\pi_B}\{\mathsf{cr}(G, \pi_A, \pi_B)\ \}$, where π_B ranges over all permutations for B. Let $v < w$ denote an ordered pair of vertices on the same layer, and let v, w denote an unordered pair of vertices. Sometimes it is convenient to denote unordered pairs of vertices by (v, w), which is also used to denote an edge. The meaning will be clear from the context. Throughout this chapter, the term "pair" refers to a pair of *distinct* objects. The following two simple observations reinforce the important fact that the 1-SIDED CROSSING MINIMIZATION problem is combinatorial in nature.

Fact 3.1. *Two edges (v, v') and (w, w'), where $v, w \in B$ and $v', w' \in A$, cross in a 2-layer drawing if and only if $v < w$ and $w' < v'$, or $w < v$ and $v' < w'$.*

Fact 3.2. *For vertices v and w in the free layer B, the number of crossings of the edges incident to v with the edges incident to w is completely determined by the relative ordering of v and w.*

Having pointed out that the nature of the problem fundamentally concerns vertex orderings, the next definition and fact relate the orderings of pairs to the total number of edge crossings in any drawing, the goal of our optimization. In fact, from pairs alone, we get a lower and upper bound for $\mathsf{cr}(G, \pi_A, \pi_{opt})$.

Definition 3.1. Consider a problem instance $\langle G, \pi_A, k\rangle$, and let v and w be vertices in B. The *crossing number* c_{vw} is the number of crossings that edges incident with v make with edges incident with w in drawings having $v < w$; the *crossing number* c_{wv} is for $w < v$.

Fact 3.3. *[35] The total number of crossings in a 2-layer drawing (G, π_A, π_B) is:*

$$\mathsf{cr}(G, \pi_A, \pi_B) = \sum_{\forall v<w\in\pi_B} c_{vw}, \tag{3.1}$$

where the summation is over all ordered pairs $v < w$ of elements of π_2; furthermore,

$$\sum_{v,w\in B} \min(c_{vw}, c_{wv}) \leq \mathsf{cr}(G, \pi_A, \pi_{opt}) \leq \sum_{v,w\in B} \max(c_{vw}, c_{wv}), \tag{3.2}$$

where the summations are over all unordered pairs v, w of vertices of B.

Let $\mathsf{lb}(G, \pi_A)$ denote the lower bound $\sum_{v,w\in B} \min(c_{vw}, c_{wv})$. As pointed out in the introduction, Eades and Wormald [70] showed that $\mathsf{cr}(G, \pi_A, \pi_{opt}) \leq 3\,\mathsf{lb}(G, \pi_A)$. Recently, Nagamochi [152] improved this to $\mathsf{cr}(G, \pi_A, \pi_{opt}) \leq 1.47\,\mathsf{lb}(G, \pi_A)$.

3.2 Properties of optimal drawings

In this section, we establish a property of optimal drawings (G, π_A, π_{opt}) that will be fundamental for our algorithm in the next section (see Lemma 3.1).

For each vertex v in B, let l_v denote the leftmost neighbor of v in A, and let r_v denote the rightmost neighbor of v in A. Note that if $v \in B$ has degree 1, then $l_v = r_v$.

Now consider two vertices v and w in B. We say that v and w are a *suited pair* if $r_v \leq l_w$, or if $r_w \leq l_v$; otherwise we call the pair *unsuited*. For example, in Fig. 3.2 v, u is a suited pair and so is pair w, u; pair v, w is not suited. If v and w each have degree 1 and have the same neighbor in A (i.e. $l_v{=}r_v{=}l_w{=}r_w$), we say that v and w are a *trivial* suited pair. The following fact, which is an immediate consequence of Fact 1 and the definition of unsuited pair, indicates the importance of the notion of suitable pairs.

Fact 3.4. *A pair of vertices $v, w \in B$ is unsuited if and only if $c_{vw} \geq 1$ and $c_{wv} \geq 1$.*

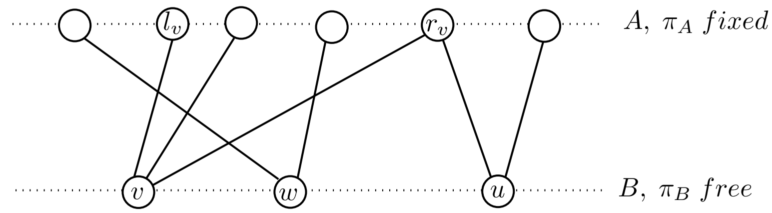


FIGURE 3.2: Pair v, w is unsuited, while v, u and w, u are suited pairs.

On the other hand, the edges of a suited pair v, w do not cross if v and w appear in their *natural ordering* in π_B, i.e., if $v < w$ when $r_v \leq l_w$, and $w < v$ when $r_w \leq l_v$. If v, w is a trivial suited pair, then $c_{vw} = 0$ and $c_{wv} = 0$, and we say that both $v < w$ and $w < v$ are natural orderings for the pair v, w.

The notion of natural ordering leads to a useful fact for our algorithm. Let deg(v) denote the degree of a vertex v.

Fact 3.5. *Suppose v, w is a suited pair for π_A with natural ordering $v < w$. Then for any π_B, the drawing (G, π_A, π_B) satisfies: (i) c_{vw}=0; (ii) if $r_v \neq l_w$, then c_{wv}= deg$(v) \cdot$ deg(w); (iii) if $r_v = l_w$, then c_{wv}= (deg$(v) \cdot$ deg$(w)) - 1$; and finally, (iv) unless v and w are a trivial suited pair, $c_{wv} > 0$.*

Note that natural ordering is only defined for pairs of suited vertices. The following lemma is the basis for our algorithm.

Lemma 3.1. *For fixed π_A, let $\Gamma_{opt} = (G, \pi_A, \pi_{opt})$ be a drawing with the minimum possible number of crossings. Then all suited pairs appear in π_{opt} in their natural ordering.*

To prove Lemma 3.1 the following lemma will be useful.

Lemma 3.2. *For $1 \leq i \leq |B|$, let v_i denote the vertex in the i^{th} position in π_B of some drawing (G, π_A, π_B) with π_A fixed. Moving any vertex $v_i \in B$ from its starting position i across the t consecutive vertices $v_{i+1}, v_{i+2}, \ldots, v_{i+t}$ to the right creates a new drawing (G, π_A, π'_B) with:*

$$cr(G, \pi_A, \pi'_B) = \mathsf{cr}(G, \pi_A, \pi_B) \ + \ \sum_{j=1}^{t} (c_{v_{i+j}v_i} \ - \ c_{v_i v_{i+j}}). \tag{3.3}$$

Similarly, if v_i is moved to the left over t consecutive vertices, then the above summation is from $j = -1$ to $j = -t$ and the sign in front of the summation is "$-$".

Proof. Assume, without loss of generality, that vertex v_i moves to the right across t consecutive vertices in (G, π_A, π_B). This creates a new drawing (G, π_A, π'_B). For instance, in Figure 3.3, v_i moves from its starting position over $t = 2$ (shaded) vertices to its final position. Dotted lines depict v_i and its incident edges as they travel across 2 shaded vertices

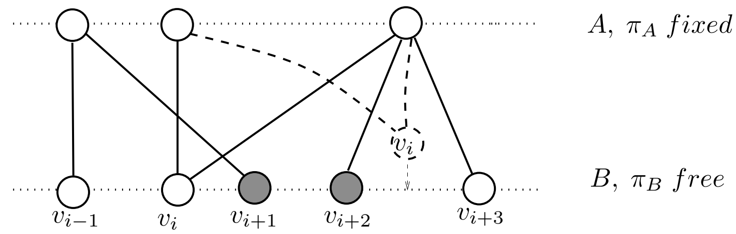


FIGURE 3.3: Illustration for the proof of Lemma 3.2.

to the new position of v_i, between v_{i+2} and v_{i+3}. The only pairs of vertices in (G, π_A, π_B) whose relative ordering changes in (G, π_A, π'_B) are the pairs v_i, v_j for $i+1 \leq j \leq i+t$. Hence by Fact 3.2, these are the only pairs whose crossing number might change. In particular, the crossing number for a pair v_i, v_j for j in the range $[i+1, i+t]$ changes from $c_{v_i v_j}$ to $c_{v_j v_i}$. Substituting these changes into equation (3.1) of Fact 3.3 gives equation (3.3) above. □

Now we give the proof of Lemma 3.1.

Proof of Lemma 3.1. The proof is by contradiction. Assume that in $\Gamma_{opt} = (G, \pi_A, \pi_{opt})$ there is a suited pair v, w whose ordering in π_{opt} is not its natural ordering. Note that v and w are not degree-1 vertices with a common neighbor, as both orderings would be natural in that case. Assume that $v < w$ is the (unique) natural ordering of v, w.

By Fact 3.5 the crossing number $c_{vw} = 0$ and the crossing number $c_{wv} > 0$.

For a contradiction, we now prove that either v or w can be moved in Γ_{opt} such that the resulting drawing $\Gamma_{new} = (G, \pi_A, \pi_{new})$ satisfies $cr(\Gamma_{new}) < cr(\Gamma_{opt})$.

Let i and j denote the positions of w and v, respectively, in π_{opt}. Here $i < j$ since v and w appear in the order $w < v$ in π_{opt}.

If $|j - i| = 1$, we can interchange v and w without affecting any other pair of vertices in Γ_{opt}. Equation (3.3) in Lemma 3.2 gives the number of crossings in the resulting drawing Γ_{new}:

$$cr(\Gamma_{new}) = cr(\Gamma_{opt}) - c_{wv} + c_{vw} = cr(\Gamma_{opt}) - c_{wv} + 0.$$

Since $c_{wv} > 0$, we have $cr(\Gamma_{new}) < cr(\Gamma_{opt})$, which contradicts the optimality of Γ_{opt}.

If $|j - i| > 1$, let $u_{i+1}, u_{i+2}, \ldots, u_{j-1}$ denote the vertices between w and v in π_{opt}, listed in order of appearance in π_{opt}. Regard these vertices as a frozen block U inside which no changes are made. Figure 3.4 illustrates this terminology.

According to Lemma 3.2, moving v or w from one side of block U to the other may only affect the crossing number contributions of pairs of the form u, w and u, v for $u \in U$. Let c_{Up} denote the number of crossings that the edges incident to vertices in U have with the

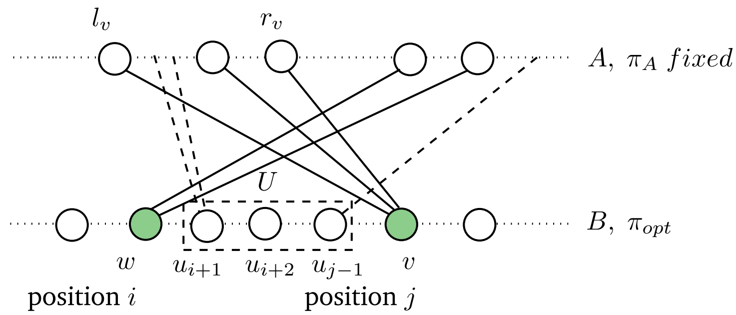


FIGURE 3.4: Γ_{opt} for case $|j-i| > 1$ of Lemma 3.1

edges incident to a vertex p to the right of U: $c_{Up} = \Sigma_{u \in U}\ c_{up}$. Similarly, let c_{pU} denote this number of crossings when p lies to the left of U.

Since Γ_{opt} is optimal, we claim we have the strict inequality

$$c_{Uv} < c_{vU}. \tag{3.4}$$

Otherwise, we could move v to the left side of U and then interchange v with w to obtain a drawing with the following total number of crossings: $cr(\Gamma_{new}) = \Gamma_{opt} - c_{Uv} + c_{vU} - c_{wv} + 0$. Since $c_{wv} > 0$, if $c_{Uv} \geq c_{vU}$, then $cr(\Gamma_{new}) < cr(\Gamma_{opt})$, a contradiction.

Observation: To conclude the case $|j - i| > 1$, it suffices to show that $c_{Uv} < c_{vU}$ implies $c_{wU} \geq c_{Uw}$, for this means we can move w to the right side of U without increasing the total number of crossings in the resulting drawing and then interchange w and v to produce a drawing with fewer crossings than Γ_{opt}. This gives a contradiction, and so proves that the assumption that π_{opt} contains a suited pair not ordered by its natural ordering cannot hold.

To establish the desired inequality $c_{wU} \geq c_{Uw}$, we first derive some intermediate inequalities for c_{Uv}, c_{vU}, c_{wU}, and c_{Uw} in terms of sizes of the following sets: E_R = the set of edges in Γ_{opt} with one endpoint in U and the other endpoint strictly greater than r_v in the vertex ordering π_A; E_L = the set of edges with one endpoint in U and the other endpoint strictly less than r_v in the vertex ordering π_A; N_v = the neighbors of v; and N_w = the neighbors of w.

By the definition of E_R, all the vertices in N_v occur in π_A strictly before the A endpoint of each edge in E_R. By the definition of E_L and by the fact that v, w is a suited pair with the natural ordering $v < w$, the vertices in N_w occur in π_A strictly after the A endpoints of the edges in E_L.

Fact 3.2 implies the following inequalities for crossing numbers:

$c_{Uv} \geq$ deg$(v) \cdot |E_R|$: The edges incident to v and the edges in E_R all pairwise intersect, creating deg$(v) \cdot |E_R|$ crossings. Since E_R is a subset of the edges incident to U, $c_{Uv} \geq$ deg$(v) \cdot |E_R|$.

$c_{vU} \leq \deg(v) \cdot |E_L|$: This holds because no edge incident to v crosses any edge incident to U that is not in E_L.

$c_{wU} \geq \deg(w) \cdot |E_L|$: The edges incident to w and the edges in E_L all pairwise intersect, so $c_{wU} \geq \deg(w) \cdot |E_L|$.

$c_{Uw} \leq \deg(w) \cdot |E_R|$: This holds because no edge incident to w intersects any edge incident to U that is not in E_R.

Recall inequality (3.4), that $c_{Uv} < c_{vU}$. Since $c_{Uv} \geq \deg(v) \cdot |E_R|$ and $c_{vU} \leq \deg(v) \cdot |E_L|$, we have $\deg(v) \cdot |E_R| \leq c_{Uv} < c_{vU} \leq \deg(v) \cdot |E_L|$, which implies that $|E_R| < |E_L|$. This, and the fact that $c_{wU} \geq \deg(w) \cdot |E_L|$, and the fact that $c_{Uw} \leq \deg(w) \cdot |E_R|$ together imply that $c_{wU} > c_{Uw}$. By the *observation* above, this completes the proof. □

3.3 An efficient FPT algorithm

3.3.1 The bounded search tree approach for the algorithm

In this section we present an FPT algorithm for the 1-SIDED CROSSING MINIMIZATION problem based on the bounded search tree approach. The key observations for building a search tree for this problem lie in Lemma 3.1 and Fact 3.4. Here is an overview of our algorithm.

Lemma 3.1 allows us, at the start, to fix the relative ordering of each non-trivial suited pair of vertices in B according to its unique natural ordering. The remaining unordered pairs of vertices in B are either trivial suited pairs, or unsuited pairs which will each, by Fact 3.4, create a crossing no matter which relative ordering is chosen. We build a search tree based on the unsuited pairs. (It turns out that the trivial pairs neighbours can be dealt with later in the algorithm.) The input to every node of the search tree is a budget b giving the remaining number of allowed edge crossings, and a relation D containing all pairs of B ordered thus far. We will formally define relation D shortly. At each node of the search tree some unordered pair (v, w) is chosen (i.e. a pair not in D) such that $c_{vw} \neq c_{wv}$. Then the node branches to two recursive subproblems. In one branch, the ordering of (v, w) is fixed to $v < w$ and the budget b is reduced by c_{vw}. In the other branch the ordering of (v, w) is fixed to $w < v$ and b is reduced by c_{wv}. Since we only work with unsuited pairs in building the tree, we know that $c_{vw} \geq 1$ and $c_{wv} \geq 1$. Therefore, since the initial budget $b = k$, the height of the search tree is at most k.

As a matter a fact the situation is better than that, for two reasons. Firstly, since $c_{vw} \neq c_{wv}$, then either c_{vw} or c_{wv} is at least 2, so one of the two branches of the search tree node reduces b by at least 2. Secondly, since $<$ is a transitive relation, fixing an ordering of the pair (v, w) at a node of the search may in fact impose an ordering of another as yet unordered pair (p, q) in the relation D at that node. Hence b can be reduced not only by c_{vw}, but also by either c_{pq} or c_{qp}, depending on which relative ordering is imposed on (p, q).

3.3.2 The algorithm

The following definitions will be useful for the description of the algorithm.

Let D be a directed acyclic graph (DAG) that represents a binary relation $<$ on the set of vertices B. In particular, the vertices of B are represented by nodes of a DAG D and an ordered pair of vertices $v < w$ is represented by a directed edge from v to w (denoted henceforth by vw) in D. The DAG D is stored as an $|B| \times |B|$ matrix. We use D to denote both the set of pairs in the current binary relation " $<$ " and the associated DAG that represents these pairs as directed edges. The algorithm labels nodes in the search tree with DAGs. The DAG associated with the root will be transitively closed. As the algorithm progresses, it computes a DAG label for each child node it generates in the search by choosing a directed edge to add to the DAG of the parent and then taking the transitive closure of this. The following algorithm solves the 1-SIDED CROSSING MINIMIZATION problem.

Algorithm: 1-Sided Crossing Minimization
Input: $\langle G, \pi_A, k \rangle$
Output: π_{opt} if $\langle G, \pi_A, k \rangle$ is a YES instance, else NO

Step 0. **Computing crossing numbers:** Compute the crossing numbers c_{vw} and c_{wv} for all pairs of vertices in B, stopping the computation of a particular crossing number as soon as it is known to exceed k. (The algorithm for computing the crossing numbers c_{vw} and c_{wv} efficiently can be found in Section 3.6.)

Step 1. **Checking for extreme values**: Compare k with the upper and lower bound as per Fact 3.3.

if $k < \Sigma_{(v,w)} \min(c_{vw}, c_{wv})$ then output NO and HALT;
if $k \geq \Sigma_{(v,w)} \max(c_{vw}, c_{wv})$ then output an arbitrary π_B and HALT.

Step 2. **Initialization**: Precompute the following information required by the search tree.

$C = \{(v, w) | c_{vw} = c_{wv}\}$;

D_0 = a DAG (V, E), where $V = B$, and the directed edges $vw \in E$ correspond to the naturally ordered pairs (v, w) that satisfy $c_{vw} = 0$ and $c_{wv} \neq 0$. (It is easy to check that D_0 is transitively closed.);

b_0 = initial budget = $k - \Sigma_{vw \in D_0} c_{vw} - \Sigma_{(v,w) \in C} c_{vw}$. Note that $\Sigma_{vw \in D_0} c_{vw} = 0$. Also note that we reduce the budget k by the eventual cost of the pairs in C even though these pairs do not appear in D_0.

Step 3. **Building and exploring the search tree**: This step simultaneously builds and explores the search tree. A node of the search tree has at most two children. Each node has a label (D, b). The label D of a node represents a "possible" partial solution, i.e. a partial order of vertices of B. The label b represents the remaining budget for crossings.

We now build the search tree as follows. Label the root of the tree with (D, b) where $D = D_0$ and $b = b_0$. In general, for a non-leaf node labeled (D, b), choose a pair (v, w) such that D contains no edge joining v and w and such that $c_{vw} \neq c_{wv}$. A pair (v, w) is thus an unordered pair not in C. In any vertex ordering π_B, the pair (v, w) is ordered as either vw or wv, so we create at most two children (D_1, b_1) and (D_2, b_2) of the non-leaf node (D, b) corresponding to these two possibilities. No child is created if its budget would be negative. Thus a node labeled (D, b) is a leaf if and only if, either it does not have an unordered pair $(v, w) \notin C$; or it has an unordered pair (v, w) for which both b_1 and b_2 are negative.
For a non-leaf node (D, b), we label one of its two children by (D_1, b_1) where:

$D_1 =$ transitive closure of $D \cup vw$, and $b_1 = b - c_{vw} - \sum_{pq} c_{pq}$.

Here $D \cup vw$ represents the addition of directed edge vw to D. The summation in b_1 is over the directed edges that are added to $D \cup vw$ by the transitive closure and that have $c_{pq} \neq c_{qp}$. That is, the sum is over pq s.t. $pq \in D_1$ and $pq \notin D \cup vw$ and $c_{pq} \neq c_{qp}$.
Similarly, we label the other child of node (D, b) with (D_2, b_2), where:

$D_2 =$ transitive closure of $D \cup wv$, and $b_2 = b - c_{wv} - \sum_{pq} c_{pq}$.

These concepts are illustrated in Figure 3.5.
If a leaf is created whose label D has the property that

$\forall (v, w)$ if $vw \notin D$ and $wv \notin D$ then $(v, w) \in C$,

then output pit as topological sort of D. Also, update the minimum number of crossings found so far to $k - b$, where b is the budget of the leaf, and update the best vertex ordering so far to π_B. We call such a node a *solution leaf*.

If, after exploring the entire tree, no solution leaf is found, output NO and HALT; otherwise, output the best vertex ordering found, which is π_{opt}, and HALT.

In Step 0, we stop computing c_{vw} as soon as it becomes $k + 1$, even though c_{vw} may be bigger than that. This is because a child with $v < w$ would have a negative budget. Hence it suffices to know that $c_{vw} \geq k + 1$.

Step 3 of the algorithm effectively creates and explores the search tree simultaneously. This can be done by depth-first search, or by breath-first search. The depth-first way requires less space and is thus the preferred choice.

Also notice that, when creating a child by choosing, say, to order v and w as vw, we reduce the budget for the child by an amount computed not only for the ordered pair vw, but also for the pairs that are newly ordered by the transitive closure of $D \cup vw$. However, we only do this for newly ordered pairs whose two crossing numbers are not the same.

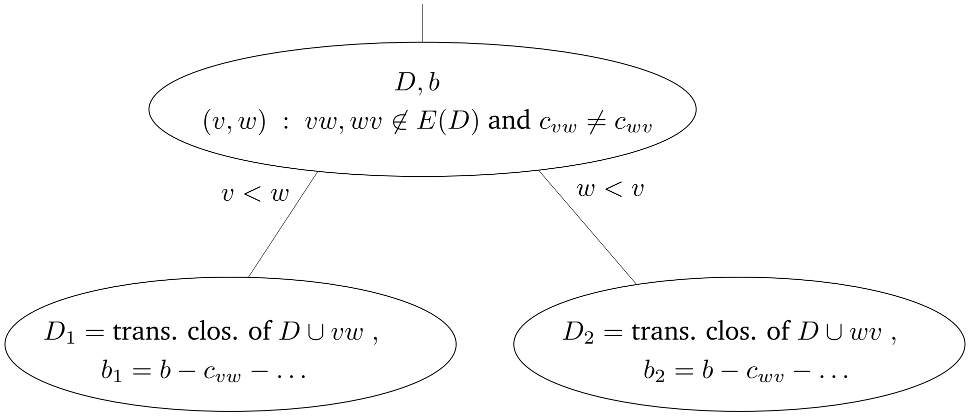


FIGURE 3.5: Illustration for step 3 of the algorithm.

Those whose crossing numbers are the same have already been accounted for in b_0.

Theorem 3.1. *Given a bipartite graph $G = (A, B; E)$, a fixed vertex ordering π_A of A, and an integer k, algorithm* 1-Sided Crossing Minimization(G, π_A, k) *determines in $\mathcal{O}(\phi^k \cdot |B|^2 + |A||B|)$ time if* $\mathsf{cr}(G, \pi_A, \pi_{opt}) \leq k$ *and if yes produces a 2-layer drawing (G, π_A, π_B) with the optimal number of crossings. The constant ϕ in the running time is the golden ratio $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$.*

Proof. Step 3 simultaneously creates and explores the search tree. For every node (D, b) of the search tree we maintain the following two invariants: (i) D is a transitively closed, directed, acyclic graph; (ii) The budget b at node (D, b) is

$$b = k - \sum_{vw \in C} c_{vw} - \sum_{vw \in D \& vw \notin C} c_{vw} \ .$$

This is true for the root node (D_0, b_0). Suppose this is true for a node labeled (D, b). At this node, the algorithm chooses an unordered pair (v, w) with $c_{vw} \neq c_{wv}$. This pair is used to create up to two child nodes. We claim that both $D \cup vw$ and $D \cup wv$ are acyclic. Suppose, on the contrary, that $D \cup vw$ contains a directed cycle. Then D must contain a directed path from w to v. Since D is transitively closed, it contains edge wv, contradicting the fact that (v, w) is unordered in D. Similarly for $D \cup wv$. Since the transitive closure of a directed acyclic graph is again acyclic, the graph labels D_1, D_2 for any child nodes created at a node labeled D are again transitive and acyclic. Thus all the graph labels in the search tree are directed, acyclic, and transitively closed. The fact that labels b_1 and b_2 agree with formula (ii), follows directly from the formulas used to compute these two labels from the parent label b in Step 3 of the algorithm.

As the tree is built, either a solution leaf is found, or the tree is completely explored without finding such a leaf. A solution leaf (D, b) has, by definition, a non-negative budget b. By the invariant (ii), the cost of all the crossings arising from the ordered pairs in D has been taken into account. Furthermore, all pairs (v, w) not ordered by directed edges in D are in C and satisfy $c_{vw} = c_{wv}$; hence the total cost directly attributable to them has already been deducted from the initial budget k. Hence any topological sort of D produces a vertex ordering consistent with D and having total cost $k - b$, where $0 \leq b \leq k$. By the invariant (i), the label D of every node of the search tree is an acyclic graph and it necessarily has a topological sort. Based on this argument, a solution leaf (D, b) encodes a vertex ordering π_B such that $cr(G, \pi_A, \pi_B) \leq k$.

It is not difficult to verify that the solution leaves of the search tree implicitly store all the vertex orderings π_B for which $cr(G, \pi_A, \pi_B) \leq k$ and in which all the suited pairs are ordered by their natural ordering. Lemma 3.1 implies that in order to decide if $\langle G, \pi_A, k\rangle$ is a YES or NO instance it is enough to consider only such vertex orderings π_B. Therefore, if there is a vertex ordering π_B such that $cr(G, \pi_A, \pi_B) \leq k$ the algorithm finds one. In fact, since the algorithm updates the best solution found so far, when it terminates it outputs an optimal vertex ordering π_{opt}.

We now discuss the running time of the algorithm.

Only for an unordered unsuited pair v, w that has $c_{vw} \neq c_{wv}$ are child nodes created, of which there are at most two. Therefore, in one child node the budget is reduced by at least 1 and in the other by at least 2. A node with $b = 0$ must be a leaf node, because any child of such a node would have a negative budget. Therefore, no further branching is allowed. At a node for which budget $b = 1$, at most one child can have a budget b_1 that is non-negative, and in this case, $b_1 = 0$ and the child must be a leaf. Thus a recurrence relation that generates an upper bound for the number of nodes in this search tree is:

$$s_b = s_{b-1} + s_{b-2} + 1 \text{ for } b \geq 2;\ s_0 = 1,\ s_1 = 2\ .$$

It can be verified by induction that for $b \geq 0$, $s_b = F_{b+2} + F_{b+1} - 1$ where F_b is the b-th Fibonacci number. From the bound on Fibonacci numbers with $b_0 \leq k$, it follows that $s_{b_0} < \frac{\phi^{k+2}}{\sqrt{5}} + \frac{\phi^{k}+1}{\sqrt{5}} - 1 < 1.2 \cdot \phi^{k+1}$. Thus the search tree has $\mathcal{O}(\phi^k)$ nodes.

The time taken at each node of the search tree is dominated by updating a transitive closure of its label D after insertion of one ordered pair $v < w$ (or $w < v$). Updating the transitive closure after one insertion can be done in $\mathcal{O}(|B|^2)$ time (see problem 25-1, page 641 in [27]). These updates are needed to generate the labels for the children, of which there are at most two. Thus the time taken in the third step of the 1-Sided Crossing Minimization algorithm is $\mathcal{O}(\phi^k \cdot |B|^2)$.

It can be shown that the time taken in steps 0 - 2 of the algorithm is $\mathcal{O}(k \cdot |B|^2 + |A||B|)$. (For details see Section 3.6.) Thus the total running time of the algorithm is $\mathcal{O}(\phi^k \cdot |B|^2 + |A||B|)$. □

Corollary 3.1. *Given a bipartite graph $G = (A, B; E)$, and a fixed vertex ordering π_A of A, algorithm* 1-Sided Crossing Minimization$(G, \pi_A, \lceil 1.47\,\mathsf{lb}(G, \pi_A)\rceil)$ *produces a 2-layer drawing* (G, π_A, π_B) *with the optimal number of crossings in* $\mathcal{O}(\phi^{\mathsf{cr}(G,\pi_A,\pi_{opt})} \cdot |B|^2 + |A||B|)$ *time.*

Proof. By the results of [152] and Fact 3.3, $\mathsf{lb}(G, \pi_A) \leq \mathsf{cr}(G, \pi_A, \pi_{opt}) \leq 1.47\,\mathsf{lb}(G, \pi_A)$. Therefore, by Theorem 3.1, the algorithm 1-Sided Crossing Minimization$(G, \pi_A, \lceil 1.47\,\mathsf{lb}(G, \pi_A)\rceil)$ finds the optimal solution.

Since parameter k is not given as a part of the input, and is instead set to the value $k = \lceil 1.47\,\mathsf{lb}(G, \pi_A)\rceil$ we need to consider the time it takes to compute $\mathsf{lb}(G, \pi_A)$. To determine this lower bound, we need to compute the smaller of the two crossing numbers, c_{vw} and c_{wv}, for each pair of vertices v, w. This can be easily achieved by slightly modifying the algorithm in Section 3.6. In particular, the while loop needs to compute both crossing numbers simultaneously until one of them, say c_{vw}, is completed. If at that moment $c_{vw} \leq c_{wv}$, then the while loop terminates, otherwise it continues computing c_{wv} until either c_{wv} becomes greater than c_{vw} or, until the computation of c_{vw} terminates. For the same reasons as those presented in the proof of Lemma 3.3, the running time of this modified algorithm is $\mathcal{O}(\mathsf{lb}(G, \pi_A)|B|^2 + |A||B|) \in \mathcal{O}(\mathsf{cr}(G, \pi_A, \pi_{opt})|B|^2 + |A||B|)$. This together with Theorem 3.1 implies the running time claimed in this corollary. □

3.4 Two generalizations

In this section we show how to deal with multiple edges and extend the algorithm to allow drawing edges within single layers.

3.4.1 Multiple edges

We now briefly discuss how to deal with instances of the 1-SIDED CROSSING MINIMIZATION problem that have multiple edges. For every pair of vertices connected by s edges, replace the edges by one edge with *weight* s. Thus, we obtain a variant of the 1-SIDED CROSSING MINIMIZATION problem where each edge has a positive weight. If two edges weighted s_1 and s_2 cross in a drawing, their contribution to the total number of crossings is $s_1 \cdot s_2$. Having this in mind the Definition 3.1 of crossing numbers c_{vw} and c_{wv} for a pair of vertices v, w remains the same. All the other definitions, facts and lemmas, except for Fact 3.5 and Lemma 3.1, follow through without any changes. We now modify the definition of deg(v) to mean the sum of the weights of all the edges incident to a vertex v. Then case (iii) of Fact 3.5 becomes $c_{wv} = \deg(v) \cdot \deg(w) - s_{vr_v} \cdot s_{wl_w}$ where s_{vr_v} and s_{wl_w} denote the weights of the two edges vr_v and wl_w. In the proof of Lemma 3.1, we also modify the definition of $|E_L|$ and $|E_R|$ to mean the sum of the weights of all the edges in the sets E_L and E_R, respectively. The correctness of the lemma, and hence the whole algorithm, follows through without any other changes.

3.4.2 Improper 2-layer drawings

Our 1-Sided Crossing Minimization algorithm can easily be extended to manage the version of 2-layer drawings called *improper 2-layer drawings*. Here, as in improper track layouts, edges are allowed between consecutive vertices in the same layer (see [80]).

Let $G[A]$ and $G[B]$ be the graphs induced by the vertex sets A and B, respectively, and suppose that $G[A]$ and $G[B]$ are forests of paths. Let P be a path in $G[B]$ with vertices $p_1, p_2, \ldots p_j$. Furthermore, let c_P be the minimum number of crossings amongst the edges incident to a path P in one of the two possible ways to draw that path P in π_B (one way is to have $p_1 p_2 \ldots p_j$ consecutively in π_B, and the other it to have $p_j p_{j-1} \ldots p_1$ consecutively in π_B).

The algorithm is now modified by adding the following preprocessing step. For each path P in $G[B]$, all the edges (p_i, p_{i+1}) of P in $G[B]$ are contracted into one vertex. Consequently, the parameter k is reduced by c_P for each path P (as illustrated in Figure 3.6). Contracting all the paths in $G[B]$ gives an instance of 1-SIDED CROSSING MINIMIZATION problem that may have multiple edges, which our algorithm can deal with as described earlier in this section. This completes the description of the modifications to the original algorithm.

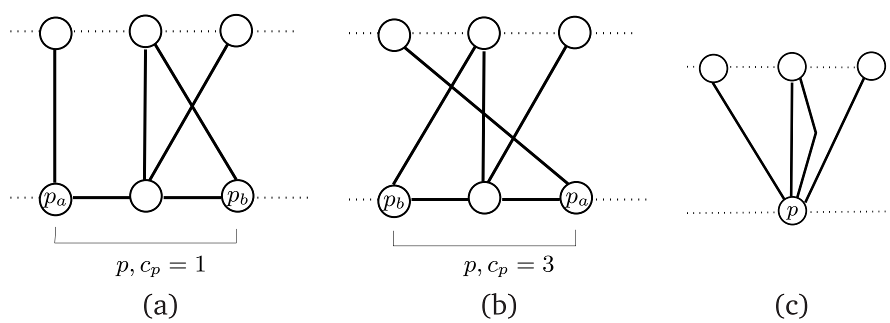


FIGURE 3.6: Edges of P are depicted in bold. (a)(b) The two ways to draw P. The first creates 1 crossing, the second creates 3 crossings; (c) After P is contracted, k is reduced by $c_P = \min\{1, 3\} = 1$.

3.5 Conclusion and bibliographic notes

In this chapter, we have studied the 1-SIDED CROSSING MINIMIZATION problem and presented an efficient FPT algorithm for its solution. Moreover, the algorithm finds a drawing with the smallest possible number of crossings in the case that this number does not exceed k. In case an optimal solution is desired no matter how many crossings it has, as Corollary 3.1 points out, our algorithm finds it by setting k to $1.47\,\mathsf{lb}(G,\pi_A)$, in time $\mathcal{O}(\phi^{\mathsf{lb}(G,\pi_A)} \cdot n^2) \in \mathcal{O}(\phi^{\mathsf{cr}(G,\pi_A,\pi_{opt})} \cdot n^2)$.

The exponential part of the running time of the algorithm is 1.618^k. In many instances the base of this exponent will be even smaller. The reason is that a pair of vertices v, w will often have both crossing numbers c_{vw} and c_{wv} bigger than 1, or at least one of them bigger than 2; thus each time a node of the search tree branches on such a pair of vertices, the resulting search tree will be even smaller. Furthermore, in a branch of the search tree, the ordering of more than one pair v, w will often be fixed due to the transitivity property.

In fact, the author together with Henning Fernau and Michael Kaufmann, has recently improved the results presented in this chapter and obtained an $\mathcal{O}(1.466^k + kn^2)$ algorithm for the 1-SIDED CROSSING MINIMIZATION problem [51]. Instrumental to this improvement is the exploitation of transitivity, and a study of the structural properties of pairs v, w with $c_{vw} = 1$ and $c_{wv} = 2$. In addition to improving the bounded search tree algorithm, these authors also derived a set of reduction operations that reduce each problem instance to an equivalent instance of size at most $1.5\,k$, thus obtaining a small problem kernel for the problem.

From the practical standpoint, an interesting investigation would be to compare experimentally the performance of the other known method for optimal 1-SIDED CROSSING MINIMIZATION, namely, integer linear programming [122], with our FPT algorithm. In the case of the related 2-layer planarization problem, recent experimental comparisons carried out by Suderman and Whitesides [182] suggest that the FPT method is competitive with the ILP method. Hence an experimental study of 1-SIDED CROSSING MINIMIZATION would be worthwhile. [1]

From the theoretical standpoint, investigating parameters other than the total number of crossings for the 1-SIDED CROSSING MINIMIZATION problem could be advantageous. Henning Fernau [private communication, 2003] suggested the following interesting problem.

Open Problem 3.1. [H. Fernau (2003)] Is the 1-SIDED CROSSING MINIMIZATION problem in the class $\mathcal{FPT}$ when parameterized by the number of crossings by which a 2-layer drawing is allowed to exceed the lower bound?

The results of this chapter have appeared in [54].

[1] In fact, my colleague Matthew Suderman has already begun this study.

3.6 Computing crossing numbers

For completeness, to justify the correctness of the running time claimed in Theorem 3.1, we give here an $\mathcal{O}(k|B|^2 + |A||B|)$ time algorithm for Step 0 of the 1-Sided Crossing Minimization algorithm. The algorithm computes the crossing numbers c_{vw} and c_{wv} for all pairs of vertices. Although the algorithm is simple, some care needs to be taken not to exceed the claimed running time. For instance, we must stop the computation of a particular crossing number as soon as it is known to exceed k. In place of crossing numbers that do exceed k we record some number strictly bigger than k.

Let the graph G in the input instance $\langle G, \pi_A, k\rangle$ be given as an $|B| \times |A|$ adjacency matrix $A = [a_{i,j}]$. The columns are labeled by the vertices of A in the order of their appearance in π_A. The rows are labeled by the vertices of B. An element $a_{i,j} = 1$ if vertex i of B is adjacent to vertex j of A; otherwise $a_{i,j} = 0$. We augment every element $a_{i,j}$ of the adjacency matrix A with the following information:

$p_{i,j}$: the index of the first neighbor of vertex i that is to the right of j. More precisely, if $\exists j' > j$ s.t. $a_{i,j'} = 1$ and $a_{i,j''} = 0, \forall j < j'' < j'$ then $p_{i,j} = j'$; otherwise $p_{i,j} = |A| + 1$.

$r_{i,j}$: the number of neighbors of i that are to the right of j. More precisely, $r_{i,j} = \sum_{j'=j+1}^{|A|} a_{i,j'}$.

In addition, for every row i, we store the following information:

l_i : the left-most neighbor of vertex i. More precisely $l_i = j$ where $a_{i,j} = 1$ and $a_{i,j'} = 0, \forall j' < j$

r_i : the right-most neighbor of vertex i. More precisely, $r_i = j$ where $a_{i,j} = 1$ and $a_{i,j'} = 0, \forall j' > j$.

The following algorithm commutes an $|B| \times |B|$ matrix $C = [c_{v,w}]$. At the end of the algorithm, the matrix entry $c_{v,w}$ equals the crossing number c_{vw} provided this is equal or less than k; otherwise, the entry $c_{v,w}$ equals some number greater than k.

Algorithm: Pair Crossing Numbers
Input: $|A| \times |B|$ adjacency matrix A of G
Output: $|B| \times |B|$ matrix C

---

1. Augment adjacency matrix A as described above.
2. **for** $v = 1$ to $|B|$ **do**
3.   **for** $w = 1$ to $|B|$ **do**
4.   **if** $v \neq w$ **then**
5. $c_{v,w} = 0$ /* initialize $c_{v,w}$ */

6. $w' = l_w$ /* start examining the neighbours w' of w starting with the left-most neighbour, l_w */
7. **while** $w' \leq r_w$ and $w' < r_v$ and $c_{v,w} \leq k$ **do**
8. $c_{v,w} = c_{v,w} + r_{v,w'}$ /* increment $c_{v,w}$ by the number of crossing points on edge (w, w') created by edges incident to v */
9. $w' = p_{w,w'}$ /* advance to the next neighbour of w */

Lemma 3.3. *The* Pair Crossing Numbers *algorithm computes the exact values of all the crossing numbers that do not exceed k in time $\mathcal{O}(k|B|^2 + |A||B|)$.*

Proof. Lines $5 - 9$ of the algorithm compute the crossing number c_{vw} for a pair of vertices v, w ordered $v < w$. The correctness of the computation follows from the next observation.

By Fact 3.1, for an ordered pair $v < w$, the number of edges incident to v that cross an edge (w, w') incident to w is precisely the number of neighbours of v strictly to the right of w'. By definition that number is $r_{v,w'}$. Thus $c_{vw} = \sum r_{v,w'}$ where the sum is over all w' adjacent to w. This formula is still correct if the sum is taken only over $\{w' \mid w' < r_v\}$, since otherwise $r_{v,w'} = 0$ by Fact 3.1.

Now consider the complexity of this algorithm. It is simple to verify that the original matrix A can be augmented by traversing each of its rows once from right to left. Per one iteration of the inner for-loop, the while-loop is executed at most $k + 1$ times, since each execution of the while loop increases c_{vw} by at least one. Given that the inner for-loop is executed $|B|^2$ times, the total running time of the algorithm is $\mathcal{O}(k|B|^2 + |A||B|)$. $\square$

Chapter 4

Planarization

In this chapter we study a parameterized analogue of the 1- and 2-LAYER PLANARIZATION problems, where the parameter k is the number of allowed edge deletions. In particular, the 2-LAYER PLANARIZATION problem asks if k edges can be deleted from a given graph G so that the remaining graph is biplanar. If the vertex ordering in one layer is fixed, then we speak of the 1-LAYER PLANARIZATION (or 1-SIDED PLANARIZATION) problem. We prove that these problems are fixed-parameter tractable. Specifically, by the kernelization method we obtain an $\mathcal{O}(\sqrt{k} \cdot 17^k + |G|)$ time algorithm for the 2-LAYER PLANARIZATION problem, which we improve to $\mathcal{O}(k \cdot 6^k + |G|)$ using the bounded search tree method combined with kernelization. Furthermore, we solve the 1-LAYER PLANARIZATION problem in $\mathcal{O}(3^k \cdot |G|)$ time using the bounded search tree method. As a by-product of this study, we derive a polynomial-time 3-approximation algorithm for the optimization version of the problem.

As noted in the introduction, in the companion work [49], we proved using bounded pathwidth techniques that the h-layer generalizations of the 1-and 2-LAYER PLANARIZATION problems are in $\mathcal{FPT}$, where the number of layers h is also considered a parameter of the problem. The running time of the algorithm is $\mathcal{O}(2^{32(h+2k)^3} n)$.

This chapter is organized as follows. After definitions and preliminary results in Section 4.1, we apply kernelization and bounded search tree methods to the 2-LAYER PLANARIZATION problem in Section 4.2. In Section 4.3 we consider the 1-LAYER PLANARIZATION problem, and present a bounded search tree algorithm for its solution. Section 4.4 describes constant approximation algorithms for the optimization versions of the 1- and 2-LAYER PLANARIZATION problems. We give final remarks in Section 4.5.

4.1 Preliminaries

In this section we introduce notation, recall a characterization of biplanar graphs and formalize the problem statements.

A vertex with degree one is a *leaf*. If vw is the edge incident to a leaf w, then we say w is a *leaf at* v and vw is a *leaf-edge at* v. The *non-leaf degree* of a vertex v in graph G is the number of non-leaf edges at v in G, and is denoted by $\deg'_G(v)$, or $\deg'(v)$ if the graph G is clear from the context.

4.1.1 Biplanar graphs

A graph is a *caterpillar* if deleting all the leaves produces a (possibly empty) path, as illustrated in Figure 4.1(a). This path is the *spine* of the caterpillar. A *2-claw* is a graph consisting of one degree-3 vertex, the *centre*, coloured black in Figure 4.1(b), which is adjacent to three degree-2 vertices, coloured gray in Figure 4.1(b), each of which is adjacent to the centre and one leaf. The edges of a 2-claw C that are incident to its center are called *primary* edges of C. The edges of C that are incident to the leaves of C are called *secondary* edges of C.

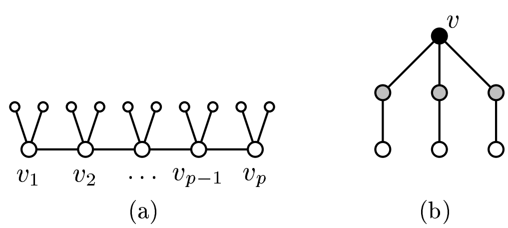


FIGURE 4.1: (a) caterpillar with spine $v_1, \dots, v_p$, (b) 2-claw centred at v.

As described in Lemma 2.4, biplanar graphs are easily characterized, and there is a simple linear-time algorithm to recognize biplanar graphs. The next lemma recalls and augments the characterization given in Lemma 2.4.

Lemma 4.1 ([66, 103, 188]). *Let G be a graph. The following are equivalent:*

(a) G is biplanar.

(b) G is a forest of caterpillars (see Figure 4.2).

(c) G is acyclic and contains no 2-claw.

(d) The graph obtained from G by deleting all leaves is a forest and contains no vertex of degree three or greater.

Lemma 4.1 implies that any planarization algorithm must destroy all cycles and 2-claws. Hence the vertices with non-leaf degree at least three are of particular interest since each such vertex lies on a cycle or a 2-claw, as demonstrated in the next lemma.

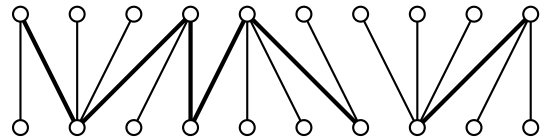


FIGURE 4.2: A biplanar graph is a forest of caterpillars. Spine edges are dark.

Lemma 4.2. *If there exists a vertex v in a graph G such that* $\text{deg'}_G(v) \geq 3$ *then G contains a 2-claw or a 3- or 4-cycle containing v.*

Proof. Let w_1, w_2, w_3 be three distinct non-leaf neighbours of v. If some pair of these neighbours is adjacent then there is a 3-cycle containing v. Otherwise, let x_i be a neighbour of w_i such that $x_i \neq v$, $1 \leq i \leq 3$. Such an x_i exists since w_i is not a leaf. If all x_i are distinct then $\{v, w_1, w_2, w_3, x_1, x_2, x_3\}$ forms a 2-claw, otherwise G contains a 4-cycle through v. □

We define $V_3 = \{v \in V : \text{deg'}(v) \geq 3\}$ and $V_3' = \{w \in V \setminus V_3 : \text{deg}(w) \geq 2,\ and\ \exists v \in V_3\ s.t.\ vw \in E\}$. That is, V_3 is the set of vertices with at least three non-leaf neighbours, and V_3' is the set of non-leaf neighbours of vertices in V_3 that are not themselves in V_3. Observe that the centre of a 2-claw is in V_3. In Figure 4.1 and subsequent illustrations, vertices in V_3 are black, vertices in V_3' are gray and vertices that belong to neither V_3 nor V_3' are white.

4.1.2 Problem statements

A set T of edges of a (not necessarily bipartite) graph G is called a *biplanarizing set* if $G \setminus T$ is biplanar. The *bipartite planarization number* of a graph G, denoted by $\mathsf{bpr}(G)$, is the size of a minimum biplanarizing set for G. Thus the 2-LAYER PLANARIZATION problem is: given a graph G and an integer k, is $\mathsf{bpr}(G) \leq k$? For a given bipartite graph $G = (A, B; E)$ and a vertex ordering π of A, the *1-layer biplanarization number* of G and π, denoted by $\mathsf{bpr}(G, \pi)$, is the minimum number of edges in G whose deletion produces a graph that admits a biplanar drawing with π as the vertex ordering of the vertices in A. The 1-LAYER PLANARIZATION problem asks if $\mathsf{bpr}(G, \pi) \leq k$.

4.1.3 Terminology

To describe our kernelization algorithm we introduce some terminology. A *component caterpillar* of a graph is a connected component that is a caterpillar. Let $P = (v_1, v_2, \ldots, v_p)$ be a path in G with $p \geq 3$ vertices. If $\text{deg'}_G(v_1) \geq 3$, $\text{deg'}_G(v_i) = 2$ for all i, $1 < i < p$, and $\text{deg'}_G(v_p) = 1$, then P together with all the leaves at vertices $v_2, \ldots, v_p$ comprises a *pendant caterpillar*. A pendant caterpillar is said to be *connected* at v_1, its *connection point*. If $\text{deg'}_G(v_1) \geq 3$, $\text{deg'}_G(v_i) = 2$ for all i, $1 < i < p$, and $\text{deg'}_G(v_p) = 3$, then P together with all the leaves at vertices $v_2, \ldots, v_{p-1}$ comprises an *internal caterpillar*. An internal caterpillar is said to be *connected* at v_1 and v_p, its *connection points*. An internal caterpillar where $p = 4$, $\deg_G(v_2) = 2$ and $\deg_G(v_3) = 2$ is called an *internal 3-path*. Edge v_2v_3 in an internal 3-path is called its *middle edge*. The *size* of a pendant (or internal) caterpillar is equal to the total number of its edges.

These graphs are illustrated in Figure 4.3.

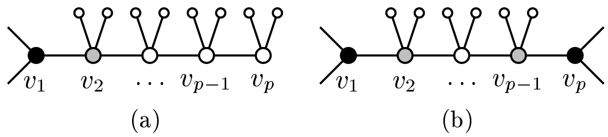


FIGURE 4.3: (a) pendant caterpillar, (b) internal caterpillar.

A graph consisting of a cycle and possibly some leaf-edges attached to the cycle is a *sun*. A *component sun* of a graph is a connected component that is a sun. Let $C = (v_1, v_2, \ldots, v_p)$ denote a cycle in G, where $v_1 = v_p$. If deg'$_G(v_1 = v_p) \geq 3$, and deg'$_G(v_i) = 2$ for all i, $1 < i < p$, then C together with all the leaves at vertices $v_2, \ldots, v_{p-1}$ comprises a *pendant sun*. A pendant sun is said to be *connected* at v_1. The *size* of a pendant sun is equal to the total number of its edges. A pendant sun of size three is called a *pendant triangle*. Edge v_2v_3 in a pendant triangle is called its *middle edge*. Edges that lie on C are called *cycle edges* of a (pendant) sun, and C is called a *sun cycle*. These graphs are illustrated in Figure 4.4.

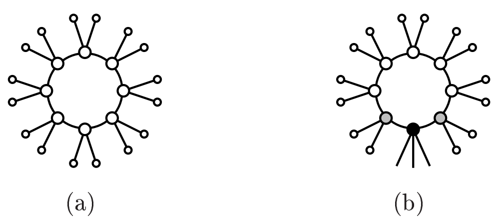


FIGURE 4.4: (a) sun, (b) pendant sun.

Notice that a connected graph that does not have a vertex v with deg'$(v) \geq 3$ is either a caterpillar or a sun, and that any two of the structures defined above are edge-disjoint. For example, an edge of G cannot belong to two internal caterpillars, or to a pendant caterpillar and an internal caterpillar. In particular, these structures are maximal: for example an internal caterpillar cannot contain another internal caterpillar.

4.2 2-Layer planarization

We now give an overview of our approach to the 2-LAYER PLANARIZATION problem. In the next section, we show that to solve the 2-LAYER PLANARIZATION problem, it suffices to search through a subset of the edges in the given input graph. In particular, for an input graph $G = (V, E)$, we define a *candidate* set of edges $\mathcal{K} \subseteq E$, and prove that $\mathcal{K}$

contains a minimum biplanarizing set of G. Moreover, as established in Section 4.2.2, the size of the candidate set and the biplanarization number of G are tied, specifically, $\mathsf{bpr}(G) \leq |\mathcal{K}| \leq 6\,\mathsf{bpr}(G)$. This automatically gives rise to an FPT algorithm that runs in $\mathcal{O}(17^k|G|)$ time. By devising appropriate reduction rules in Section 4.2.3, we obtain a problem kernel for the 2-LAYER PLANARIZATION problem, and consequently improve the running time to $\mathcal{O}(\sqrt{k}\,17^k + |G|)$. All this in combination with the bounded search tree method applied in Section 4.2.4, gives rise to our final FPT algorithm for the 2-LAYER PLANARIZATION problem that runs in $\mathcal{O}(\sqrt{k}\,6^k + |G|)$.

4.2.1 The candidate set

We define a *candidate set* $\mathcal{K}$ to be a set of edges $\mathcal{K} \subseteq E$ that contains

- any one edge from the cycle of each component sun,
- every edge vw such that $v \in V_3$, $w \in V_3 \cup V_3'$, and vw is neither in an internal 3-path nor in a pendant triangle,
- every middle edge.

An edge $e \in E$ is *good* if $e \in \mathcal{K}$, and *bad* otherwise. When convenient, we will refer to vertices of G in V_3 as *black*, those in V_3' as *gray*, and vertices that are neither in V_3 nor in V_3' as *white*. The following theorem is the basis for our kernelization algorithm (see Section 4.2.3).

Theorem 4.1. *Let T be a biplanarizing set of G. There exists a biplanarizing set T^* of G such that $T^* \subseteq \mathcal{K}$ and $|T^*| \leq |T|$.*

The remainder of this section is dedicated to proving Theorem 4.1. We do this by demonstrating that every bad edge in a biplanarizing set can be replaced by a good edge. In order to simplify the proof we first rule out the trivial cases in the next lemma. We say that a biplanarizing set is *normal* if it contains neither leaf-edges of G nor bad edges that belong to component suns of G. We will use the following simple observations which follow directly from the definitions.

Fact 4.1. *Every cycle of G has at least one good edge. Every 2-claw of G has at least one good edge.*

Fact 4.2. *Let vw be an edge in G such that $v \in V_3$ and $w \in V_3'$. If w has a neighbour $u \neq v$ such that edge wu is not a middle edge, then vw is a good edge of G.*

Lemma 4.3. *Let T be a biplanarizing set of G. There exists a normal biplanarizing set T' of G such that $|T'| \leq |T|$.*

Proof. To prove this lemma we first obtain a biplanarizing set T'' that contains no leaf-edges of G and has $|T''| \leq |T|$. Suppose T contains a leaf edge vw, where w is its leaf. Let F denote $G \setminus T$. Then F is a forest of caterpillars that contains a component $F_w = \{w\}$ and a component F_v such that $v \in F_v$. To prove that T'' exists, it suffices to show that if $F' = F_v \cup F_w \cup \{vw\}$ is not a caterpillar, then there exists a non-leaf edge e such that $F' \setminus \{e\}$ is a forest of caterpillars. Since w is isolated in $F_v \cup F_w$, therefore F' is a tree. Now suppose there is a 2-claw C in F'. Since w is a leaf, neither v nor w may be centers of any 2-claw in F'. Let x be the center of C. Since vw is contained in all 2-claws in F', it must be a secondary edge of C. Since F_v is a caterpillar and thus has no 2-claws, it follows that v has no neighbours other than x and w in F'. Therefore $F' \setminus \{xv\}$ is biplanar since it consists of two components, one component is vw and the other component is a subgraph of F_v. This shows that T'' exists since xv is not a leaf-edge in G.

Now let $vw \in T''$ be a bad edge that belongs to a component sun W. If $(G \setminus T'') \cup \{vw\}$ is not a forest of caterpillars then $(G \setminus T'') \cup \{vw\}$ consists of W and a forest of caterpillars. By Fact 4.1, the sun-cycle of W contains a good edge e; thus $((G \setminus T'') \cup \{vw\}) \setminus \{e\}$ is biplanar. Repeating this for every edge of T'' that is bad and belongs to some component sun gives the normal biplanarizing set T' that has $|T'| \leq |T|$. □

There are two possible types of bad edges that may appear in a normal biplanarizing set:

type-1: edges vw such that neither v nor w is black and vw is not a middle edge,

type-2: edges vw such that v is black and w is gray and vw belongs to some internal 3-path or pendant triangle in G.

Lemmas 4.4 and 4.5 below prove that every bad edge of type-1 and type-2 respectively, can be replaced by a good edge in a normal biplanarizing set.

Lemma 4.4. *Let T be a normal biplanarizing set of G. Let $vw \in T$ be a bad edge such that $v, w \notin V_3$. Then either $T \setminus \{vw\}$ is a biplanarizing set for G, or there exists a good edge $e \in E$ such that $T' = (T \setminus \{vw\}) \cup \{e\}$ is a biplanarizing set for G.*

Proof. Let F denote $G \setminus T$. Then F is a forest of caterpillars. Let $F_1, F_2, \ldots, F_s$ be the component caterpillars of F. Every vertex of G belongs to exactly one component of F. Since T contains no leaf-edge of G, every leaf-edge of G belongs to exactly one component of F. We prove the lemma by considering two possible scenarios, depending on whether v and w belong to the same component caterpillar of F or not.

Case 1: $v \in F_i$ and $w \in F_j$ where $i \neq j$.

$F \cup \{vw\}$ is a forest comprised of $s-2$ component caterpillars and a tree $F_{ij} = F_i \cup F_j \cup \{vw\}$. Suppose now that there is a 2-claw in F_{ij}. Neither v nor w is black in G; thus neither vertex is the center of any 2-claw in F_{ij}. Thus at least one of $F_i \cup \{vw\}$ and $F_j \cup \{vw\}$ contains a 2-claw.

Assume first that both $F_i \cup \{vw\}$ and $F_j \cup \{vw\}$ contain a 2-claw. Let x be a center of a 2-claw in $F_i \cup \{vw\}$; and let y be a center of a 2-claw in $F_j \cup \{vw\}$, as illustrated in Figure 4.5(a). Both of these 2-claws must contain vw, so there is an edge $xv \in F_i \cup \{vw\}$ and an edge $yw \in F_j \cup \{vw\}$. Thus there is path $P = \{x, v, w, y\}$ in G that is potentially an internal 3-path in G with middle edge vw. By our assumption, vw is not a middle edge. Hence at least one of v or w, say v, has a neighbour v' in G that is not in P. That neighbour v' is a leaf in G, since by the assumption, deg'$(v) = 2$ in both G and F_{ij}. Since vv' is a leaf-edge in G and since T has no leaf-edges then vv' is also in F_i. However that implies that there is a 2-claw in F_i, which is a contradiction.

Assume now, without loss of generality, that only $F_i \cup \{vw\}$ contains a 2-claw, and that $F_j \cup \{vw\}$ is a caterpillar. Let x be a center of a 2-claw in $F_i \cup \{vw\}$, as illustrated in Figure 4.5(b). This 2-claw must contain edge vw, so there is an edge $xv \in F_i \cup \{vw\}$. Vertex v can have no other neighbours in F_i as otherwise there would be 2-claw in F_i. Thus $F_{ij} \setminus \{xv\}$ consists of two components $F_i \setminus \{xv\}$ and $F_j \cup \{vw\}$ that are both biplanar. Therefore, $F_{ij} \setminus \{xv\}$ is biplanar as well. This completes the proof for this case since by assumption vw is not a middle edge and thus xv must be a good edge.

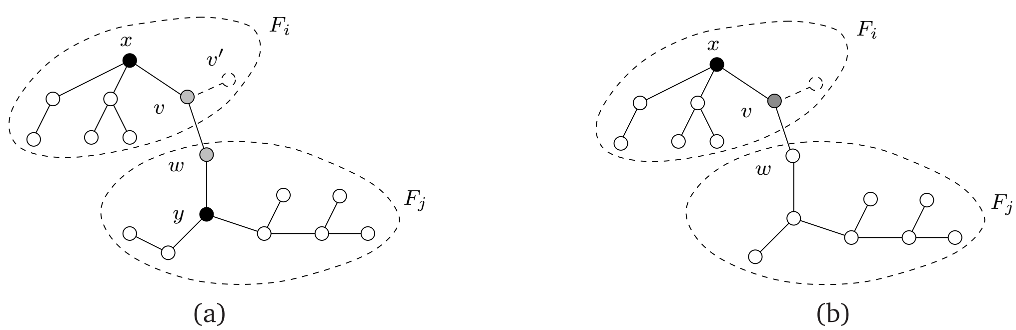


FIGURE 4.5: Illustration for the proof of Lemma 4.4 — Case 1: (a) $F_i \cup \{vw\}$ and $F_j \cup \{vw\}$ both contain a 2-claw, (b) $F_i \cup \{vw\}$ contains a 2-claw, $F_j \cup \{vw\}$ is biplanar.

Case 2: $v, w \in F_i$.

$F \cup \{vw\}$ is comprised of $s-1$ component caterpillars and a component $F_i' = F_i \cup \{vw\}$ that contains exactly one cycle. That cycle contains edge vw. Let $P = (v_1, v_2, \ldots, v_p)$ denote the spine of caterpillar F_i. Since $v, w \notin V_3$, neither v nor w can be equal to any vertex $v_2, v_3, \ldots v_{p-1}$. Having that in mind together with the assumption that vw is not a middle edge in G, it is simple to verify that either F_i' is a sun, or nearly-sun. *Nearly-sun* is comprised of a vertex x, one pendant sun and one pendant caterpillar both connected at x and possibly some leaves at x, as illustrated in Fig. 4.6(b).

If F_i' is a sun then by Fact 4.1 there is a good cycle edge $e \in F_i'$. Clearly $F_i' \setminus \{e\}$ is a

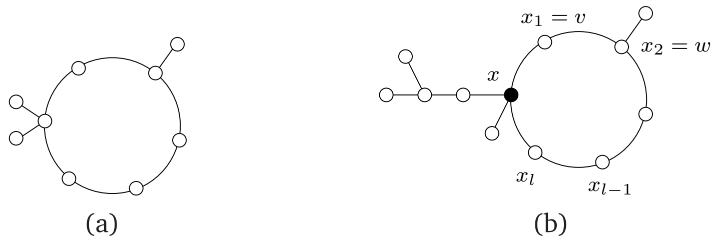


FIGURE 4.6: Illustration for the proof of Lemma 4.4 — Case 2 : (a) sun, (b) nearly-sun.

caterpillar. Otherwise let F_i' be a nearly-sun, and let $x, x_1, x_2, \ldots, x_{l-1}, x_l$ denote the vertices on the pendant sun cycle. Since every 2-claw and the cycle in F_i' must contain edge vw and since neither v nor w is black, it follows that vw is either x_1x_2 or x_{l-1}, x_l (notice that x_1x_2 and x_{l-1}, x_l may in fact be the same edge). Let, without loss of generality, vw be x_1x_2. Since F_i' is a nearly-sun, it is easy to see that $F_i' \setminus \{xx_1\}$ is a caterpillar. This completes the proof for this case since vw is not a middle edge and thus by Fact 4.2 xx_1 is a good edge. □

Lemma 4.5. *Let T be a normal biplanarizing set of G. Let $vw \in T$ be a bad edge such that $v \in V_3$ and $w \in V_3'$. Then either $T \setminus \{vw\}$ is a biplanarizing set of G, or there exists a good edge $e \in E$ such that $T' = (T \setminus \{vw\}) \cup \{e\}$ is a biplanarizing set of G.*

Proof. Suppose that T contains a bad edge vw such that $v \in V_3$ and $w \in V_3'$. Then vw belongs to either an internal 3-path or a pendant triangle I in G. Let $wx \in \mathcal{K}$ be the middle edge of I. Let F denote $(G \setminus T) \cup \{vw\}$. To prove the lemma it suffices to show that if F is not biplanar then there exists a good edge $e \in F$ such that $F \setminus \{e\}$ is a biplanar.

If F is not biplanar then every cycle and 2-claw in F must contain vw. Since w has degree two, every cycle in F also contains wx. Thus $F \setminus \{wx\}$ is a forest. If $F \setminus \{wx\}$ is biplanar we are done; otherwise, there is at least one 2-claw, C, in $F \setminus \{wx\}$. Since w is a leaf in $F \setminus \{wx\}$, vw must be a secondary edge of C. Thus C is centered at some neighbour $u \neq w$ of v, and therefore $u \in V_3$ and $uv \in \mathcal{K}$. Thus $\deg_F(v) = 2$ (as otherwise $G \setminus T$ has a 2-claw), and therefore in F, vertices v, w and x have degree at most two. Thus every cycle in F contains uv. Hence $F \setminus \{uv\}$ is a forest. Furthermore, $\deg_{F \setminus \{uv\}}(v) = 1$ and all the vertices in the distance two neighbourhood of v in $F \setminus \{uv\}$ have degree in F at most two, and thus $F \setminus \{uv\}$ is biplanar, which completes the proof since $uv \in \mathcal{K}$. □

Proof of Theorem 4.1. Let T be a biplanarizing set for G. By Lemma 4.3 there is a normal biplanarizing set T' for G such that $|T'| \leq |T|$. Lemmas 4.4 and 4.5 imply that T' can be transformed into biplanarizing set T^* for G such that all the edges in T^* are good, that is $T^* \in \mathcal{K}$, and $|T^*| \leq |T|$. □

Guided by Theorem 4.1, we now identify a set of edges $\mathcal{S}_G$ of G that may be assumed without loss of generality to be in a minimum biplanarizing set. More precisely, there exists a minimum biplanarizing set that contains $\mathcal{S}_G$. For each vertex $v \in V_3$, let $\alpha(v)$ be the number of pendant caterpillars connected at v in G. We define $\mathcal{S}_G$ to be the set of edges $\mathcal{S}_G \subset E$ that contains:

- the good edge from each component sun,
- the good edge (that is, middle edge) from each pendant triangle,
- one of the two good edges from each pendant sun,
- for each vertex v, the good edges from $\max\{\alpha(v)-2, 0\}$ pendant caterpillars connected at v.

Lemma 4.6. *There exists a minimum biplanarizing set T of G that contains $\mathcal{S}_G$*

Proof. Exactly one edge of each component sun and pendant triangle is in $\mathcal{K}$ and that is their good edge. Since they contain a cycle, each component sun and pendant triangle needs to have one of their edges in any biplanarizing set. Thus by Theorem 4.1, the lemma is true for component suns and pendant triangles.

Now consider the third bullet and let W be a (non-triangle) pendant sun connected at v_1. Only two edges of W, v_1v_2 and v_1v_p, are in $\mathcal{K}$. Thus there exists minimum biplanarizing set T that contains at least one of them. Say T contains v_1v_p and we placed $v_1v_2 \in \mathcal{S}_G$. If T also contains v_1v_2 we are done. Otherwise the subgraph of W that is in $G \setminus T$, is comprised of a pendant caterpillar at v_1. The subgraph of W that is in $G \setminus ((T \cup \{v_1v_2\}) \setminus \{v_1v_p\})$ is also comprised of one pendant caterpillar at v_1. Thus $(T \cup \{v_1v_2\}) \setminus \{v_1v_p\}$ is a biplanarizing set of G.

Now consider the last bullet and let Q be a subgraph of G containing v and $\alpha(v) \geq 3$ pendant caterpillars at v. Exactly one edge from each pendant caterpillar at v is in $\mathcal{K}$. At least $\alpha(v) - 2$ of these edges has to be in any biplanarizing set of G, as otherwise G has a 2-claw. Let T be a minimum biplanarizing set of G containing $\alpha(v) - 2 \leq p \leq \alpha(v)$ good edges of Q. Then the subgraph of Q that is in $G \setminus T$, is comprised of $\alpha(v) - p$ pendant caterpillars at v. Which $\alpha(v) - p$ pendant caterpillars at v does $G \setminus T$ contain is irrelevant, thus we can chose arbitrary $\alpha(v) - 2$ good edges of Q to put in $\mathcal{S}_G$. ☐

4.2.2 Size of the candidate set

The next lemma demonstrates that the size of the candidate set of graph G is tied to biplanarization number of G. Let d be the average non-leaf degree of vertices in V_3.

Lemma 4.7. $\mathsf{bpr}(G) \leq |\mathcal{K}| \leq 2\,\frac{d}{d-2}\mathsf{bpr}(G) \leq 6\,\mathsf{bpr}(G)$.

Before proving this lemma we show that Theorem 4.1 and Lemma 4.7 give an FPT algorithm for 2-LAYER PLANARIZATION as demonstrated in the next corollary.

Corollary 4.1. *Let $k' = k - |\mathcal{S}_G|$ and $\mathbf{e}$ be the base of the natural logarithm. There is a $\mathcal{O}((\frac{2\mathbf{e}\,d}{d-2})^{k'}|G|) \in \mathcal{O}(17^{k'}|G|)$ algorithm for the* 2-LAYER PLANARIZATION *problem.*

Proof. Consider the following algorithm. Remove all edges in $\mathcal{S}_G$ from G to obtain graph $G' = G \setminus \mathcal{S}_G$. Compute the candidate set $\mathcal{K}'$ of G'. If $k' < \frac{d-2}{2d}|\mathcal{K}'|$ return NO. Else if $k' \geq |\mathcal{K}'|$ then return YES. Else for every subset $S \subseteq \mathcal{K}'$ such that $|S| = k'$, test if S is a biplanarizing set for G'. If no such set S is found return NO, otherwise return YES and biplanarizing set $T = \mathcal{S}_G \cup S$. The correctness of this algorithm follows from Theorem 4.1 and Lemmas 4.6 and 4.7.

Consider now the running time of this algorithm. Computing $\mathcal{S}_G$ for G and $\mathcal{K}'$ for G' takes $\mathcal{O}(|G|)$ time. Testing whether some set S is biplanarizing set for G' takes $\mathcal{O}(|G'|)$ time. The algorithm performs that test at most $\binom{|\mathcal{K}'|}{k'}$ times. Since the testing is performed only if $|\mathcal{K}'| \leq 2\,\frac{d}{d-2}k'$, $\binom{|\mathcal{K}'|}{k'} \leq \binom{2\,\frac{d}{d-2}k'}{k'} < \frac{(2k'\,d/(d-2))^{k'}}{k'!} < \frac{(2\mathbf{e}\,d/(d-2))^{k'}}{\sqrt{k'}}$ by Stirling's Formula. Since $d \geq 3$, in the worst case the running time of the algorithm is $\mathcal{O}((6\mathbf{e})^{k'}\,|G|) \in \mathcal{O}(17^{k'}\,|G|)$. $\square$

To enable us to prove Lemma 4.7, we introduce the following potential function, whose definition is suggested by Lemma 4.1(d). For a graph $G = (V, E)$, define

$$\forall v \in V,\ \Phi_G(v) = \max\{\text{deg'}_G(v) - 2, 0\}, \ \text{ and } \ \Phi(G) = \sum_{v \in V} \Phi(v) \ .$$

Intuitively, $\Phi(v)$ approximates the number of edges in the distance-2 neighbourhood of v that must be included in a biplanarizing set of G.

Lemma 4.8. $\Phi(G) = 0$ *if and only if G is a collection of caterpillars and suns.*

Proof. Since neither caterpillars nor suns have vertices with non-leaf degree greater than two, their potential function is clearly equal to zero. For the other direction, suppose $\Phi(G) = 0$ and consider a graph G' obtained from G by deleting all its leaves. G' does not have a vertex of degree three or more, so G' is a collection of paths and cycles. Therefore, G is a collection of caterpillars and suns. $\square$

Notice that Lemma 4.8 proves another characterization of biplanar graphs. Namely, G is biplanar if and only if G is acyclic and $\Phi(G) = 0$. For graphs G with $\Phi(G) = 0$, a minimum biplanarizing set of G consists of one cycle edge from each component sun. For graphs with $\Phi(G) > 0$ the following observation will be useful.

Lemma 4.9. *Let G be a graph with $\Phi(G) > 0$ (that is, $V_3 \neq \emptyset$). If d is the average non-leaf degree of vertices in V_3 then $|V_3| = \frac{\Phi(G)}{d-2}$.*

Proof. By definition,

$$d|V_3| = \sum_{v \in V_3} \text{deg'}(v) = \sum_{v \in V_3} (\Phi_G(v) + 2) = \Phi(G) + 2|V_3| \ .$$

Thus, $(d-2)|V_3| = \Phi(G)$, and the result follows. □

We now prove that $\Phi(G)$ provides a lower bound for $\mathsf{bpr}(G)$.

Lemma 4.10. *For every graph G,* $\mathsf{bpr}(G) \geq \frac{1}{2}\Phi(G)$.

Proof. The result follows from Lemma 4.8 if we prove that deleting one edge vw from G with $\Phi(G) > 0$ reduces $\Phi(G)$ by at most two.

If at least one of v and w (say v) is a leaf, then $\Phi(v) = 0$ and $\Phi(w)$ does not change by deleting vw. If w becomes a leaf by deleting vw, then w has one neighbour x for which Φ is reduced by at most one.

If neither v nor w are leaves in G, then there are three possible outcomes when the edge vw is deleted.

Case 1. $\Phi(v)$ and $\Phi(w)$ both decrease: Then before deleting vw, deg'$(v) \geq 3$ and deg'$(w) \geq 3$. Thus, v and w do not become leaves by deleting vw, and Φ does not decrease for any other vertices.

Case 2. Exactly one of $\Phi(v)$ and $\Phi(w)$, say $\Phi(v)$, decreases: Then deg'$(v) \geq 3$ and deg'$(w) \leq 2$ before deleting vw. Thus, w has at most one neighbour x ($\neq v$) such that $\Phi(x)$ decreases. Furthermore $\Phi(x)$ decreases by at most one. Thus for no neighbour of v, except possibly x, is Φ reduced.

Case 3. Neither $\Phi(v)$ nor $\Phi(w)$ decreases: Thus, deg'$(v) \leq 2$ and deg'$(w) \leq 2$ before deleting vw. Each of v and w has at most one neighbour for which Φ may decrease. If these neighbours are distinct, then Φ may decrease by at most one for each neighbour; if they are the same then Φ may decrease by at most two for the common neighbour. □

Proof of Lemma 4.7. By Theorem 4.1 there exists a minimum biplanarizing set T of G such that $T \subseteq \mathcal{K}$. Therefore $|\mathcal{K}| \geq \mathsf{bpr}(G)$. To prove that $|\mathcal{K}| \leq 2\,\frac{d}{d-2}\mathsf{bpr}(G)$ we count the number of edges in $\mathcal{K}$ with respect to the vertices in V_3.

$$|\mathcal{K}| \ \leq \ \sum_{v \in V_3} \text{deg'}_G(v) \ = \ \sum_{v \in V_3} (\Phi_G(v) + 2) \ = \ 2|V_3| + \Phi(G) \ .$$

By Lemma 4.9, $|\mathcal{K}| \leq \Phi(G)(1 + 2/(d-2)) = \Phi(G)\,d/(d-2)$. Since $\Phi(G) \leq 2\mathsf{bpr}(G)$, $|\mathcal{K}| \leq 2\mathsf{bpr}(G)\,d/(d-2)$. Since $d \geq 3$, $|\mathcal{K}| \leq 6\mathsf{bpr}(G)$. □

The graph illustrated in Figure 4.7(a) has biplanarization number one, and its candidate set of six edges is shown in Figure 4.7(b). Thus our analysis for the size of the candidate set is tight.

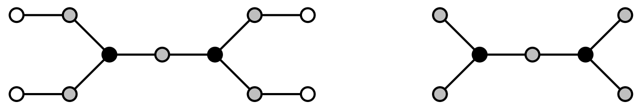


FIGURE 4.7: A graph with biplanarization number one and with a candidate set of 6 edges.

In the example of Figure 4.7, it is not necessary to include the edges contained in the pendant caterpillars in any biplanarizing set. This observation suggests the following methods for further reducing the size of the candidate set.

Observation 4.1. *Let G be a graph with $\mathcal{S}_G$=0. Let $0 \leq \alpha(v) \leq 2$ be the number of pendant caterpillars connected at v in G. For each vertex $v \in V$,*

1. *if $\alpha(v) = 2$ and $\text{deg'}_G(v) = 3$ then none of the edges in these two pendant caterpillars need be in the candidate set $\mathcal{K}$ for G;*

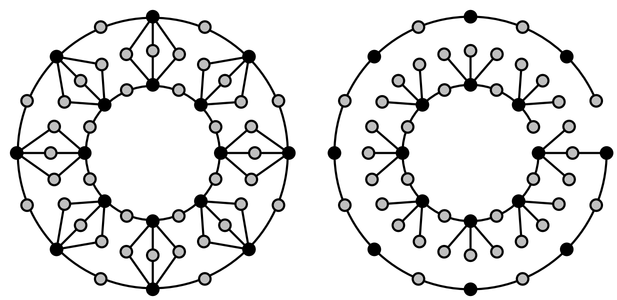
2. *if $\alpha(v) = 1$, then none of the edges in the pendant caterpillar need be in $\mathcal{K}$.*

Proof. Consider the first observation and let $T \subseteq \mathcal{K}$ be a biplanarizing set of G. Let x, y and z be the non-leaf neighbours of v where vx and vy belong to the two pendant caterpillars connected at v. By Theorem 4.1, no edge of the two pendant caterpillars other than vx and vy may belong to T. If z is an endpoint of a middle edge in G, let $w \neq v$ be the neighbour of z and let $T' = (T \setminus \{vx, vy\}) \cup \{zw\}$. Otherwise, let $T' = (T \setminus \{vx, vy\}) \cup \{vz\}$. $G \setminus T'$ is clearly biplanar. Since both vz and zw are good and since neither vz nor zw belong to any pendant caterpillars in G, the correctness of the first observation follows.

Consider now the second observation. Let x denote the non-leaf neighbour of v that belongs to the pendant caterpillar connected at v. Let $T \subseteq \mathcal{K}$ be a minimum biplanarizing set of G. If T contains no edge of the pendant caterpillar we are done; otherwise, by Theorem 4.1, we may assume that T contains vx and no other edge of the pendant caterpillar. $(G \setminus T) \cup \{vx\}$ is acyclic and by minimality of T it contains a 2-claw C. C must contain vx. Thus v has a non-leaf neighbour $z \neq x$. If z is an endpoint of a middle edge in G let $w \neq v$ be the neighbour of z and let $T' = (T \setminus \{vx\}) \cup \{zw\}$. Otherwise, let $T' = (T \setminus \{vx\}) \cup \{vz\}$. $G \setminus T'$ is clearly biplanar. Since both vz and zw are good and since neither vz nor zw belong to any pendant caterpillars in G, the correctness of the second observation follows. $\square$

While in many cases arising in practice the above observations could lead to improved running time for our algorithm, we now describe a pathological family of graphs for which our analysis in Lemma 4.7 for the size of the candidate set is tight even with the above improvements. Consider the graph $G_{p,q}$ ($p, q \in \mathbb{N}$) consisting of an *inner* cycle $(v_1, \ldots, v_{2p})$

and an *outer* cycle $(w_1, \dots, w_{2p})$ with v_{2i} connected by q 2-paths to w_{2i} for all i, $1 \le i \le p$, as illustrated in Figure 4.8(a) in the case of $G_{8,3}$. All vertices in V_3 have non-leaf degree $d = q+2$. $G_{p,q}$ has $(d+2)p$ vertices and $2dp$ edges. It is easily verified that the candidate set of $G_{p,q}$ is the whole graph. As shown in Figure 4.8(b), $G_{p,q}$ has a spanning caterpillar with $p(d+2)-1$ edges. There is no larger biplanar subgraph than a spanning caterpillar. Thus $\mathsf{bpr}(G_{p,q}) = 2dp - (p(d+2)-1) = p(d-2)+1$. The ratio of the number of edges in the candidate set of $G_{p,q}$ to $\mathsf{bpr}(G_{p,q})$ is $\frac{2dp}{p(d-2)+1} \to \frac{2d}{d-2}$ as $p \to \infty$. Thus the analysis of the size of the candidate set in Lemma 4.7 is tight for all d.



(a) (b)

FIGURE 4.8: The graph $G_{8,3}$ and a spanning caterpillar of $G_{8,3}$.

4.2.3 The kernelization algorithm

The algorithm in Corollary 4.1 tests for every subset S of $\mathcal{K}$ such that $|\mathcal{K}| = k$, if S is a biplanarizing set for G, thus giving rise to the running time of the form $\mathcal{O}(f(k)\,|G|)$. In this section we describe how to reduce the input graph G to graph G_{kr} of size $\mathcal{O}(k)$ such that the testing if S is a biplanarizing set for G can be performed on G_{kr}. Consequently, the cost of testing becomes $\mathcal{O}(k)$, thus giving the final running time of the from $\mathcal{O}(k\,f(k) + |G|)$.

By Lemma 4.6, instead of working with a problem instance G' with parameter k' where $\mathcal{S}_{G'} \neq \emptyset$, we can work with $G = G' \setminus \mathcal{S}_{G'}$ and parameter $k = k' - |\mathcal{S}_{G'}|$. Therefore, without loss of generality, we may now assume that the input graph G has $\mathcal{S}_G = \emptyset$, and thus has no component suns.

The graph induced by the white vertices in any graph is comprised of component suns and a forest of caterpillars. Since G has no suns by the assumption, the induced graph is a forest of caterpillars. This motivates the following construction. Let a *kernel graph* $G_{\mathsf{kr}} = (V_{\mathsf{kr}}, E_{\mathsf{kr}})$ be a graph obtained from G by performing the following reduction operations on G in the order in which there are given below.

Reduction operations:

1. For each vertex $v \in V$, replace a set of leaf-edges at v by a single leaf-edge at v.

2. While there is an edge $vw \in E$ with both v and w white in G, contract vw.

3. Delete isolated vertices.

Since the graph induced by the white vertices of G is a forest of caterpillars, the above operations create neither loops nor multiple edges. Therefore, the cycle structure of G is not affected by the reduction operations as there is a bijection between the cycles of G before and after these operations. Similarly, the set of non-leaf edges of G with at least one non-white endpoint, is also preserved by the reduction operations. The only good edges that have both endpoints white are the good edges that belong to component suns. Since G has no component suns by assumption, the candidate set of G and the candidate set of G_{kr} are identical sets.

Lemma 4.11. *Let G be a graph with $\mathcal{S}_G = 0$. Let $\mathcal{K}$ be the candidate set of G and its kernel graph G_{kr}. A set $T \subseteq \mathcal{K}$ is a biplanarizing set of G if and only if T is a biplanarizing set of G_{kr}.*

Proof. Let G' be a graph obtained from G after completing reduction operation 1. Since T contains no leaves, it is simple to verify that T is a biplanarizing set of G if and only if T is a biplanarizing set of G_{kr}. Therefore we need only prove that T is a biplanarizing set of G' if and only if T is a biplanarizing set of G_{kr}.

Since, by the assumption, G' has no component suns, and as previously pointed out, the cycle structure of G' is not affected by the reduction operations. Thus the existence of a cycle in $G \setminus T$ implies that there is a cycle in $G_{\mathsf{kr}} \setminus T$ and equivalently, the existence of a cycle in $G \setminus T$ implies that there is a cycle in $G_{\mathsf{kr}} \setminus T$. Furthermore, since no 2-claw contains an edge with both endpoints white, there is a bijection between the 2-claws in G' and the 2-claws in G_{kr}. Therefore, the existence of a 2-claw in $G \setminus T$ implies that there is a 2-claw in $G_{\mathsf{kr}} \setminus T$ and equivalently, the existence of a 2-claw in $G \setminus T$ implies that there is a 2-claw in $G_{\mathsf{kr}} \setminus T$. ☐

Lemma 4.12. *Let G be a graph with $\mathcal{S}_G = 0$. The kernel graph G_{kr} of G has $|E_{kr}| \leq 20\,\mathsf{bpr}(G)$ and $|G_{\mathsf{kr}}| \in \mathcal{O}(\mathsf{bpr}(G))$.*

Proof. If $|V_3| = 0$, then $\Phi(G) = 0$. In that case, since G has no component suns, G is biplanar and $\mathsf{bpr}(G) = 0$. Furthermore, since all the vertices of G are white, G_{kr} is the empty graph and thus $|G_{\mathsf{kr}}| \leq \mathsf{bpr}(G) = 0$.

Consider now the case that $|V_3| > 0$. We count the edges in G_{kr} with respect to the black vertices, that is, vertices in V_3. Since each gray vertex in G_{kr} has at most two non-leaf neighbours, at least one of which is black, the number of non-leaf edges in G_{kr} is at most $2\sum_{v \in V_3} \deg'_{G_{\mathsf{kr}}}(v)$. Furthermore, since every leaf vertex is white, only black and gray vertices may be incident to leaf-edges. Therefore the total number of leaf-edges in G_{kr} is at most $|V_3| + |V_3'|$. Since the number of vertices in V_3' is at most $\sum_{v \in V_3} \text{deg'}_{G_{\mathsf{kr}}}(v)$, we have

$$|E_{\mathsf{kr}}| \leq |V_3| + \sum_{v \in V_3} 3\text{deg'}_{G_{\mathsf{kr}}}(v) = |V_3| + \sum_{v \in V_3} (3\Phi_{G_{\mathsf{kr}}}(v) + 6) = 3\Phi(G_{\mathsf{kr}}) + 7|V_3| \ .$$

By Lemma 4.9 applied to G_{kr} and since $d \geq 3$, we have $|V_3| \leq \Phi(G_{\mathsf{kr}})$. Therefore $|E_{\mathsf{kr}}| \leq 10\,\Phi(G_{\mathsf{kr}})$. Since by Lemmas 4.10 and 4.11 $\Phi(G_{\mathsf{kr}}) \leq 2\mathsf{bpr}(G_{\mathsf{kr}}) = 2\mathsf{bpr}(G)$, $|E_{\mathsf{kr}}| \leq 20\,\mathsf{bpr}(G)$, and since G_{kr} has no isolated vertices, $|G_{\mathsf{kr}}| \in \mathcal{O}(\mathsf{bpr}(G))$. □

Algorithm 2-Layer Kernelization
input: graph $G = (V, E)$
parameter: non-negative integer k
output: NO if $\mathsf{bpr}(G) > k$; otherwise, YES and a biplanarizing set of G

---

1. compute $\mathcal{S}_G$ and let $k' = k - |\mathcal{S}_G|$
2. compute $\mathcal{K}$ of $G \setminus \mathcal{S}_G$
3. **if** $k' < \frac{d-2}{2d}|\mathcal{K}|$ **then** return NO
4. **else if** $k' \geq |\mathcal{K}|$ **then** return YES and biplanarizing set $\mathcal{K} \cup \mathcal{S}_G$
5. **else** compute the kernel graph $G_{\mathsf{kr}} = (V_{\mathsf{kr}}, E_{\mathsf{kr}})$ of $G \setminus \mathcal{S}_G$ and

   if $\exists\, T \subseteq \mathcal{K}$ such that $|T| = k'$, $G_{\mathsf{kr}} \setminus T$ is acyclic, and $\Phi(G_{\mathsf{kr}} \setminus T) = 0$ return YES and biplanarizing set $\mathcal{S}_G \cup T$

   else return NO

Theorem 4.2. *Given a graph $G = (V, E)$ and integer k, the algorithm* 2-Layer Kernelization *(G, k) determines if $\mathsf{bpr}(G) \leq k$ and if so, returns a biplanarizing set of size at most k. The running time is $\mathcal{O}(\sqrt{k} \cdot (\frac{2\mathrm{e}\,d}{d-2})^k + |G|) \in \mathcal{O}(\sqrt{k} \cdot 17^k + |G|)$, where d is the average non-leaf degree of vertices in V_3, and e is the base of the natural logarithm.*

Proof. The correctness of the algorithm follows from Theorem 4.1, Lemma 4.6, Lemma 4.7 and Lemma 4.11. Consider now the running time of the algorithm.

Computing $\mathcal{S}_G$, $\mathcal{K}$ and G_{kr} takes $\mathcal{O}(|G|)$ time. Testing whether $T \subseteq \mathcal{K}$ is a biplanarizing set of G_{kr} can be carried out in $\mathcal{O}(|G_{\mathsf{kr}}|) \in \mathcal{O}(\mathsf{bpr}(G)) \in \mathcal{O}(k)$ time, [1] by Lemma 4.12. The number of times the algorithm performs this test is at most the number of k'-edge subsets of

[1] $\mathsf{bpr}(G)$ is bounded by $\mathcal{O}(k)$ in step 5, since $\frac{d-2}{2d}|\mathcal{K}| \leq \mathsf{bpr}(G) \leq |\mathcal{K}|$ and $\frac{d-2}{2d}|\mathcal{K}| \leq k \leq |\mathcal{K}|$.

$\mathcal{K}$; and that is $\binom{|\mathcal{K}|}{k'} \leq \binom{2k'\,d/(d-2)}{k'} < \frac{(2k\,d/(d-2))^{k'}}{k'!} < \frac{(2\mathbf{e}\,d/(d-2))^{k'}}{\sqrt{k'}}$, by Stirling's Formula. Thus the total running time of the algorithm is $\mathcal{O}(\sqrt{k'} \cdot (\frac{2\mathbf{e}\,d}{d-2})^{k'} + |G|)$ which is $\mathcal{O}(\sqrt{k} \cdot (\frac{2\mathbf{e}\,d}{d-2})^{k} + |G|)$ as $k' \leq k$. Furthermore, since $d \geq 3$, in the worst case the running time of algorithm 2-Layer Kernelization is $\mathcal{O}(\sqrt{k} \cdot (6\mathbf{e})^k + |G|) \in \mathcal{O}(\sqrt{k} \cdot 17^k + |G|)$. □

4.2.4 The bounded search tree algorithm

In this section we present an algorithm for the 2-LAYER PLANARIZATION problem based on a bounded search tree method. Each node of the search tree corresponds to a subproblem (G', k'), where $G' \subseteq G$ and $k' \leq k$. At each node we find, if possible, a subgraph C that is a 2-claw or a small cycle. Since every biplanarizing set must contain at least one of the edges in C, our algorithm recursively solves $|C|$ subproblems with one of the edges in C deleted from the graph in each subproblem. Recall that Lemma 4.2 provided a sufficient condition for the existence of such a set C.

Algorithm 2-Layer Bounded Search Tree
input: graph $G_0 = (V_0, E_0)$;
parameter: non-negative integer k_0
output: NO if $\mathsf{bpr}(G_0) > k_0$ otherwise, YES.

---

1. compute $\mathcal{K}$ of G_0
2. **if** $k_0 < \frac{d-2}{2d}|\mathcal{K}|$ **then** return NO
3. **else if** $k_0 \geq |\mathcal{K}|$ **then** return YES
4. **else** ($\exists\, v \in V_0$ such that deg'$_{G_0}(v) \geq 3$)

   if $k_0 > 0$

   (a) find a 2-claw, 3-cycle or 4-cycle C in G_0 containing v as described in Lemma 4.2;

   (b) **for each** edge $xy \in C \cap \mathcal{K}$ **do**

   if 2-Layer Bounded Search Tree($G_0 \setminus \{xy\}$, $k_0 - 1$) returns YES **then** return YES.

   return NO.

Note that the algorithm can be easily modified to return a biplanarizing set for YES instances of the 2-LAYER PLANARIZATION problem. We could solve 2-LAYER PLANARIZATION by running 2-Layer Bounded Search Tree (G, k). Instead, we apply 2-Layer Bounded Search Tree to the kernel of G so that the running time at each node of the search tree is $\mathcal{O}(k)$ rather than $\mathcal{O}(|G|)$.

The above description of our algorithm is recursive and we do not explicitly build a search tree. However, as is standard practice when analyzing recursive algorithms, we associated a *recursion tree* [27], also called search tree, with our algorithm.

Theorem 4.3. *Given a graph G and integer k, let G_{kr} be the kernel graph of $G \setminus \mathcal{S}_G$. The algorithm* 2-Layer Bounded Search Tree $(G_{\mathsf{kr}}, k - |\mathcal{S}_G|)$ *determines if* $\mathsf{bpr}(G) \leq k$ *in* $\mathcal{O}(k \cdot 6^k + |G|)$ *time.*

Proof. The correctness of Steps 1, 2, and 3 follows immediately from Theorem 4.1, Lemma 4.7 and Lemma 4.11. The correctness of Step 4 follows from Lemma 4.2 and Theorem 4.1.

In each recursive call k is reduced by one. Thus the height of the search tree is at most k. At each node of the search tree, there are $|C|$ branches. Since $|C| \leq 6$, the search tree has at most 6^k nodes. At any given node of the search tree, the algorithm takes $\mathcal{O}(|G_0|)$ time. Each G_0 is a subgraph of G_{kr}. Since the algorithm immediately terminates if $k_0 < \frac{d-2}{2d}|\mathcal{K}|$ or $k_0 \geq |\mathcal{K}|$, then $\frac{d-2}{2d}|\mathcal{K}| \leq k_0 \leq |\mathcal{K}|$ and thus by Lemma 4.7, $\mathsf{bpr}(G) \in \mathcal{O}(k)$. By Lemma 4.12, that further implies that $\mathcal{O}(|G_{\mathsf{kr}}|) \in \mathcal{O}(k)$. Hence the time taken at each node of the search tree is $\mathcal{O}(k)$. Therefore, the running time of the algorithm is $\mathcal{O}(k \cdot 6^k + |G|)$.

□

We now compare the exponential terms of the time bounds for the 2-Layer Kernelization and 2-Layer Bounded Search Tree algorithms. The exponential term for 2-Layer Kernelization is $(\frac{2\mathrm{e}\,d}{d-2})^k$, while the exponential term for 2-Layer Bounded Search Tree is 6^k. In the worst case, when $d = 3$, the 2-Layer Kernelization term is approximately 17^k, which is considerably more than 6^k. However, for $d \geq 22$, $\frac{2\mathrm{e}\,d}{d-2} < 6$, and the 2-Layer Kernelization algorithm provides an exponential term with a smaller base than the 2-Layer Bounded Search Tree algorithm.

4.3 1-Layer planarization

We now consider the 1-LAYER PLANARIZATION problem defined in Section 4.1.2: given a bipartite graph $G = (A, B; E)$ and a vertex ordering π of A, is $\mathsf{bpr}(G, \pi) \leq k$? If $\mathsf{bpr}(G, \pi) = 0$ we say that G is π-biplanar. The figures in this section show vertices in A as gray and vertices in B as white. We found it elusive to design an algorithm for this problem based on the kernelization method. However we did find an algorithm based on the bounded search tree method.

The following result characterizes π-biplanar graphs.

Lemma 4.13. *A bipartite graph $G = (A, B; E)$ with a fixed vertex ordering π of A is π-biplanar if and only if G is acyclic and the following condition holds.*

For every path (x, v, y) of G with $x, y \in A$, and for every vertex $u \in A$ between x and y in π, the only edge incident to u (if any) is uv. $(\star)$

Proof. $(\Longrightarrow)$ The fact that every biplanar drawing is a forest of caterpillars implies the necessity for G to be acyclic. The necessity of condition $(\star)$ is also easily verified by observing that if $(\star)$ does not hold for some path (x, v, y) and vertex u, then u has a neighbour $w \neq v$. Regardless of the relative positions of w and v in the vertex ordering of B, uw must cross xv or yv, as illustrated in Figure 4.9(a). This observation was also made by Mutzel and Weiskircher [151].

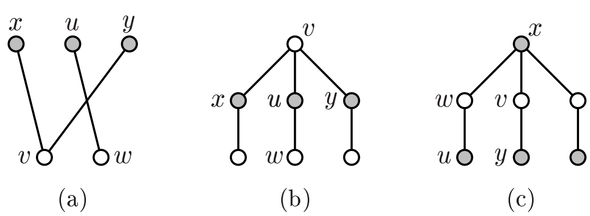


FIGURE 4.9: Forbidden structures for π-biplanarity.

$(\Longleftarrow)$ Suppose G is acyclic and condition $(\star)$ holds. We now prove that these two conditions are sufficient for the π-biplanarity of G. To construct a 2-layer drawing of G, we describe the vertex ordering of B. Let $(1, 2, \ldots, |A|)$ be the vertex ordering of A defined by π. For each vertex $v \in B$, define $l_v = \min\{i : iv \in E\}$; that is, l_v is the leftmost neighbour of v in the fixed vertex ordering of A. We say a vertex $v \in B$ *belongs* to i if $l_v = i$. Order the vertices $v \in B$ by increasing value of l_v, breaking ties as follows. For each i, $1 \leq i \leq |A|$, there is at most one non-leaf vertex belonging to i, as otherwise condition $(\star)$ is violated (see Figure 4.10(a)). Therefore, if i has a non-leaf neighbour, place all the leaf neighbours of i to the left of its non-leaf neighbour. This defines a 2-layer drawing.

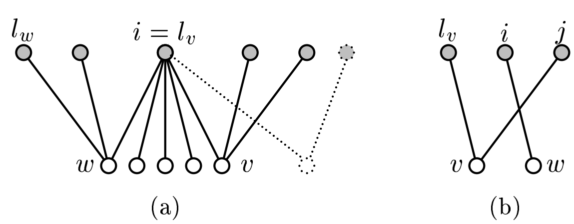


FIGURE 4.10: Construction of the vertex ordering of B.

Suppose there is a crossing between some edges iw and jv with $i, j \in A$ $(i < j)$ and $v, w \in B$. Then v is to the left of w in the vertex ordering of B, and thus $l_v \leq l_w \leq i$. If $l_v < i$ then the condition $(\star)$ is violated for the path (l_v, v, j) and vertex i, as illustrated in Figure 4.10(b). Otherwise, if $l_v = i$ then vertex w cannot be a leaf as otherwise w would be to the left of v. If w is not a leaf, then let l be another neighbour of w. We know that $l \neq j$ as otherwise there would be a cycle in G. Then the condition $(\star)$ is violated either for the path (i, w, l) and vertex j, or for the path (l_v, v, j) and vertex l. Thus there is no crossing in the 2-layer drawing of G. □

Lemma 4.14. *If $G = (A, B; E)$ is a bipartite graph and π is a vertex ordering of A that satisfies condition $(\star)$, then all the cycles of G are 4-cycles and any two non-edge-disjoint cycles share exactly two edges. Moreover, the degree of any vertex in B that appears in a cycle is exactly two.*

Proof. Suppose G contains a cycle C with $2k$ edges with $k \geq 3$. Let $C = (v_1, v_2, \ldots, v_{2k}, v_{2k+1})$ with $v_1 = v_{2k+1} \in A$. Suppose without loss of generality that v_1 is to the left of v_3 in π. If v_5 is between v_1 and v_3 then condition $(\star)$ is not satisfied for the path (v_1, v_2, v_3) and vertex v_5. If v_5 is to the left of v_1, then condition $(\star)$ is not satisfied for the path (v_3, v_4, v_5) and vertex v_1. Thus v_5 is to the right of v_3. Continuing this argument, v_{2i+1} is to the right of v_{2i-1} for all i, $1 \leq i \leq k$. Thus $v_{2k+1}(= v_1)$ is to the right of v_1, which is a contradiction. Thus every cycle in G has four edges.

If G contains two distinct 4-cycles C_1 and C_2 that share exactly one edge vw, then $(C_1 \cup C_2) \setminus \{vw\}$ is a 6-cycle, which is a contradiction. No two distinct 4-cycles in a simple graph can share more than two edges. Thus, any two non-edge-disjoint cycles share exactly two edges.

Let (x, a, y, b) be a 4-cycle of G with x to the left of y in π. Suppose there is an edge aw in G with $x \neq w \neq y$. If w is between x and y in π, then condition $(\star)$ is not satisfied for the path (x, b, y) and vertex w. Otherwise, without loss of generality, say y is between x and w in π. Then condition $(\star)$ is not satisfied for the path (x, a, w) and vertex y. Thus there is no such edge aw. Hence, the degree of all vertices in B that appear in a cycle is exactly two. □

Let $G = (A, B; E)$ be a bipartite graph with a fixed vertex ordering of A that satisfies condition $(\star)$. Let $H = K_{2,p}$ be a complete bipartite subgraph of G with $H \cap A = \{x, y\}$, and $H \cap B = \{v \in B : vx \in E, vy \in E, \deg_G(v) = 2\}$, and $|H \cap B| = p$. Then H is called a *p-diamond* (see Figure 4.11).

It follows from Lemma 4.14 that every cycle of G is in some p-diamond with $p \geq 2$. The next lemma gives the 1-layer biplanarization number $\mathsf{bpr}(G, \pi)$ of G in terms of its p-diamonds, where G is a graph with vertex ordering π satisfying condition $(\star)$.

Lemma 4.15. *If $G = (A, B; E)$ is a bipartite graph and π is a vertex ordering of A satisfying condition $(\star)$ then*

$$\mathsf{bpr}(G, \pi) = \sum_{\textit{maximal } p\textit{-diamonds of } G} (p - 1) \ .$$

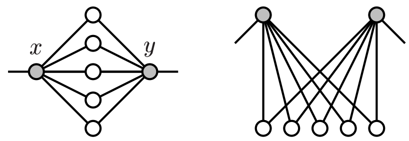


FIGURE 4.11: (a) 5-diamond, (b) 2-layer drawing of a 5-diamond.

Proof. For each maximal p-diamond H of G, delete $p-1$ of the edges incident to one of the vertices in $H \cap A$. The resulting graph is acyclic and satisfies condition ($\star$), and thus, by Lemma 4.13, is π-biplanar. To remove all cycles from G requires the deletion of at least $p-1$ edges from each maximal p-diamond since maximal p-diamonds are edge-disjoint. The result follows. □

We now have the following bounded search tree algorithm for the 1-LAYER PLANARIZATION problem. Our recursive description of the algorithm assumes that a bipartite graph $G = (A, B; E)$ and vertex ordering π of A are given.

Algorithm 1-Layer Bounded Search Tree

input: graph $G_0 = (A_0, B_0, E_0)$; vertex ordering π_0 of A_0

parameter: non-negative integer k_0

output: NO if $\mathsf{bpr}(G_0, \pi_0) > k$ otherwise, YES.

---

1. **if** ($\star$) fails for some path (x, v, y) and vertex u of G_0 **then**

    if $k_0 > 0$

    for each edge $e \in \{xv, yv, uw\}$ **do**

    if 1-Layer Bounded Search Tree ($G_0 \setminus \{e\}$,π_0, $k-1$) returns YES **then**

    return YES.

    return NO.

2. **else if** $k \geq \sum_{\text{maximal } p\text{-diamonds of } G_0} (p-1)$; return YES.

3. **else** return NO.

As in Section 4.2.4 we associate the search (recursion) tree with the recursive description of our algorithm.

Theorem 4.4. *Given a bipartite graph $G = (A, B; E)$, a fixed vertex ordering π of A, and integer k, the algorithm* 1-Layer Bounded Search Tree (G, π, k) *determines if* $\mathsf{bpr}(G, \pi) \leq k$ *in* $\mathcal{O}(3^k \cdot |G|)$ *time.*

Proof. The correctness of the algorithm follows from Lemmas 4.13 and 4.15. We now analyze the running time of the algorithm. First we reorder the adjacency lists of vertices in B by π in $\mathcal{O}(|G|)$-time. For each vertex $v \in B$, let $l_v = \min\{i : iv \in E\}$; and $r_v = \max\{i : iv \in E\}$, that is, l_v and r_v are the leftmost and rightmost neighbours of v in the fixed vertex ordering of A. We now check if condition $(\star)$ holds in $\mathcal{O}(|A|)$ time as follows. For every non-leaf vertex $v \in B$ we test if $(\star)$ is satisfied for a 2-path l_v, v, r_v and all the vertices of A in the open interval (l_v, r_v). This procedure stops when a 2-path and a vertex are found that violate condition $(\star)$ or when all non-leaf vertices $v \in B$ are considered. The procedure runs in $\mathcal{O}(|A|)$ time since it stops the first time it encounters two intervals (l_v, r_v) and (l_w, r_w) for $v \neq w$ with non-empty intersection; otherwise all the intervals (l_v, r_v) and (l_w, r_w) for $v \neq w$ have empty intersection. To count the number and size of the diamonds in G takes $\mathcal{O}(|G|)$ time. Thus, the algorithm takes $\mathcal{O}(|G|)$ time at each node of the search tree. Since each node of the search tree has three children, and the height of the tree is at most k, the algorithm runs in $\mathcal{O}(3^k \cdot |G|)$ time. □

4.4 Approximations for 1- and 2-LAYER PLANARIZATION

As a by-product of Lemma 4.7, we immediately have that the 2-LAYER PLANARIZATION problem has a linear-time $\frac{2d}{d-2}$-approximation, where $d \geq 3$ is the average non-leaf degree of vertices in V_3. However, it is easy to do better. The following observation seems to have gone unnoticed in the literature.

Lemma 4.16. *There is a linear-time 2-approximation algorithm for the optimization version of the* 2-LAYER PLANARIZATION *problem.*

Proof. Let $G = (V, E)$ be a connected graph with n vertices and m edges. Let $r = m-(n-1)$. Then $\mathsf{bpr}(G) \geq r$. Consider the following algorithm. Let S be a set of edges of G such that $G \setminus S$ is a spanning tree T. Then $|S| = r$. Apply the linear-time algorithm of Shahrokhi *et al.* [179] to obtain a minimum set of edges $S_T \subseteq E(T)$ such that $T \setminus S_T$ is biplanar. The number of edges deleted from G is $r + |S_T| = r + \mathsf{bpr}(T) \leq r + \mathsf{bpr}(G) \leq 2\,\mathsf{bpr}(G)$. Thus this algorithm is a 2-approximation, and it clearly runs in $\mathcal{O}(n + m)$ time. □

We now show that there is a constant approximation algorithm for the 1-LAYER PLANARIZATION problem.

Lemma 4.17. *There is a polynomial-time 3-approximation algorithm for the optimization version of the* 1-LAYER PLANARIZATION *problem.*

Proof. Consider an instance (G, π) of the 1-LAYER PLANARIZATION problem with a bipartite graph $G = (A, B; E)$ and a fixed permutation π of A. A path (x, v, y) with $x, y \in A$ and an edge uw with $w \neq v$, $u \in A$ and $x < u < y$ is called a *forbidden structure* in (G, π).

Consider the following algorithm. While condition $(\star)$ is violated by some forbidden structure $(x, v, y), uw$ delete all three edges xv, vy and uw. Let S be the set of deleted edges The instance $(G \setminus S,\, \pi)$ satisfies the constraints imposed by Lemma 4.15. Therefore, $(G \setminus S,\, \pi)$ can be solved optimally.

The number of edge-disjoint forbidden structures in the instance (G, π) is at least $\frac{|S|}{3}$. By Lemma 4.13, at least one of the edges from each of the forbidden structures has to be deleted. Therefore, $\mathsf{bpr}(G, \pi) = \frac{|S|}{3} + \mathsf{bpr}(G \setminus R,\, \pi)$, for some $R \subset S$ and $|R| = \frac{|S|}{3}$. The number of edges deleted from G by the algorithm is $|S| + \mathsf{bpr}(G \setminus S,\, \pi)$. Since $G \setminus S$ is a spanning subgraph of $G \setminus R$, $\mathsf{bpr}(G \setminus R,\, \pi) \geq \mathsf{bpr}(G \setminus S,\, \pi)$. Therefore $|S| + \mathsf{bpr}(G \setminus S,\, \pi) \leq 3\left(\frac{|S|}{3} + \mathsf{bpr}(G \setminus R,\, \pi)\right) = 3\,\mathsf{bpr}(G, \pi)$ and thus the algorithm is a 3-approximation. A running time analysis, similar to the one presented in the proof of Theorem 4.4, reveals that the algorithm can be implemented to run in $\mathcal{O}(|A||B|^2)$ time. □

4.5 Conclusion and bibliographic notes

In this chapter we have presented two methods for producing FPT algorithms in the context of 2-layer and 1-layer planarization. In particular, for fixed k, we have linear-time algorithms to determine if $\mathsf{bpr}(G) \leq k$ and $\mathsf{bpr}(G, \pi) \leq k$. For small values of k our algorithms provide a feasible method for the solution of these $\mathcal{NP}$-complete problems.

The results in this chapter suggest the following open problems.

Open Problem 4.1. Is there a c-approximation algorithm for the optimization version of the 2-LAYER PLANARIZATION problem with $c < 2$? Is there an FPT algorithm for the 2-LAYER PLANARIZATION problem parameterized by the number of edge deletions k, with the exponential part of the running time better than 6^k?

Open Problem 4.2. Is there a c-approximation algorithm for the optimization version of the 1-LAYER PLANARIZATION problem with $c < 3$? Is there an FPT algorithm for the 1-LAYER PLANARIZATION problem parameterized by the number of edge deletions k, with the exponential part of the running time better than 3^k? Is there a problem kernel of size $f(k)$ for the problem?

Notice that the exact values for $\mathsf{bpr}(G)$ or $\mathsf{bpr}(G, \pi)$ can be determined by running our algorithms for each $k = 0, 1, 2 \ldots$, until the first value of k is reached that returns "YES". Clearly that value is equal to $\mathsf{bpr}(G)$ (or $\mathsf{bpr}(G, \pi)$). Therefore, our algorithms can be used to compute optimal solutions for 1- and 2-LAYER PLANARIZATION in time $\mathcal{O}(3^{\mathsf{bpr}(G, \pi)} \cdot |G|)$

and $\mathcal{O}(\mathsf{bpr}(G)(6^{\mathsf{bpr}(G)} + |G|))$, respectively. An initial experimental study, carried out by Suderman and Whitesides [182], compares these algorithms with the other known method for optimal 1- and 2-layer planarization, namely integer linear programming [149, 151]. The results of this study suggest that the FPT method is competitive with the ILP method

Note that there can be many ways of formulating a parameterized version of an optimization problem. For example, in the case of 2-layer planarization, consider the problem of determining whether a given graph $G = (V, E)$ has a spanning forest with at most ℓ component caterpillars (the SPANNING CATERPILLAR FOREST problem). G has a biplanarizing set with k edges if and only if G has a spanning forest with $\ell = k - (|E| - |V|)$ component caterpillars. Thus, from a traditional complexity point of view, the SPANNING CATERPILLAR FOREST problem is equivalent to the 2-LAYER PLANARIZATION problem. In particular, both are $\mathcal{NP}$-complete. In fact, the SPANNING CATERPILLAR FOREST problem with $\ell = 1$ is $\mathcal{NP}$-complete by a simple reduction from HAMILTONIAN PATH. Therefore, unless $\mathcal{P}$=$\mathcal{NP}$, SPANNING CATERPILLAR FOREST is not in $\mathcal{FPT}$, as a polynomial time algorithm for $\ell = 1$ would imply $\mathcal{P}$=$\mathcal{NP}$. Thus, from the perspective of parameterized complexity, unless $\mathcal{P}$=$\mathcal{NP}$, the complexities of the SPANNING CATERPILLAR FOREST and the 2-LAYER PLANARIZATION problems are different. In this sense, parameterized complexity provides a more fine-grained classification of the complexity of problems compared to the traditional complexity approach.

The results of this chapter have appeared in [50].

Chapter 5

Basics of Track Layouts

In this chapter, we first study various fundamental questions regarding track layouts in Section 5.1. In particular, we consider how to colour the edges in a track assignment so that no monochromatic edges form an X-crossing. Furthermore, we prove that every n-vertex (k,t)-track graph has at most $k\big(n(t-1) - k\binom{t}{2}\big)$ edges, and that for every $k \geq 1$, $t \geq 2$ and $n \geq kt$, there is a graph with exactly that many edges. This result provides a lower bound for the track number of every graph. In Section 5.2, several results describing how to convert one type of layout of a graph G into another type of layout of G are presented. These manipulations are critical for a number of results in Chapters 6 and 7. In Section 5.3 we explore the relationship between track layouts and a graph parameter called *geometric thickness*. Specifically, we show that geometric thickness is bounded by track-number and queue-number. In Section 5.4, we consider stack, queue, mixed and track layouts of trees.

5.1 Basics

5.1.1 Fixed track assignment

Consider how to colour the edges in a track assignment so that no monochromatic edges form an X-crossing. The corresponding problems of assigning the edges of a graph G to the minimum number of stacks or queues given a fixed vertex ordering of G, have been studied in the literature. Details can be found in Section 2.2.1.

A track assignment with k pairwise X-crossing edges needs at least k edge colours to be a track layout. The following converse result is a generalization of the fact that permutation graphs are perfect.

Lemma 5.1. *A t-track assignment $\{V_i : 1 \leq i \leq t\}$ of a graph G can be extended into a (k,t)-track layout, where k is the maximum number of pairwise X-crossing edges.*

Proof. Clearly we can consider each pair of tracks V_i and V_j separately. Consider the vertex ordering $\sigma = (V_i, V_j)$ of $G[V_i, V_j]$. Two edges in $G[V_i, V_j]$ form an X-crossing in the track assignment if and only if they are nested in σ. Thus there at most k edges in a rainbow in σ. By Lemmas 2.2 and 5.10, there is an edge k-colouring of $G[V_i, V_j]$ with no monochromatic X-crossing. □

5.1.2 An extremal question

We now consider the maximum number of edges in a track layout. The corresponding problems for stack and queue layouts of graphs have been investigated in the literature. Every s-stack n-vertex graph has at most $(s+1)n-3s$ edges. This bound is tight for all even $n \geq 4$ and all $1 \leq s \leq \frac{n}{2}$. These bounds are due to Bernhart and Kainen [7] and Cottafava and D'Antona [29]. Pemmaraju [160] proved that every q-queue graph with n vertices has at most $2qn - q(2q+1)$ edges. That this bound is tight for all values of n and q has been demonstrated in [55].

It follows from Lemma 2.4 that an n-vertex 2-track graph has at most $n-1$ edges, which generalizes to $(k,2)$-track graphs as follows.

Lemma 5.2. *Let $\{A, B\}$ be a $(k,2)$-track layout of a graph G. Then G has at most $k(|A| + |B| - k)$ edges. Moreover, for all $k \geq 1$ and $n_1, n_2 \geq k$, there exists a $(k,2)$-track layout with $k(n_1+n_2-k)$ edges, and with n_1 vertices in the first track and n_2 vertices in the second track.*

Proof. First we prove the upper bound. Suppose $A = (v_1, v_2, \dots, v_{|A|})$, and $B = (w_1, w_2, \dots, w_{|B|})$. For each edge v_iw_j, let $\lambda(v_iw_j) = i + j$. Observe that $2 \leq \lambda(e) \leq |A| + |B|$ for each edge e. If distinct edges e and f have $\lambda(e) = \lambda(f)$ then e and f form an X-crossing. Thus at most k edges have the same λ value. Moreover, for all $1 \leq i \leq k-1$, at most i edges e have $\lambda(e) = i+1$, and at most i edges e have $\lambda(e) = |A| + |B| + 1 - i$. Thus the number of edges is at most

$$2\sum_{i=1}^{k-1} i \;+\; \big(|A| + |B| - 1 - 2(k-1)\big)k \;=\; k\big(|A| + |B| - k\big) \;.$$

Now we prove the lower bound. Let $A = (v_1, v_2, \dots, v_{n_1})$ and $B = (w_1, w_2, \dots, w_{n_2})$. Construct a graph G with $V(G) = A \cup B$. For each $1 \leq \ell \leq k$, let E_ℓ be the set of edges

$$\big\{v_\ell, w_j : 1 \leq j \leq n_2 + 1 - \ell\big\} \bigcup \big\{v_i, w_{n_2+1-\ell} : \ell + 1 \leq i \leq n_1\big\} \;.$$

Observe that $E_{\ell_1} \cap E_{\ell_2} = \emptyset$ for distinct ℓ_1 and ℓ_2. Let $E(G) = \bigcup_\ell E_\ell$. Clearly no two edges in each E_ℓ form an X-crossing (see Figure 5.1). Thus G has a $(k,2)$-track layout. The number of edges is

$$\sum_{\ell=1}^{k}\big((n_2+1-\ell)+(n_1-\ell)\big) \;=\; k(n_1+n_2)-\sum_{\ell=1}^{k}(2\ell-1) \;=\; k(n_1+n_2-k) \;\;.$$

□

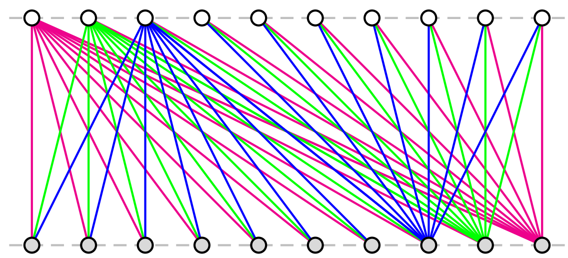


FIGURE 5.1: An edge-maximal $(3,2)$-track layout.

Lemma 5.2 generalizes to (k,t)-track layouts as follows.

Lemma 5.3. *Every n-vertex (k,t)-track graph has at most $k\big(n(t-1)-k\binom{t}{2}\big)$ edges, and for every $k\geq 1$, $t\geq 2$ and $n\geq kt$ there exists a (k,t)-track graph with n vertices and $k\big(n(t-1)-k\binom{t}{2}\big)$ edges.*

Proof. First we prove the upper bound. Let n_i denote the number of vertices in the i^{th} track. By Lemma 5.2, the number of edges between the i^{th} and j^{th} tracks is at most $k(n_i+n_j-k)$. Hence the total number of edges is at most

$$\sum_{1\leq i<j\leq t} k(n_i+n_j-k) \;=\; k\Big(\sum_{1\leq i<j\leq t}(n_i+n_j) \;-\; k\tbinom{t}{2}\Big) \;=\; k\Big((t-1)n-k\tbinom{t}{2}\Big) \;\;.$$

Now we prove the lower bound. Given any $k\geq 1$, $t\geq 2$ and $n\geq kt$, arbitrarily partition n into t integers $n=n_1+n_2+\cdots+n_t$ with each $n_i\geq k$. Construct a (k,t)-track layout with n_i vertices in the i^{th} track, and $k(n_i+n_j-k)$ edges between the i^{th} and j^{th} tracks, as in Lemma 5.2. By the above analysis, the total number of edges is $k\big((t-1)n-k\binom{t}{2}\big)$. □

5.1.3 A lower bound on track number

Since $\binom{t}{2}\geq 1$, Lemma 5.3 implies the following lower bound on $\mathsf{tn}_k(G)$.

Corollary 5.1. *For all $k\geq 1$, every graph G with n vertices and $m\geq 1$ edges has $\mathsf{tn}_k(G)\geq \frac{k^2+m}{kn}+1$.* □

5.1.4 Computational complexity

We have seen in the introduction that it is $\mathcal{NP}$-complete to recognizing 2-stack and 1-queue graphs. As indicated in Section 2.2.4 graphs admitting 2-track layouts are forest of caterpillars and such graphs can be recognized in linear time.

Open Problem 5.1. What is the computational complexity of recognizing track graphs? Is it $\mathcal{NP}$-complete to recognize $(2,2)$-track graphs? Is it $\mathcal{NP}$-complete to recognize 3-track graphs?

5.2 Manipulation of layouts

We first prove two results that show how track layouts can be manipulated without introducing an X-crossing. The first result shows that the number of vertices in different tracks of a track layout can be balanced without introducing an X-crossing. The proof is based on an idea due to Pach *et al.* [158] for balancing the size of the colour classes in a colouring.

Lemma 5.4. *If a graph G has a t-track layout, then for every $t' > 0$, G has an $\lfloor t + t' \rfloor$-track layout with at most $\lceil \frac{n}{t'} \rceil$ vertices in each track.*

Proof. For each track with $q > \lceil \frac{n}{t'} \rceil$ vertices, replace it by $\lceil q / \lceil \frac{n}{t'} \rceil \rceil$ 'sub-tracks' each with exactly $\lceil \frac{n}{t'} \rceil$ vertices except for at most one sub-track with $q \bmod \lceil \frac{n}{t'} \rceil$ vertices, such that the vertices in each sub-track are consecutive in the original track, and the original order is maintained. There is no X-crossing between sub-tracks from the same original track as there are no edges between such sub-tracks. There is no X-crossing between sub-tracks from different original tracks as otherwise there would be an X-crossing in the original. There are at most $\lfloor t' \rfloor$ tracks with $\lceil \frac{n}{t'} \rceil$ vertices. Since there are at most t tracks with less than $\lceil \frac{n}{t'} \rceil$ vertices, one for each of the original tracks, there is a total of at most $\lfloor t + t' \rfloor$ tracks. □

5.2.1 The wrapping lemma

The following lemma describes how to 'wrap' a track layout. The proof is a generalization of the 'wrapping' algorithm of Felsner *et al.* [80], who implicitly proved the case when the (non-partial) span is one.

Lemma 5.5. *Let $\{V_{i,j} : i \geq 0, 1 \leq j \leq b_i\}$ be a (k,t)-track layout of a graph G with maximum partial span s (for some irrelevant value t). For each $0 \leq \alpha \leq s$, let $t_\alpha = \max\{b_i : i \equiv \alpha \pmod{s+1}\}$. For each $0 \leq \alpha \leq 2s$, let $t'_\alpha = \max\{b_i : i \equiv \alpha \pmod{2s+1}\}$. Then*

$$\text{(a) } \mathsf{tn}_{2k}(G) \leq \sum_{\alpha=0}^{s} t_\alpha \text{ , and } \text{(b) } \mathsf{tn}_{k}(G) \leq \sum_{\alpha=0}^{2s} t'_\alpha \ .$$

Proof. Let $\{E_\ell : 1 \leq \ell \leq k\}$ be the edge colouring in the given track layout. First we prove (a). By adding extra empty tracks where necessary, we can assume that the track layout is indexed $\{V_{i,j} : i \geq 0, 1 \leq j \leq t_\alpha, \alpha = i \bmod (s+1)\}$. For each $0 \leq \alpha \leq s$ and $1 \leq j \leq t_\alpha$, let

$$W_{\alpha,j} = \bigcup\{V_{i,j} : i \equiv \alpha \pmod{s+1},\ i \geq 0\} \ .$$

Order $W_{\alpha,j}$ by

$$(V_{\alpha,j},\ V_{\alpha+(s+1),j},\ V_{\alpha+2(s+1),j},\ \dots) \ .$$

Since every edge $vw \in E(G)$ has partial span at most s, if $v \in W_{\alpha_1,j_1}$ and $w \in W_{\alpha_2,j_2}$ then $\alpha_1 \neq \alpha_2$ or $j_1 \neq j_2$. Hence $\{W_{\alpha,j} : 0 \leq \alpha \leq s, 1 \leq j \leq t_\alpha\}$ is a track assignment of G. For each $1 \leq \ell \leq k$, let

$$\begin{aligned} E'_\ell &= \{vw \in E_\ell \,:\, v \in V_{i_1,j_1} \cap W_{\alpha_1,j_1},\ w \in V_{i_2,j_2} \cap W_{\alpha_2,j_2}, i_1 \leq i_2,\ \alpha_1 \leq \alpha_2\},\ \text{and} \\ E''_\ell &= \{vw \in E_\ell \,:\, v \in V_{i_1,j_1} \cap W_{\alpha_1,j_1},\ w \in V_{i_2,j_2} \cap W_{\alpha_2,j_2},\ i_1 < i_2,\ \alpha_2 < \alpha_1\} \ . \end{aligned}$$

An X-crossing between edges both from some E'_ℓ (or both from some E''_ℓ) implies that the same edges form an X-crossing in the original track layout. Thus $\{E'_\ell, E''_\ell : 1 \leq \ell \leq k\}$ defines an edge $2k$-colouring with no monochromatic X-crossing. Thus we have a $(2k, \sum_{\alpha=0}^{s} t_\alpha)$-track layout of G.

We now prove (b). Again by adding extra empty tracks where necessary, we can assume that the track layout is indexed $\{V_{i,j} : i \geq 0, 1 \leq j \leq t'_\alpha, \alpha = i \bmod (2s+1)\}$. For each $0 \leq \alpha \leq 2s$ and $1 \leq j \leq t'_\alpha$, let

$$W_{\alpha,j} = \bigcup\{V_{i,j} : i \equiv \alpha \pmod{2s+1},\ i \geq 0\} \ .$$

Order $W_{\alpha,j}$ by

$$(V_{\alpha,j},\ V_{\alpha+(2s+1),j},\ V_{\alpha+2(2s+1),j},\ \dots) \ .$$

Clearly $\{W_{\alpha,j} : 0 \leq \alpha \leq 2s, 1 \leq j \leq t'_\alpha\}$ is a track assignment of G. It remains to prove that there is no monochromatic X-crossing, where edge colours are inherited from the given track layout. Notice that each $E_\ell = E'_\ell \cup E''_\ell$. As in part (a), edges in E'_ℓ or in E''_ℓ do not form an X-crossing. In the track layout defined for part (b), edges in E'_ℓ have partial span at most s, and edges in E''_ℓ have partial span at least $s+1$. Thus an edge from E'_ℓ and an edge from E''_ℓ do not form an X-crossing. Hence we have a $(k, \sum_{\alpha=0}^{2s} t'_\alpha)$-track layout of G. □

The full generality of Lemma 5.5 will be used in Section 7.3.6. For other applications the following two special cases will suffice. By Lemma 5.5 with $b_i = b$ for all $i \geq 0$, we have:

Lemma 5.6. *Let $\{V_{i,j} : i \geq 0, 1 \leq j \leq b\}$ be a (k,t)-track layout of a graph G with maximum partial span s. Then* (a) $\mathsf{tn}_{2k}(G) \leq (s+1)b$, *and* (b) $\mathsf{tn}_k(G) \leq (2s+1)b$. □

The next special case is with $b = 1$.

Lemma 5.7. *Let G be a (k,t)-track graph with maximum span s. Then* (a) $\mathsf{tn}_{2k}(G) \leq s+1$*, and* (b) $\mathsf{tn}_k(G) \leq 2s+1$. □

5.2.2 Track layouts into track layouts

We now show how to reduce the number of tracks in a track layout, at the expense of increasing the number of edge colours.

Lemma 5.8. *Let G be a (k,t)-track graph with maximum span s $(\leq t-1)$. For every vertex colouring $\{V_i : 1 \leq i \leq c\}$ of G, there is a $(2sk, c)$-track layout of G with tracks $\{V_i : 1 \leq i \leq c\}$.*

Proof. Let $\{T_j : 1 \leq j \leq t\}$ be a (k,t)-track layout of G with maximum span s and edge colouring $\{E_\ell : 1 \leq \ell \leq k\}$. Given a vertex colouring $\{V_i : 1 \leq i \leq c\}$ of G, order each V_i by $(V_i \cap T_1, V_i \cap T_2, \dots, V_i \cap T_t)$. Thus $\{V_i : 1 \leq i \leq c\}$ is a c-track assignment of G. Now we define an edge $2sk$-colouring. For each ℓ and α such that $1 \leq \ell \leq k$ and $1 \leq |\alpha| \leq s$, let

$$E_{\ell,\alpha} = \{vw \in E_\ell : v \in V_{i_1} \cap T_{j_1},\, w \in V_{i_2} \cap T_{j_2},\, i_1 < i_2,\, j_1 - j_2 = \alpha\} \ .$$

Consider two edges vw and xy in some $E_{\ell,\alpha}$ between a pair of tracks V_{i_1} and V_{i_2}. Without loss of generality $i_1 < i_2$, $v \in V_{i_1} \cap T_{j_1}$, $w \in V_{i_2} \cap T_{j_1+\alpha}$, $x \in V_{i_1} \cap T_{j_2}$, $y \in V_{i_2} \cap T_{j_2+\alpha}$, and $j_1 \leq j_2$. If $j_1 = j_2$ then vw and xy are between the same pair of tracks in the given track layout, and the relative order of the vertices is preserved. Thus if vw and xy form an X-crossing in the c-track assignment then they are coloured differently. If $j_1 < j_2$ then $v <_{i_1} x$ and $w <_{i_2} y$, and the edges do not form an X-crossing. Hence vw and xy do not form a monochromatic X-crossing, and we have a $(2sk, c)$-track layout of G. □

Wood [204] showed how to reduce the number of edge colours in a track layout, at the expense of increasing the number of tracks. Recall the definition of star colouring given in Section 2.1.

Lemma 5.9. [204] *Let G be a (k,t)-track graph in which the underlying vertex t-colouring is a star colouring. Then G has track-number $\mathsf{tn}(G) \leq t(k+1)^{t-1}$.*

Lemmas 5.8 and 5.9 imply:

Corollary 5.2. *Let G be a (k,t)-track graph with maximum span s $(\leq t-1)$. If G has star chromatic number $\chi_{\mathrm{st}}(G) \leq c$ then G has track-number $\mathsf{tn}(G) \leq c(2sk+1)^{c-1}$.* □

5.2.3 Queue layouts into track layouts

The following lemma highlights the fundamental relationship between track layouts, and queue and stack layouts. Its proof follows immediately from the definitions, and is illustrated in Figure 5.2 for $k = 1$.

Lemma 5.10. *Let $\{A, B\}$ be a track assignment of a bipartite graph G. Then the following are equivalent:*

(a) *$\{A, B\}$ admits a $(k, 2)$-track layout of G,*

(b) *the vertex ordering (A, B) admits a k-queue layout of G, and*

(c) *the vertex ordering $(A, \overleftarrow{B})$ admits a k-stack layout of G.* □

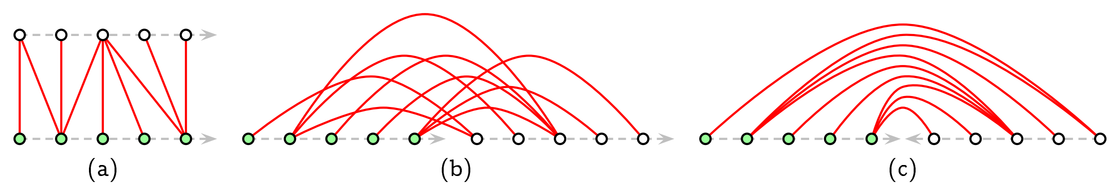


FIGURE 5.2: Layouts of a caterpillar: (a) 2-track, (b) 1-queue, (c) 1-stack.

We now consider how to convert a queue layout into a track layout. The proof of the next result follows immediately from the definitions (see Figure 5.3).

Lemma 5.11. *Let σ be a vertex ordering of a graph G. Let $\{V_i : 1 \leq i \leq c\}$ be a vertex colouring of G. For all $1 \leq i, j \leq c$, a pair of edges $vw, xy \in E(G[V_i, V_j])$ form an X-crossing in the track assignment $\{(V_i, \sigma) : 1 \leq i \leq c\}$ if and only if:*

- *vw and xy are nested in σ* (see Figures 5.3(a)-(b))*, or*
- *vw and xy cross in σ with $v <_\sigma y <_\sigma w <_\sigma x$, and $v, x \in V_i$ and $w, y \in V_j$* (see Figure 5.3(d)). □

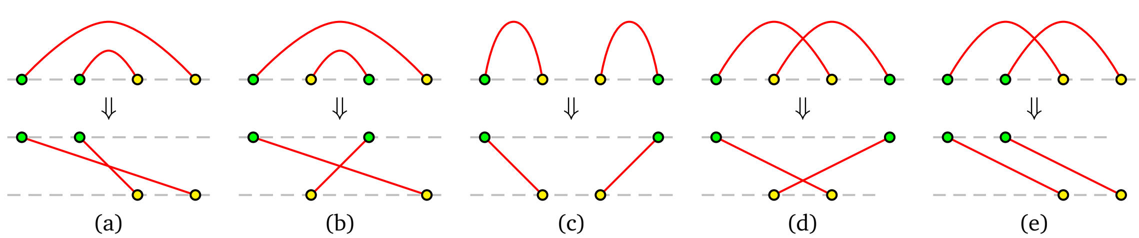


FIGURE 5.3: From a vertex ordering to a track layout: (a)-(b) nested, (c) disjoint, (d)-(e) crossing.

Lemma 5.12. *For every vertex colouring $\{V_i : 1 \leq i \leq c\}$ of a q-queue graph G, there is a $(2q, c)$-track layout of G with tracks $\{V_i : 1 \leq i \leq c\}$.*

Proof. Let σ be the vertex ordering in a q-queue layout of G with queues $\{E_\ell : 1 \leq \ell \leq q\}$. Let $\{(V_i, \sigma) : 1 \leq i \leq c\}$ be a c-track assignment of G, and for each $1 \leq \ell \leq q$, let

$$E'_\ell = \{vw \in E_\ell : v \in V_i, w \in V_j, i < j, v <_\sigma w\}, \text{ and}$$

$$E''_\ell = \{vw \in E_\ell : v \in V_i, w \in V_j, i < j, w <_\sigma v\} \ .$$

By Lemma 5.11, an X-crossing in the track assignment between edges both from some E'_ℓ (or both from some E''_ℓ) implies that these edges are nested in σ. Since no two edges in E_ℓ are nested in σ, the set $\{E'_\ell, E''_\ell : 1 \leq \ell \leq q\}$ defines an edge $2q$-colouring with no monochromatic X-crossing in the track assignment. Thus we have a $(2q, c)$-track layout of G. □

Lemma 5.12 is similar to a result by Pemmaraju [160] which says that a queue layout can be 'separated' by a vertex colouring, although the proof by Pemmaraju, which is based on the characterization of 1-queue graphs as arched levelled planar, is much longer. Pemmaraju used separated queue layouts to prove the next result, which follows from Lemmas 5.10 and 5.12, and implies an affirmative solution to Open Problem 1.1 for bipartite graphs.

Theorem 5.1. [160] *Stack-number is bounded by queue-number for bipartite graphs. In particular, $\mathsf{sn}(G) \leq 2\,\mathsf{qn}(G)$ for every bipartite graph G.* □

Dujmović and Wood [55] proved that every q-queue graph is $4q$-colourable. Thus Lemma 5.12 implies:

Corollary 5.3. *Every q-queue graph has a $(2q, 4q)$-track layout.* □

The next corollary of Lemmas 5.9 and 5.12 slightly improves an analogous result by Wood [204].

Lemma 5.13. *Every q-queue graph G with $\chi_{\mathsf{st}}(G) \leq c$ has track-number $\mathsf{tn}(G) \leq c(2q+1)^{c-1}$.* □

In the case of 1-queue graphs, much improved bounds can be obtained. Di Giacomo and Meijer [40] proved that every 1-queue graph has a 5-track layout. An alternative proof was found by Dujmović and Wood [55].

5.2.4 Track layouts into queue layouts

We now consider how to convert a track layout into a queue layout. First we give a simple proof of a generalization of a result by Wood [204]. Note that Lemma 5.14 with $t = 2$ is nothing more than Lemma 5.10(b).

Lemma 5.14. *Queue-number is bounded by track-number. In particular, every (k,t)-track graph with maximum span s $(\leq t-1)$ has a ks-queue layout.*

Proof. Let $\{V_i : 1 \leq i \leq t\}$ be a (k,t)-track layout of G with maximum span s and edge colouring $\{E_\ell : 1 \leq \ell \leq k\}$. Let σ be the vertex ordering $(V_1, V_2, \dots, V_t)$ of G. Let $E_{\ell,\alpha}$ be the set of edges in E_ℓ with span α in the given track layout. Two edges from the same pair of tracks are nested in σ if and only if they form an X-crossing in the track layout. Since no two edges in E_ℓ form an X-crossing in the track layout, no two edges in E_ℓ and between the same pair of tracks are nested in σ. If two edges not from the same pair of tracks have the same span then they are not nested in σ. (This idea is due to Heath and Rosenberg [114].) Thus no two edges are nested in each $E_{\ell,\alpha}$, and we have a ks-queue layout of G. □

Note that Lemmas 5.8 and 5.10 imply an analogous result to Lemma 5.14 for stack layouts of bipartite graphs.

Lemma 5.15. *Every bipartite (k,t)-track graph with maximum span s $(\leq t-1)$ has a $2ks$-stack layout.* □

Observe that Lemmas 2.1, 5.13 and 5.14 prove the result of Wood [204] about the following strong relationship between queue layouts and track layouts.

Theorem 5.2. [204] *Queue-number and track-number are tied for any proper minor-closed graph family.*

An affirmative solution to the following open problem would imply that queue-number and track-number are tied (for all graphs).

Open Problem 5.2. Is star chromatic number bounded by queue-number?

5.3 Geometric thickness

The *geometric thickness* of a graph G, denoted by $\overline{\theta}(G)$, is the minimum number of colours such that G can be drawn in the plane with edges as coloured straight-line segments, such that monochromatic edges do not cross [42, 123]. Stack-number (when viewed as book-thickness) is equivalent to geometric thickness with the additional requirement that the vertices are in convex position [7]. Thus $\overline{\theta}(G) \leq \mathsf{sn}(G)$ for every graph G. While it is an open problem whether stack number is bounded by track-number or by queue-number, we will prove the following two weaker results that geometric thickness is bounded by track-number (Theorem 5.3), and geometric thickness is bounded by queue-number (Corollary 5.5).

Theorem 5.3. *Geometric thickness is bounded by track-number. In particular, every (k,t)-track graph G has geometric thickness $\overline{\theta}(G) \leq k\lceil \frac{t}{2} \rceil \lfloor \frac{t}{2} \rfloor$.*

Proof. Let $p \geq t$ be a prime. Position the j^{th} vertex in the i^{th} track at $(i, pj + (i^2 \bmod p))$. Wood [206] proved that in this layout no three vertices are collinear, unless all three are in a single track. Since a track is an independent set, the only vertices that an edge intersects are its own endpoints. The vertices in each track are positioned on a line parallel to the Y-axis, in the order defined by the track layout. Thus monochromatic edges between any pair of tracks do not cross. If we let each pair of tracks use a distinct set of k edge colours, then we obtain a drawing of G with $k\binom{t}{2}$ edge colours, such that monochromatic edges do not cross. That is, $\overline{\theta}(G) \leq k\binom{t}{2}$.

This bound can be improved by partitioning the pairs of tracks into sets that can use the same palette of k colours. This amounts to edge-colouring the complete graph K_t with a fixed vertex ordering $(v_1, v_2, \dots, v_t)$, so that overlapping edges receive distinct colours. To this end, define a partial order on $E(K_t)$ as follows. For all edges v_iv_j and v_av_b (with $i < j$ and $a < b$), let $v_iv_j \prec v_av_b$ if $j \leq a$. Clearly $\preceq$ is a partial order on $E(K_t)$, such that distinct edges are overlapping if and only if they are incomparable under $\preceq$. By Dilworth's Theorem [43], there is a partition of $E(K_t)$ into r sets, each pairwise non-overlapping, where r is the largest set of pairwise overlapping edges. Clearly $r = \lceil \frac{t}{2} \rceil \lfloor \frac{t}{2} \rfloor$. For each such set, assign a distinct palette of k colours to the edges between pairs of tracks corresponding to edges of K_t in this set. In total we have kr edge colours, and $\overline{\theta}(G) \leq kr = k\lceil \frac{t}{2} \rceil \lfloor \frac{t}{2} \rfloor$. □

Theorem 5.3 and Lemma 5.12 imply:

Corollary 5.4. *Every q-queue c-colourable graph G has geometric thickness $\overline{\theta}(G) \leq 2q\lceil \frac{c}{2} \rceil \lfloor \frac{c}{2} \rfloor$.* □

Dujmović and Wood [55] showed that every q-queue graph is $4q$-colourable. Thus Lemma 5.12 implies:

Corollary 5.5. *Geometric thickness is bounded by queue-number. In particular, every graph G has geometric thickness $\overline{\theta}(G) \leq 8\,\mathsf{qn}(G)^3$.* □

5.4 Layouts of trees

Let T be a rooted tree. A vertex ordering σ of T is *breadth-first* if for all nodes $x, y \in V(T)$, $x <_\sigma y$ whenever depth$(x) <$ depth(y), or depth$(x) =$ depth(y) and $\rho(x) <_\sigma \rho(y)$.

Lemma 5.16. [114] *A breadth-first vertex ordering of a tree T yields a 1-queue layout of T.*

Proof. Since the depths of adjacent nodes differ by exactly one, and the nodes are ordered by non-decreasing depth, the endpoints of a nested pair of edges must be at consecutive depths. By construction, such a pair of edges are not nested, as illustrated in Figure 5.4. □

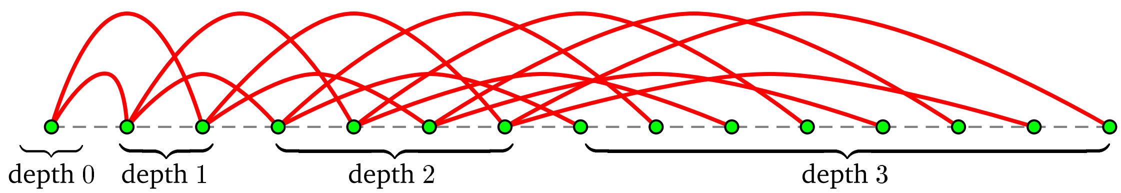


FIGURE 5.4: A 1-queue layout of a complete binary tree.

A *depth-first* vertex ordering σ of a tree T is defined recursively as follows. Let r be the root node of T with child nodes $x_1, x_2, \ldots, x_d$. Let T_i be the subtree rooted at x_i, $1 \leq i \leq d$. Then σ is defined by $\sigma(T) = (r, \sigma(T_1), \sigma(T_2), \ldots, \sigma(T_d))$.

Lemma 5.17. [25] *A depth-first vertex ordering σ of a tree T yields a 1-stack layout of T.*

Proof. For the sake of contradiction, suppose that a pair of edges vw and xy cross in σ. Without loss of generality $v <_\sigma x <_\sigma w <_\sigma y$. Since w is a child of v and $v <_\sigma x <_\sigma w$, we have that x (and y) are in some subtree T_i rooted at a child v_i of v. Since $x <_\sigma w$ we have $v_i \neq w$ and $V(T_i) <_\sigma w$. Since $y \in V(T_i)$, we have $y <_\sigma w$, which is the desired contradiction. Thus no two edges cross in σ, as illustrated in Figure 5.5. $\square$

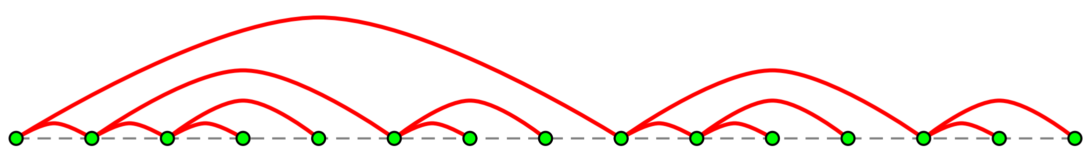


FIGURE 5.5: A 1-stack layout of a complete binary tree.

The next lemma is the starting point for our results on mixed layouts in Section 7.3.5. An edge 2-colouring of a rooted tree T with colours red and black is *good*, if for each node $x \in V(T)$ with an incoming red edge, no other edge incident to x is red. Recall that a vertex ordering of a directed graph is topological if all edges are directed from left to right.

Lemma 5.18. *Let T be a rooted tree with a good edge 2-colouring. Then T has a topological vertex ordering in which the red edges form a stack, and the black edges form a queue.*

Proof. Let h be the height of T. For each $0 \leq d \leq h$, let V_d be the set of nodes of T at depth d. For each $1 \leq d \leq h$, let R_d and B_d denote the sets of nodes in V_d with an incoming red and black edge, respectively. Let σ be the vertex ordering $(V_0, R_1, B_1, R_2, B_2, \ldots, R_h, B_h)$ of T, where for each $1 \leq d \leq h$, the nodes in B_d are ordered with respect to the order of their parents (in V_{d-1}), and the nodes in R_d are in reverse order to that of their parents (in V_{d-1}). More precisely, for all $v, w \in B_d$ we have $v <_\sigma w$ whenever $\rho(v) <_\sigma \rho(w)$, and for all $v, w \in R_d$ we have $v <_\sigma w$ whenever $\rho(w) <_\sigma \rho(v)$. The resulting ordering is clearly topological.

Since the depths of adjacent nodes differ by exactly one, and the nodes are ordered by non-decreasing depth, the endpoints of a nested pair of edges must be at consecutive depths. By construction, such a pair of black edges are not nested. Hence the black edges form a queue.

Suppose, for the sake of contradiction, that the red edges vw and pq cross. Without loss of generality $v <_\sigma p <_\sigma w <_\sigma q$. Then $\text{depth}(v) \leq \text{depth}(p) \leq \text{depth}(w)$. Since $\text{depth}(w) = \text{depth}(v) + 1$, either $\text{depth}(p) = \text{depth}(v)$ or $\text{depth}(p) = \text{depth}(v) + 1$. First suppose that $\text{depth}(p) = \text{depth}(v)$. Then $\text{depth}(q) = \text{depth}(w)$. Since both q and w have incoming red edges, $q <_\sigma w$ by construction. This is a contradiction. Now suppose that $\text{depth}(p) = \text{depth}(v) + 1$. Then $\text{depth}(p) = \text{depth}(w)$. Let $d = \text{depth}(p)$. Since p has an outgoing red edge pq, the incoming edge at p is black, and $p \in B_d$. Now $w \in R_d$ since w has an incoming red edge vw. Since $R_d <_\sigma B_d$, we have $w <_\sigma p$, which is the desired contradiction. Thus no two red edges cross, and hence the red edges form a stack. □

The next result and part (b) of Lemma 5.20 is implicit in the work of Felsner *et al.* [80].

Lemma 5.19. [80] *Every rooted tree T has a (monochromatic) track layout in which every edge has span one.*

Proof. Let σ be a breadth-first vertex ordering of T starting at the root. Let V_d be the set of nodes at depth d. It is easily seen that there are no X-crossings in the track layout $\{(V_d, \sigma) : d \geq 0\}$, and clearly every edge has span one, as illustrated in Figure 5.6. □

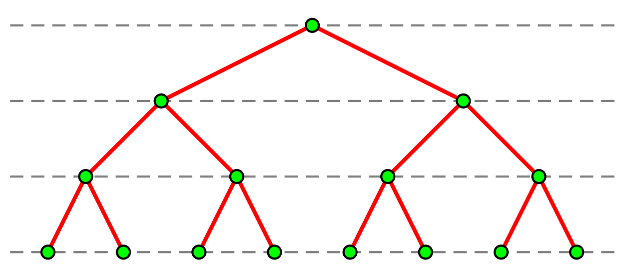


FIGURE 5.6: A track layout of a complete binary tree with every edge having span 1.

Lemmas 5.19 and 5.7 imply the next result.

Lemma 5.20. *Every tree has* (a) *a* $(2, 2)$*-track layout, and* (b) *a* $(1, 3)$*-track layout.* □

5.5 Bibliographic notes

Lemma 5.4 has appeared in [53], together with a much less general form of Lemma 5.5. The results of Sections 5.1, 5.2 and 5.3 are the subject of [57]. The results of Section 5.4 are a part of [56].

Chapter 6

Layouts of Bounded Treewidth Graphs

The main result in this chapter is that the track-number of a graph is bounded by its treewidth (Theorem 6.2), and consequently that the queue-number of a graph is bounded by its treewidth (Corollary 6.2). Treewidth, first defined by Halin [100], although largely unnoticed until independently rediscovered by Robertson and Seymour [169] and Arnborg and Proskurowski [4], is a measure of the similarity of a graph to a tree (see Section 6.1.1 for the definition).

Many graphs arising in applications of graph drawing do have small treewidth. Outerplanar and series-parallel graphs are the obvious examples. Another example arises in software engineering applications. Thorup [187] proved that the control-flow graphs of go-to free programs in many programming languages have treewidth bounded by a small constant; in particular, 3 for Pascal and 6 for C. Other families of graphs having bounded treewidth include: graphs with a feedback vertex set of bounded size, bounded band-width graphs, bounded cut-width graphs, and planar graphs of bounded radius. If the size of a maximum clique is a constant then chordal, interval and circular arc graphs also have bounded treewidth. Thus, by our result, all of these graphs (of bounded treewidth) have $\mathcal{O}(1)$ queue-number, and as we will see in Chapter 8, 3D drawings with $\mathcal{O}(n)$ volume.

Tables 6.1 and 6.2 summarize some of the known bounds on the stack-number, queue-number and track-number of various classes of graphs, including the bounds established in this thesis. A blank entry indicates that a more general result provides the best known bound.

TABLE 6.1: Upper bounds on the stack-number and queue-number.

graph family	stack-number	reference	queue-number	reference
n vertices	$\lceil \frac{n}{2} \rceil$	[25]	$\lfloor \frac{n}{2} \rfloor$	[114]
m edges	$\mathcal{O}(\sqrt{m})$	[141]	$\mathbf{e}\sqrt{m}$	[55]
complete bipartite $K_{n,m}$	$\lceil \frac{2n+m}{4} \rceil$	[146]		
proper minor-closed	bounded	[11]		
genus γ	$\mathcal{O}(\sqrt{\gamma})$	[140]		
treewidth w	w	[138]	$\mathcal{O}(6^{4^w})$	Theorem 6.2
treewidth w, max. degree Δ			$36\Delta w$	[204]
pathwidth p			p	[204]
band-width b	$b-1$	[184]	$\lceil \frac{b}{2} \rceil$	[114]
track-number t			$t-1$	Lemma 5.14, [204]
bipartite, track-number t	$2(t-1)$	Lemma 5.15		
toroidal	7	[71]		
planar	4	[207]		
bipartite planar	2	[33, 157]		
2-trees	2	[165]	3	[59, 165]
Halin	2	[84]	3	[84]
X-trees	2	[25]	2	[114]
outerplanar	1	[7]	2	[110]
arched levelled planar	2	[110]	1	[110]
trees	1	[25]	1	[114]

TABLE 6.2: Upper bounds on the track-number.

graph family	track-number	reference
n vertices	n	trivial
m edges	$15m^{2/3}$	[58]
m edges, max. degree Δ	$14\sqrt{\Delta m}$	[58]
no K_h-minor	$\mathcal{O}(h^{3/2}n^{1/2})$	[58]
genus γ	$\mathcal{O}(\gamma^{1/2}n^{1/2})$	[58]
treewidth w	$3^w \cdot 6^{(4^w-3w-1)/9}$	Theorem 6.2
treewidth w, max. degree Δ	$72\Delta w$	Lemma 6.3
pathwidth p	$p+1$	Lemma 6.2
band-width b	$b+1$	Theorem 6.3(a)
2-trees	18	[59]
Halin[4]	8	[40]
outerplanar[4]	6	[80]
arched levelled planar	5	[40]
trees	3	[80]

[4]Felsner *et al.* [80] proved that outerplanar graphs have improper 3-track layouts and that Halin graphs have

To prove our results for graphs of bounded treewidth, we employ a related structure called a tree-partition, introduced independently by Seese [177] and Halin [101]. A *tree-partition* of a graph is a partition of its vertices into 'bags' such that contracting each bag to a single vertex gives a forest (after deleting loops and replacing parallel edges by a single edge). It is well-known that a graph with treewidth k is a subgraph of a k-*tree* (see Section 6.1.1 for the definition). In a result of independent interest, we prove that every k-tree has a tree-partition such that each bag induces a connected $(k-1)$-tree, amongst other properties.

The remainder of the chapter is organized as follows. In Section 6.1 we introduce the required background material. Sections 6.2 and 6.3 bound the track-number of a graph by its "pathwidth" and "tree-partition-width". In Section 6.4 we prove the above-mentioned theorem for tree-partitions of k-trees, which is used in Section 6.5 to construct monochromatic track layouts of graphs with bounded treewidth. The resulting bound on the track-number is used in Section 6.6 to bound the queue-number of graphs by their treewidth. Final remarks are given in Section 6.7.

In this chapter all track layouts are monochromatic. Recall the definition from Section 2.2.3.

6.1 Preliminaries

6.1.1 Treewidth

Let G be a graph and let T be a tree. An element of $V(T)$ is called a *node*. Let $\{T_x \subseteq V(G) : x \in V(T)\}$ be a set of subsets of $V(G)$ indexed by the nodes of T. Each T_x is called a *bag*. The pair $(T, \{T_x : x \in V(T)\})$ is a *tree-decomposition* of G if:

- $\bigcup_{x \in V(T)} T_x = V(G)$ (that is, every vertex of G is in at least one bag),
- $\forall$ edge vw of G, $\exists$ node x of T such that $v \in T_x$ and $w \in T_x$, and
- $\forall$ nodes x, y, z of T, if y is on the path from x to z in T, then $T_x \cap T_z \subseteq T_y$.

The *width* of a tree-decomposition is one less than the maximum cardinality of a bag. A *path-decomposition* is a tree-decomposition where the tree T is a path $T = (x_1, x_2, \ldots, x_m)$, which is simply identified by the sequence of bags $T_1, T_2, \ldots, T_m$ where each $T_i = T_{x_i}$. The *pathwidth* (respectively, *treewidth*) of a graph G, denoted by $\mathsf{pw}(G)$ ($\mathsf{tw}(G)$), is the minimum width of a path- (tree-) decomposition of G. Graphs with treewidth at most one

improper 4-track layouts. Thus by Observation 2.1, the bounds in the table follow.

are precisely the forests. Graphs with treewidth at most two are called *series-parallel*[1], and are characterized as those graphs with no K_4 minor (see [14]).

A k-*tree* for some $k \in \mathbb{N}$ is defined recursively as follows. The empty graph is a k-tree, and the graph obtained from a k-tree by adding a new vertex adjacent to each vertex of a clique with at most k vertices is also a k-tree. This definition of a k-tree is by Rautenbach and Reed [164]. The following more restrictive definition of a k-tree, which we call 'strict', was introduced by Arnborg and Proskurowski [4], and is more often used in the literature. A k-clique is a *strict* k-*tree*, and the graph obtained from a strict k-tree by adding a new vertex adjacent to each vertex of a k-clique is also a strict k-tree. Obviously the strict k-trees are a proper sub-class of the k-trees. A subgraph of a k-tree is called a *partial* k-*tree*, and a subgraph of a strict k-tree is called a *partial strict* k-*tree*. The following result is well-known (see for example [14, 164]).

Lemma 6.1. *Let G be a graph. The following are equivalent:*

(a) *G has treewidth* $\mathsf{tw}(G) \leq k$*,*

(b) *G is a partial k-tree,*

(c) *G is a partial strict k-tree,*

(d) *G is a subgraph of a chordal graph with no clique on $k+2$ vertices.*

Proof Outline. Scheffler [175] proved that (a) and (c) are equivalent. That (a) and (d) are equivalent is due to Robertson and Seymour [169]. That (b) and (d) are equivalent is the characterization of chordal graphs in terms of 'perfect elimination' vertex-orderings due to Fulkerson and Gross [81]. □

6.1.2 Tree-partitions

As in the definition of a tree-decomposition, let G be graph and let $\{T_x \subseteq V(G) : x \in V(T)\}$ be a partition of $V(G)$ into subsets (called *bags*) indexed by the nodes of a tree T. The pair $(T, \{T_x : x \in V(T)\})$ is a *tree-partition* of G if

- $\forall$ distinct nodes x and y of T, $T_x \cap T_y = \emptyset$, and
- $\forall$ edge vw of G, either
  - $\exists$ node x of T with $v \in T_x$ and $w \in T_x$ (vw is called an *intra-bag* edge), or
  - $\exists$ edge xy of T with $v \in T_x$ and $w \in T_y$ (vw is called an *inter-bag* edge).

[1]'Series-parallel digraphs' are often defined in terms of certain 'series' and 'parallel' composition operations. Bodlaender [14] proved that the underlying undirected graph of such a digraph has treewidth at most two.

The main property of tree-partitions which has been studied in the literature is the maximum size of a bag, called the *width* of the tree-partition [15, 44, 45, 101, 177]. The minimum width over all tree-partitions of a graph G is the *tree-partition-width*[2] of G, denoted by $\mathsf{tpw}(G)$. A graph with bounded degree has bounded tree-partition-width if and only if it has bounded treewidth [45]. In particular, for every graph G, Ding and Oporowski [44] proved that $\mathsf{tpw}(G) \leq 24\,\mathsf{tw}(G)\Delta(G)$, and Seese [177] proved that $\mathsf{tw}(G) \leq 2\,\mathsf{tpw}(G) - 1$.

Theorem 6.1 in Section 6.4 provides a tree-partition of a k-tree G with additional features besides small width. First, the subgraph induced by each bag is a connected $(k-1)$-tree. This allows us to perform induction on k. Second, in each non-root bag T_x the set of vertices in the parent bag of x with a neighbour in T_x form a clique. This feature is crucial in the intended application (Theorem 6.2). Finally the tree-partition has width at most $\max\{1, k(\Delta(G) - 1)\}$, which represents a constant-factor improvement over the above result by Ding and Oporowski [44] in the case of k-trees.

6.2 Pathwidth bounds track-number

The following algorithm for constructing a track layout makes use of the so-called normalized path-decompositions of Gupta *et al.* [96]. (The more general notion of normalized tree-decompositions was developed earlier by Gupta and Nishimura [95].) A path-decomposition $T_1, T_2, \dots, T_m$ of width k is *normalized* if $|T_i| = k+1$ for all odd i and $|T_i| = k$ for all even i, and $T_{i-1} \cap T_{i+1} = T_i$ for all even i. The algorithm of Gupta *et al.* [96] normalizes a path-decomposition while maintaining the width in linear-time.

Lemma 6.2. *Every graph G with pathwidth $\mathsf{pw}(G)$ has track-number $\mathsf{tn}(G) \leq \mathsf{pw}(G) + 1$.*

Proof. Let $k = \mathsf{pw}(G) + 1$, and let $T_1, T_2, \dots, T_m$ be a normalized path-decomposition of G with width $k-1$. For every vertex $v \in V(G)$, let $T_{\alpha(v)}$ and $T_{\beta(v)}$ be the first and last bags containing v. Construct a k-track assignment of G as follows. Let $T_1 = \{v_1, v_2, \dots, v_k\}$, and position each v_i as the leftmost vertex on track i, $1 \leq i \leq k$. Since the path-decomposition is normalized, for all bags T_j with j even, there is a unique vertex $x_j \in T_{j-1} \setminus T_j$; that is, $\beta(x_j) = j-1$. Similarly, for all bags T_j with $j > 1$ odd, there is a unique vertex $y_j \in T_j \setminus T_{j-1}$; that is, $\alpha(y_j) = j$.

The remainder of the track assignment is constructed by sweeping through the bags of the path-decomposition as follows (see Figure 6.1). For all odd $j = 3, 5, \dots, m$, position y_j in the same track as the vertex x_{j-2} and immediately to the right of x_{j-2}. Clearly, x_{j-2} was the rightmost vertex in the track before inserting y_j. Since $j - 2 = \beta(x_{j-1}) < \alpha(y_j) = j$, there is no bag containing both x_{j-2} and y_j, and no edge $x_{j-2}y_j \in E(G)$. Two vertices

[2] Tree-partition-width has also been called *strong treewidth* [15, 177].

in the same track are not in a common bag and are not adjacent, hence we have a track assignment of G.

Suppose there is an X-crossing between edges vw and xy. Without loss of generality, $v <_i x$ and $y <_j w$ for some tracks i and j. Thus $\beta(v) < \alpha(x)$ and $\beta(y) < \alpha(w)$. Since vw is an edge, v and w appear in some bag together; that is, $\alpha(w) \leq \beta(v)$, which implies that $\beta(y) < \alpha(x)$. This is the desired contradiction since x and y appear in some bag together. $\square$

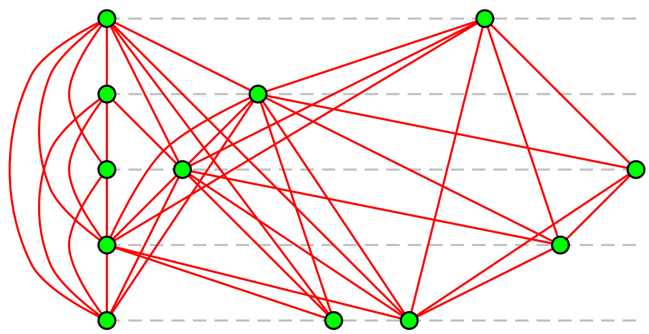


FIGURE 6.1: A 5-track layout produced by Lemma 6.2.

6.3 Tree-partition-width bounds track-number

The next lemma uses a tree-partition to construct a track layout.

Lemma 6.3. *Every graph G with maximum degree $\Delta(G)$, treewidth $\mathsf{tw}(G)$, and tree-partition-width $\mathsf{tpw}(G)$, has track-number $\mathsf{tn}(G) \leq 3\,\mathsf{tpw}(G)$ and $\mathsf{tn}(G) \leq 72\,\Delta(G)\mathsf{tw}(G)$.*

Proof. Let $(T, \{T_x : x \in V(T)\})$ be a tree-partition of G with width $\mathsf{tpw}(G)$. By Lemma 5.20(b), T has a 3-track layout. Replace each track by $\mathsf{tpw}(G)$ 'sub-tracks', and for each node x in T, place the vertices in bag T_x on the sub-tracks replacing the track containing x, with at most one vertex in T_x on a single track. The total order of each sub-track preserves the total order in each track of the track-layout of T. There is no X-crossing, since in the track layout of T, adjacent nodes are on distinct tracks and there is no X-crossing. Thus we have a monochromatic track layout of G. The number of tracks is $3\,\mathsf{tpw}(G)$, which is at most $72\,\Delta(G)\mathsf{tw}(G)$ by the theorem of Ding and Oporowski [44] discussed in Section 6.1.2. $\square$

6.4 Tree partitions

In this section we prove our theorem regarding tree-partitions of k-trees mentioned in Section 6.1.2. This result forms the cornerstone of the main theorem in this chapter, Theorem 6.2 in Section 6.5.

The *depth* of a vertex v_i in a vertex-ordering σ of graph a G is the graph-theoretic distance between v_1 and v_i in G. We say σ is a *generic breadth-first* vertex-ordering if for all vertices v and w with $v <_\sigma w$, the depth of v in σ is no more than the depth of w in σ.

Theorem 6.1. *Let G be a k-tree with maximum degree Δ. Then G has a rooted tree-partition $(T, \{T_x : x \in V(T)\})$ such that for all nodes x of T,*

(a) *if x is a non-root node of T and y is the parent node of x, then the set of vertices in T_y with a neighbour in T_x form a clique C_x of G, and*

(b) *the induced subgraph $G[T_x]$ is a connected $(k-1)$-tree.*

Furthermore the width of $(T, \{T_x : x \in V(T)\})$ is at most $\max\{1, k(\Delta-1)\}$.

Proof. We assume G is connected, since if G is not connected then a tree-partition of G which satisfies the theorem can be determined by adding a new root node with an empty bag which is adjacent to the root node of a tree-partition of each connected component of G.

It is well-known that for every vertex r of the k-tree G, there is a vertex-ordering $\sigma = (v_1, v_2, \dots, v_n)$ of G with $v_1 = r$, such that for all $i \in \{1, 2, \dots, n\}$,

(i) if G^i is the induced subgraph $G[\{v_1, v_2, \dots, v_i\}]$, then G^i is connected and the vertex-ordering of G^i induced by σ is a generic breadth-first vertex-ordering of G^i, and

(ii) the neighbours of v_i in G^i form a clique $C_i = \{v_j : v_iv_j \in E(G), j < i\}$ with $1 \leq |C_i| \leq k$ (unless $i = 1$ in which case $C_i = \emptyset$).

In the language of chordal graphs, σ is a (reverse) 'perfect elimination' vertex-ordering and can be determined, for example, by the Lex-BFS algorithm by Rose *et al.* [170] (also see [92]).

Let r be a vertex of minimum degree[3] in G. Then $\deg(r) \leq k$. Let $\sigma = (v_1, v_2, \dots, v_n)$ be a vertex-ordering of G with $v_1 = r$, and satisfying (i) and (ii). By (i), the depth of each vertex v_i in σ is the same as the depth of v_i in the vertex-ordering of G^j induced by σ, for all $j \geq i$. We therefore simply speak of *the* depth of v_i. Let V_d be the set of vertices of G at depth d.

Claim: For all $i \in \{1, 2, \dots, n\}$, for all $d \geq 1$, and for every connected component Z of $G^i[V_d]$, the set of vertices at depth $d-1$ with a neighbour in Z form a clique of G.

[3]We choose r to have minimum degree simply to prove a slightly improved bound on the width of the tree-partition. If we choose r to be an arbitrary vertex then the width is at most $\max\{1, \Delta, k(\Delta-1)\}$, and the remainder of Theorem 6.1 holds.

Proof. We proceed by induction on i. The result is trivially true for $i = 1$. Suppose it is true for $i - 1$.

Let d be the depth of v_i. Each vertex in C_i is at depth $d - 1$ or d. Let C'_i be the set of vertices in C_i at depth d, and let C''_i be the set of vertices in C_i at depth $d - 1$. Thus C'_i and C''_i are both cliques with $C_i = C'_i \cup C''_i$. Furthermore, if $i > 1$ then v_i must have a neighbour at depth $d - 1$, and thus $C''_i \neq \emptyset$.

Let X be the vertex set of the connected component of $G^i[V_d]$ such that $v_i \in X$. By induction, for all $d' \leq d$, the claim holds for all connected components Y of $G^i[V_{d'}]$ with $Y \neq X$, since such a Y is also a connected component of $G^{i-1}[V_{d'}]$.

Case 1. $C'_i = \emptyset$: Then v_i has no neighbours in G^i at depth d; that is, $X = \{v_i\}$. Thus the set of vertices at depth $d - 1$ with a neighbour in X is precisely the clique $C_i = C''_i$.

Case 2. $C'_i \neq \emptyset$: The neighbourhood of v_i in X forms a non-empty clique (namely C'_i). Thus $X \setminus v_i$ is the vertex-set of a connected component of $G^{i-1}[V_d]$. Let Y be the set of vertices at depth $d - 1$ with a neighbour in $X \setminus v_i$. By induction, Y is a clique. Since $C''_i \cup C'_i$ is a clique, $C''_i \subseteq Y$. Thus the set of vertices at depth $d - 1$ with a neighbour in X is the clique Y.

This completes the proof of the claim. □

Define a graph T and a partition $\{T_x : x \in V(T)\}$ of $V(G)$ indexed by the nodes of T as follows. There is one node x in T for every connected component of each $G[V_d]$, whose bag T_x is the vertex-set of the corresponding connected component. We say x and T_x are at *depth* d. Clearly a vertex in a depth-d bag is also at depth d. The (unique) node of T at depth zero is called the *root* node. Let two nodes x and y of T be connected by an edge if there is an edge vw of G with $v \in T_x$ and $w \in T_y$. Thus $(T, \{T_x : x \in V(T)\})$ is a 'graph-partition'.

We now prove that in fact T is a tree. First observe that T is connected since G is connected. By definition, nodes of T at the same depth d are not adjacent. Moreover nodes of T can be adjacent only if their depths differ by one. Thus T has a cycle only if there is a node x in T at some depth d, such that x has at least two distinct neighbours in T at depth $d - 1$. However this is impossible since by the above claim (with $i = n$), the set of vertices at depth $d - 1$ with a neighbour in T_x form a clique (which we call C_x), and are hence in a single bag at depth $d - 1$. Thus T is a tree and $(T, \{T_x : x \in V(T)\})$ is a tree-partition of G (see Figure 6.2).

We now prove that each bag T_x induces a connected $(k - 1)$-tree. This is true for the root node which only has one vertex. Suppose x is a non-root node of T at depth d. Each vertex in T_x has at least one neighbour at depth $d - 1$. Thus in the vertex-ordering of T_x induced by σ, each vertex $v_i \in T_x$ has at most $k - 1$ neighbours $v_j \in T_x$ with $j < i$. Thus $G[T_x]$ is a partial $(k - 1)$-tree. An induced subgraph of a k-tree is itself a k-tree. Thus $G[T_x]$ is a $(k - 1)$-tree. By definition each $G[T_x]$ is connected.

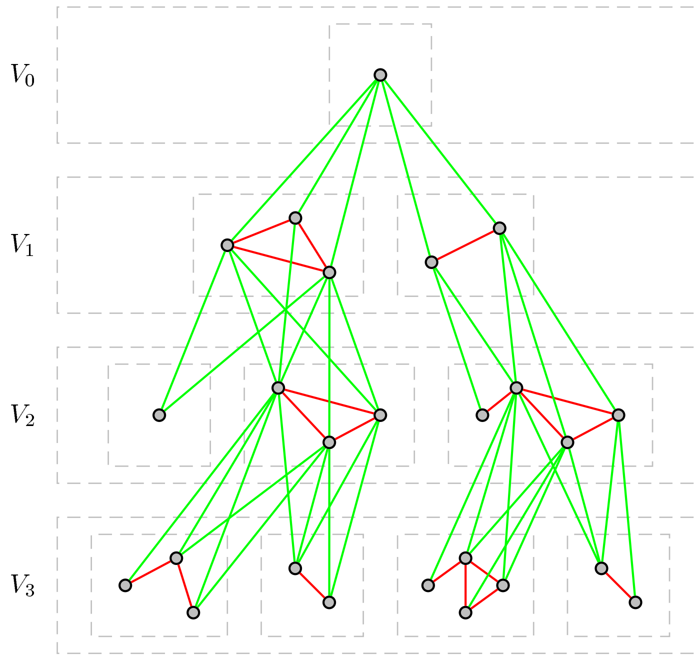


FIGURE 6.2: Illustration for Theorem 6.1 in the case of $k = 3$.

Finally, consider the size of a bag in T. We claim that each bag contains at most $\max\{1, k(\Delta - 1)\}$ vertices. The root bag has one vertex. Let x be a non-root node of T with parent node y. Suppose y is the root node. Then $T_y = \{r\}$, and thus $|T_x| \leq \deg(r) \leq k \leq k(\Delta - 1)$ assuming $\Delta \geq 2$. If $\Delta \leq 1$ then all bags have one vertex. Now assume y is a non-root node. The set of vertices in T_y with a neighbour in T_x forms the clique C_x. Let $k' = |C_x|$. Thus $k' \geq 1$, and since $C_x \subseteq T_y$ and $G[T_y]$ is a $(k-1)$-tree, $k' \leq k$. A vertex $v \in C_x$ has $k' - 1$ neighbours in C_x and at least one neighbour in the parent bag of y. Thus v has at most $\Delta - k'$ neighbours in T_x. Hence the number of edges between C_x and T_x is at most $k'(\Delta - k')$. Every vertex in T_x is adjacent to a vertex in C_x. Thus $|T_x| \leq k'(\Delta - k') \leq k(\Delta - 1)$. This completes the proof. $\square$

6.5 Treewidth bounds track-number

In this section we prove that track-number is bounded by treewidth. Let $\{V_i : 1 \leq i \leq t\}$ be a a t-track layout of a graph G. We say a clique C of G *covers* the set of tracks $\{i : C \cap V_i \neq \emptyset\}$. Let S be a set of cliques of G. Suppose there exists a total order $\preceq$ on S such that for all cliques $C_1, C_2 \in S$, if there exists a track i, and vertices $v \in V_i \cap C_1$ and $w \in V_i \cap C_2$ with $v <_i w$, then $C_1 \prec C_2$. In this case, we say $\preceq$ is *nice*, and S is *nicely ordered* by the track layout.

Lemma 6.4. *Let L be a (sub)set of tracks in a t-track layout $\{V_i : 1 \leq i \leq t\}$ of a graph G. If S is a set of cliques, each of which covers L, then S is nicely ordered by the given track layout.*

Proof. Define a relation $\preceq$ on S as follows. For every pair of cliques $C_1, C_2 \in S$, define $C_1 \preceq C_2$ if $C_1 = C_2$ or there exists a track $i \in L$ and vertices $v \in C_1$ and $w \in C_2$ with $v <_i w$. Clearly all cliques in S are comparable.

Suppose that $\preceq$ is not antisymmetric; that is, there exists distinct cliques $C_1, C_2 \in S$, distinct tracks $i, j \in L$, and distinct vertices $v_1, w_1 \in C_1$ and $v_2, w_2 \in C_2$, such that $v_1 <_i v_2$ and $w_2 <_j w_1$. Since C_1 and C_2 are cliques, the edges v_1w_1 and v_2w_2 form an X-crossing, which is a contradiction. Thus $\preceq$ is antisymmetric.

We claim that $\preceq$ is transitive. Suppose there exist cliques $C_1, C_2, C_3 \in S$ such that $C_1 \preceq C_2$ and $C_2 \preceq C_3$. We can assume that C_1, C_2 and C_3 are pairwise distinct. Thus there are vertices $u_1 \in C_1$, $u_2 \in C_2$, $v_2 \in C_2$ and $v_3 \in C_3$, such that $u_1 <_i u_2$ and $v_2 <_j v_3$ for some pair of (not necessarily distinct) tracks $i, j \in L$. Since C_3 has a vertex in V_i and since $C_3 \not\preceq C_2$, there is a vertex $u_3 \in C_3$ with $u_2 \leq_i u_3$. Thus $u_1 <_i u_3$, which implies that $C_1 \preceq C_3$. Thus $\preceq$ is transitive.

Hence $\preceq$ is a total order on S, which by definition is nice. $\square$

Consider the problem of partitioning the cliques of a graph into sets such that each set is nicely ordered by a given track layout. The following immediate corollary of Lemma 6.4 says that there exists such a partition where the number of sets does not depend upon the size of the graph.

Corollary 6.1. *Let G be a graph with maximum clique size k. Given a t-track layout of G, there is a partition of the cliques of G into $\sum_{i=1}^{k} \binom{t}{i}$ sets, each of which is nicely ordered by the given track layout.* $\square$

We do not actually use Corollary 6.1 in the following result, but the idea of partitioning the cliques into nicely ordered sets is central to its proof.

Theorem 6.2. *For every integer $k \geq 0$, there is a constant $c_k = 3^k \cdot 6^{(4^k - 3k - 1)/9}$ such that every graph G with treewidth $\mathsf{tw}(G) \leq k$ has a c_k-track layout.*

Proof. If the input graph G is not a k-tree then add edges to G to obtain a k-tree containing G as a subgraph. It is well-known that a graph with treewidth at most k is a *spanning* subgraph of a k-tree. These extra edges can be deleted once we are done. We proceed by induction on k with the following induction hypothesis.

For all $k \in \mathbb{N}$, there exists constants s_k and c_k, and sets I and S such that

1. *$|I| = c_k$ and $|S| = s_k$,*

2. *each element of S is a subset of I, and*

3. *every k-tree G has a c_k-track layout with tracks indexed by I, such that for every clique C of G, the set of tracks which C covers is in S.*

Consider the base case with $k = 0$. A 0-tree G has no edges and thus has a 1-track layout. Let $I = \{1\}$ and order $V_1 = V(G)$ arbitrarily. Thus $c_0 = 1$, $s_0 = 1$, and $S = \{\{1\}\}$ satisfy the hypothesis for every 0-tree. Now suppose the result holds for $k-1$, and G is a k-tree. Let $(T, \{T_x : x \in V(T)\})$ be a tree-partition of G described in Theorem 6.1, where T is rooted at r.

By Theorem 6.1 each induced subgraph $G[T_x]$ is a $(k-1)$-tree. By induction, there are sets I and S with $|I| = c_{k-1}$ and $|S| = s_{k-1}$, such that for every node x of T, the induced subgraph $G[T_x]$ has a c_{k-1}-track layout indexed by I. For every clique C of $G[T_x]$, if C covers $L \subseteq I$ then $L \in S$. Assume $I = \{1, 2, \dots, c_{k-1}\}$ and $S = \{S_1, S_2, \dots, S_{s_{k-1}}\}$. By Theorem 6.1, for each non-root node x of T and its parent node $\rho(x)$, the set of vertices in $T_{\rho(x)}$ with a neighbour in T_x form a clique C_x. Let $\alpha(x) = i$ where C_x covers S_i. For the root node r of T, let $\alpha(r) = 1$.

Track layout of T

To construct a monochromatic track layout of G we first construct a monochromatic track layout of the tree T indexed by the set $\{(d, i) : d \geq 0, 1 \leq i \leq s_{k-1}\}$, where the track $L_{d,i}$ consists of nodes x of T at depth d with $\alpha(x) = i$. We order the nodes of T within the tracks by increasing depth. There is only one node at depth $d = 0$. Suppose we have determined the orders of the nodes up to depth $d-1$ for some $d \geq 1$.

Let $i \in \{1, 2, \dots, s_{k-1}\}$. The nodes in $L_{d,i}$ are ordered primarily with respect to the relative positions of their parent nodes (at depth $d-1$). More precisely, for all nodes x and y in $L_{d,i}$, if their parents $\rho(x)$ and $\rho(y)$ are in the same track and $\rho(x) < \rho(y)$ in that track, then $x < y$ in $L_{d,i}$. For x and y with $\rho(x)$ and $\rho(y)$ on distinct tracks, the relative order of x and y is not important. It remains to specify the order of nodes in $L_{d,i}$ with a common parent.

Suppose P is a set of nodes in $L_{d,i}$ with a common parent node p. By construction, for every node $x \in P$, the parent clique C_x covers S_i in the track layout of $G[T_p]$. By Lemma 6.4 the cliques $\{C_x : x \in P\}$ are nicely ordered by the track layout of $G[T_p]$. Let the order of P in track $L_{d,i}$ be specified by a nice ordering of $\{C_x : x \in P\}$, as illustrated in Figure 6.3.

This construction defines a partial order on the nodes in track $L_{d,i}$, which can be arbitrarily extended to a total order. Hence we have a track assignment of T. Since the nodes in each track are ordered primarily with respect to the relative positions of their parent nodes in the previous tracks, there is no X-crossing, and hence we have a monochromatic track layout of T.

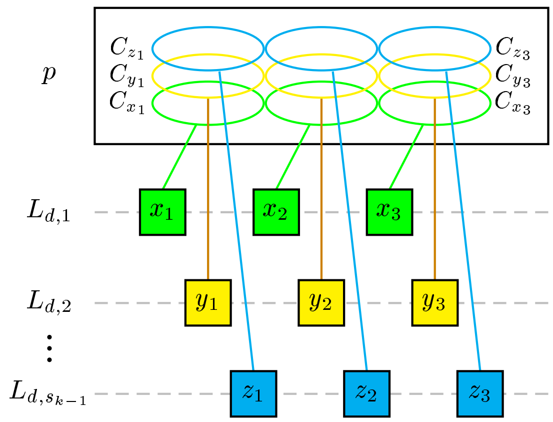


FIGURE 6.3: Track layout of nodes with a common parent p.

Track layout of G

To construct a track assignment of G from the track layout of T, replace each track $L_{d,i}$ by c_{k-1} 'sub-tracks', and for each node x of T, insert the track layout of $G[T_x]$ in place of x on the sub-tracks corresponding to the track containing x in the track layout of T (as illustrated in Figure 6.4). More formally, the track layout of G is indexed by the set

$$\{(d, i, j) : d \geq 0, 1 \leq i \leq s_{k-1}, 1 \leq j \leq c_{k-1}\} \ .$$

Each track $V_{d,i,j}$ consists of those vertices v of G such that, if T_x is the bag containing v, then x is at depth d in T, $\alpha(x) = i$, and v is on track j in the track layout of $G[T_x]$. If x and y are distinct nodes of T with $x < y$ in $L_{d,i}$, then $v < w$ in $V_{d,i,j}$, for all vertices $v \in T_x$ and $w \in T_y$ on track j. If v and w are vertices of G on track j in bag T_x at depth d, then the relative order of v and w in $V_{d,\alpha(x),j}$ is the same as in the track layout of $G[T_x]$.

Clearly adjacent vertices of G are in distinct tracks. Thus we have defined a track assignment of G. We claim there is no X-crossing. Clearly an intra-bag edge of G is not in an X-crossing with an edge not in the same bag. By induction, there is no X-crossing between intra-bag edges in a common bag. Since there is no X-crossing in the track layout of T, inter-bag edges of G which are mapped to edges of T without a common parent node, are not involved in an X-crossing.

Consider a parent node p in T. For each child node x of p, the set of vertices in T_p adjacent to a vertex in T_x forms the clique C_x. Thus there is no X-crossing between a pair of edges both from C_x to T_x, since the vertices of C_x are on distinct tracks. Consider two child nodes x and y of p. For there to be an X-crossing between an edge from T_p to T_x and an edge from T_p to T_y, the nodes x and y must be on the same track in the track layout of T. Suppose $x < y$ in this track. By construction, C_x and C_y cover the same set of tracks, and

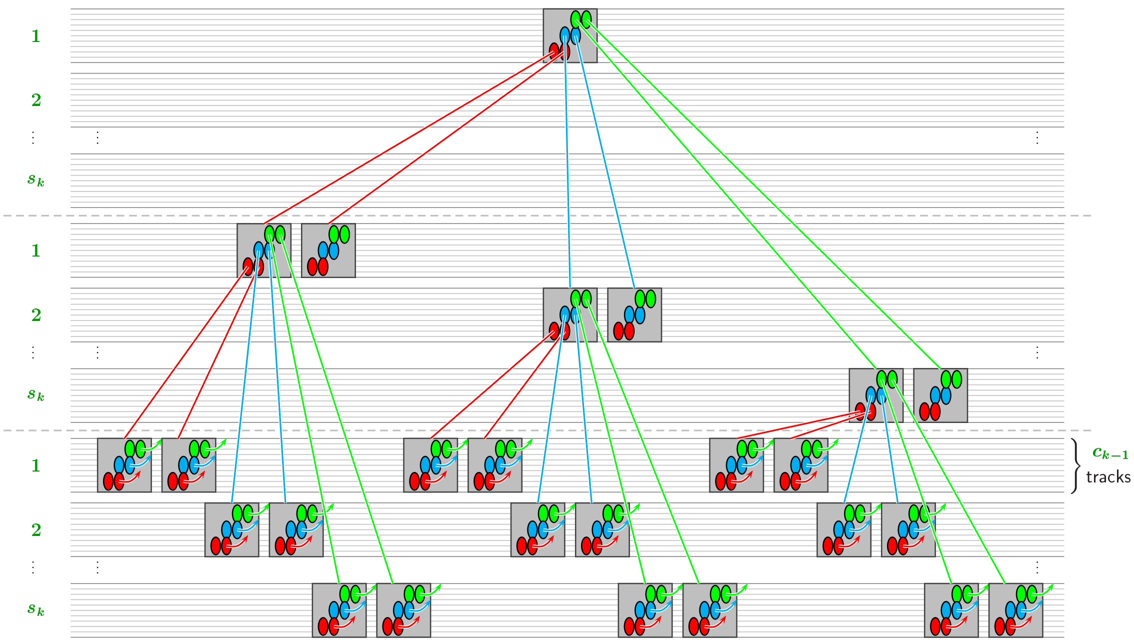


FIGURE 6.4: Track layout of G

$C_x \preceq C_y$ in the corresponding nice ordering. Thus for any track containing vertices $v \in C_x$ and $w \in C_y$, $v \leq w$ in that track. Since all the vertices in T_x are to the left of the vertices in T_y (on a common track), there is no X-crossing between an edge from T_p to T_x and an edge from T_p to T_y. Therefore there is no X-crossing, and hence we have a monochromatic track layout of G.

Wrapped track layout of G

For every edge vw of G, the depths of the bags in T containing v and w differ by at most one. Therefore, the obtained track layout of G indexed by $\{(d,i,j) : d \geq 0, 1 \leq i \leq s_{k-1}, 1 \leq j \leq c_{k-1}\}$ has partial span equal to one (considering d to be the first index and i and j together to be the second index of the track layout). Therefore by Lemma 5.6 with $s = 1$ and $b = s_{k-1} \cdot c_{k-1}$, this track layout can be wrapped into the monochromatic track layout of G with $3 \cdot s_{k-1} \cdot c_{k-1}$ tracks. In particular, we obtain the track layout of G indexed by

$$\left\{(d', i, j) : d' \in \{0,1,2\}, 1 \leq i \leq s_{k-1}, 1 \leq j \leq c_{k-1}\right\} \ ,$$

where each track

$$W_{d',i,j} \ = \bigcup \{V_{d,i,j} : d \equiv d' \pmod 3\} \ .$$

Every clique C of G is either contained in a single bag of the tree-partition or is contained in two adjacent bags. Let

$$S' = \big\{\{(d', i, h) : h \in S_j\} : d' \in \{0,1,2\}, 1 \le i, j \le s_{k-1}\big\} \ .$$

For every clique C of G contained in a single bag, the set of tracks containing C is in S'. Let

$$\begin{aligned} S'' = \big\{&\{(d', i, h) : h \in S_j\} \cup \{((d'+1) \bmod 3, p, h) : h \in S_q\} : \\ &d' \in \{0,1,2\}, 1 \le i, j, p, q \le s_{k-1}\big\} \ . \end{aligned}$$

For every clique C of G contained in two bags, the set of tracks containing C is in S'. Observe that $S' \cup S''$ is independent of G. Hence $S' \cup S''$ satisfies the hypothesis for k.

Now $|S'| = 3s_{k-1}^2$ and $|S''| = 3s_{k-1}^4$, and thus $|S' \cup S''| = 3s_{k-1}^2(s_{k-1}^2 + 1)$. Therefore any solution to the following set of recurrences satisfies the theorem:

$$\begin{aligned} s_0 &\ \ge\ 1 \\ c_0 &\ \ge\ 1 \\ s_k &\ \ge\ 3s_{k-1}^2(s_{k-1}^2 + 1) \\ c_k &\ \ge\ 3s_{k-1} \cdot c_{k-1} \ . \end{aligned} \tag{6.1}$$

We claim that

$$s_k = 6^{(4^k-1)/3} \text{ and } c_k = 3^k \cdot 6^{(4^k-3k-1)/9}$$

is a solution to (6.1). Observe that $s_0 = 1$ and $c_0 = 1$. Now

$$3s_{k-1}^2(s_{k-1}^2 + 1) \ \le\ 6s_{k-1}^4 \ ,$$

and

$$6(6^{(4^{k-1}-1)/3})^4 \ =\ 6^{1+4(4^{k-1}-1)/3} \ =\ 6^{(4^k-1)/3} \ =\ s_k \ .$$

Thus the recurrence for s_k is satisfied. Now

$$\begin{aligned} 3 \cdot s_{k-1} \cdot c_{k-1} &= 3 \cdot 6^{(4^{k-1}-1)/3} \cdot 3^{k-1} \cdot 6^{(4^{k-1}-3(k-1)-1)/9} \\ &= 3^k \cdot 6^{(3\cdot 4^{k-1}-3+4^{k-1}-3k+3-1)/9} \\ &= 3^k \cdot 6^{(4^k-3k-1)/9} \\ &= c_k \ . \end{aligned}$$

Thus the recurrence for c_k is satisfied. This completes the proof. □

In the proof of Theorem 6.2 we have made little effort to reduce the bound on c_k, beyond that it is a doubly exponential function of k. In [59] we describe a number of refinements that result in improved bounds on c_k. One such refinement uses strict k-trees. From an algorithmic point of view, the disadvantage of using strict k-trees is that at each recursive step, extra edges must be added to enlarge the graph from a partial strict k-tree into a strict k-tree, whereas when using (non-strict) k-trees, extra edges need only be added at the beginning of the algorithm.

The following theorem summarizes our bounds on the track-number of a graph. The *band-width* of a vertex ordering σ of G is the maximum $|i-j|$ of an edge v_iv_j of G in σ. The *band-width* of G, denoted by $\mathsf{bw}(G)$, is the minimum band-width over all vertex orderings of G.

Theorem 6.3. *Let G be a graph with maximum degree $\Delta(G)$, pathwidth $\mathsf{pw}(G)$, tree-partition-width $\mathsf{tpw}(G)$, and treewidth $\mathsf{tw}(G)$. The track-number of G satisfies:*

(a) $\mathsf{tn}(G) \leq 1 + \mathsf{pw}(G) \leq \begin{cases} 1 + (\mathsf{tw}(G)+1)\ \log n; \\ 1 + \mathsf{bw}(G). \end{cases}$

(b) $\mathsf{tn}(G) \leq 3\,\mathsf{tpw}(G) \leq 72\,\Delta(G)\,\mathsf{tw}(G)$,

(c) $\mathsf{tn}(G) \leq 3^{\mathsf{tw}(G)} \cdot 6^{(4^{\mathsf{tw}(G)} - 3\,\mathsf{tw}(G) - 1)/9}$.

Proof. Part (a) follows from Lemma 6.2, and since $\mathsf{pw}(G) \leq (\mathsf{tw}(G)+1)\log n$ and $\mathsf{pw}(G) \leq \mathsf{bw}(G)$ (see [14]). Note that $\mathsf{tn}(G) \leq 1 + (\mathsf{tw}(G)+1)\log n$ can be proved directly using a separator-based approach similar to that used to prove $\mathsf{pw}(G) \leq (\mathsf{tw}(G)+1)\log n$. Part (b) follows from Lemma 6.3. Part (c) is Theorem 6.2. □

6.6 Treewidth bounds queue-number

Applying Lemma 5.14 and Theorem 6.3 we have the following.

Theorem 6.4. *Let G be a graph with maximum degree $\Delta(G)$, pathwidth $\mathsf{pw}(G)$, tree-partition-width $\mathsf{tpw}(G)$, and treewidth $\mathsf{tw}(G)$. The queue-number $\mathsf{qn}(G)$ satisfies*[4]*:*

(a) $\mathsf{qn}(G) \leq \mathsf{pw}(G)$

(b) $\mathsf{qn}(G) \leq 3\,\mathsf{tpw}(G) - 1 \leq 72\Delta(G)\,\mathsf{tw}(G) - 1$,

(c) $\mathsf{qn}(G) \leq 3^{\mathsf{tw}(G)} \cdot 6^{(4^{\mathsf{tw}(G)} - 3\,\mathsf{tw}(G) - 1)/9} - 1$. □

[4] Wood [204] obtained an alternative proof that $\mathsf{qn}(G) \leq \mathsf{pw}(G)$ using the 'vertex separation number' of a graph (which equals its pathwidth), and by applying Lemma 2.3 directly, he proved that $\mathsf{qn}(G) \leq \frac{3}{2}\,\mathsf{tpw}(G)$, and thus $\mathsf{qn}(G) \leq 36\,\Delta(G)\,\mathsf{tw}(G)$.

A similar upper bound to Theorem 6.4(a) is obtained by Heath and Rosenberg [114], who proved that every graph G has $\mathsf{qn}(G) \leq \lceil \frac{1}{2}\mathsf{bw}(G) \rceil$, where $\mathsf{bw}(G)$ is the band-width of G. In many cases this result is weaker than Theorem 6.4(a) since $\mathsf{pw}(G) \leq \mathsf{bw}(G)$. More importantly, we have the following corollary of Theorem 6.4(c).

Corollary 6.2. *Queue-number is bounded by treewidth, and hence graphs with bounded treewidth have bounded queue-number.* □

6.7 Conclusion and bibliographic notes

Ganley and Heath [85] proved that for every graph G, the stack-number $\mathsf{sn}(G) \leq \mathsf{tw}(G) + 1$ (using a depth-first traversal of a tree-decomposition), and asked whether queue-number is bounded by treewidth. One of the principal results of this chapter is Theorem 6.4, which solves this question in the affirmative.

The best known upper bound on the queue-number of a planar graph is $O(\sqrt{n})$. This result can be proved using a variant of the randomized algorithm of Malitz [141] (see [110]), or the derandomized algorithm of Shahrokhi and Shi [178]. The result also follows from Theorem 6.4(a) since the pathwidth of a planar graph is $O(\sqrt{n})$ (see [14]). Heath *et al.* [110] asked the following open problem.

Open Problem 6.1. [110] Do planar graphs have bounded queue-number?

Since planar graphs have stack-number at most four [207], this question is less general than Open Problem 1.2. Heath *et al.* [110, 114] originally conjectured that both of these questions have an affirmative answer. More recently however, Pemmaraju [160] conjectured that the 'stellated K_3', a planar 3-tree, has $\Theta(\log n)$ queue-number, and provided evidence to support this conjecture (also see [85]). This suggested that the answer to both Open Problems 1.2 and 6.1 was negative. In particular, Pemmaraju [160] and Heath [private communication, 2002] conjectured that planar graphs have $\mathcal{O}(\log n)$ queue-number. However, our result provides a queue layout for *any* 3-tree, and thus in particular for the stellated K_3, with $\mathcal{O}(1)$ queues. Hence our result disproves the first conjecture of Pemmaraju [160] mentioned above, and renews hope of an affirmative answer to Open Problems 1.2 and 6.1 .

The work presented in this chapter leaves the following open problem unanswered.

Open Problem 6.2. Is the queue-number of a graph bounded by a polynomial function of its treewidth?

The result of Section 6.2 has appeared in [53]. The remainder of the material in this chapter has appeared in [59].

Chapter 7

Layouts of Subdivisions

A *subdivision* of a graph G is a graph obtained from G by replacing each edge $vw \in E(G)$ by a path $v, x_1, x_2, \dots, x_p, w$ where $p \geq 0$. Internal vertices on this path, $x_1, x_2, \dots, x_p$, are called *division* vertices, while v and w are called *original* vertices. Let G' and G'' be the subdivisions of G with exactly one and two division vertices per edge, respectively. Throughout this chapter, we implicitly use the fact that planarity and non-planarity are preserved by subdividing edges.

Let α be a graph parameter. Let $\mathsf{sub}\text{-}\alpha$ be the graph parameter defined by $\mathsf{sub}\text{-}\alpha(G) = \alpha(G')$ for every graph G. We say α is *topological* if α and $\mathsf{sub}\text{-}\alpha$ are tied. (Recall the definition of tied from Section 2.1.2.) For example, chromatic number is not topological since G' is bipartite. On the other hand treewidth is topological. In fact, it is well-known that the treewidth of a graph G equals the treewidth of every subdivision of G (see Diestel [41, Exercise 13, p. 278]).

This chapter is organized as follows. In Section 7.1 we review known results concerning layouts of graph subdivisions. In Section 7.2 we study the layouts of small subdivisions, G' and G''. Section 7.3 presents most of our main results discussed below (see Table 7.1 on page 95). Section 7.4 considers layouts of subdivisions of planar graphs. Final remarks are given in Section 7.5.

7.1 Introduction and preliminaries

7.1.1 Stack, queue and track layouts of subdivisions

Stack, queue and track layouts of graph subdivisions are a central topic of this chapter. The following fundamental result has been observed by many authors [10, 72, 73, 142]. The well-known proof, which we include for completeness, can be traced to the seminal result by Atneosen [5] that every graph has an embedding in a 3-page book. Kainen and Overbay

[124] state that, according to Jozef Przytycki, this result was also discovered by Holtz, a student of Reidemeister.

Theorem 7.1. [10, 72, 73, 142] *Every graph has a* 3*-stack subdivision.*

Proof. Let σ be an arbitrary vertex ordering of a given graph G. Consider the graph G'' with each edge of G subdivided twice. For each vertex $v \in V(G)$, insert into σ the vertices $\{x : vx \in E(G'')\}$ immediately to the right of v, and assign the edges $E^* = \{vx : v \in V(G), vx \in E(G'')\}$ to the first stack. Clearly no two edges in E^* cross in σ. It remains to assign a subdivision of the matching $E(G'') \setminus E^*$ to the remaining two stacks. This amounts to drawing a matching in the plane with no edge crossings such that the vertices are fixed to a line. Clearly this can be accomplished. An edge of $E(G'') \setminus E^*$ is subdivided every time it crosses the line. Thus every graph has a 3-stack subdivision. $\square$

Note that 3-stack layouts are important in complexity theory [82, 83, 125], and 3-stack layouts of knots and links, so called *Dynnikov digrams*, have also recently been considered [31, 60–63, 132, 145].

The proof of Theorem 7.1 provides no bound on the number of division vertices. It is interesting to determine the minimum number of division vertices in a 3-stack subdivision of a given graph. The previously best bounds are due to Enomoto and Miyauchi [72], who proved that every graph has a 3-stack subdivision with $\mathcal{O}(\log n)$ division vertices per edge. Moreover, a trade-off between the number of stacks and the number of division vertices per edge was observed. In particular, Enomoto and Miyauchi [73] proved that for all $s \geq 3$, every graph has an s-stack subdivision with $\mathcal{O}(\log_{s-1} n)$ division vertices per edge, and Enomoto *et al.* [74] proved that this bound is tight up to a constant factor for K_n (and some slightly more general families). Thus Enomoto *et al.* [74] claimed that the $\mathcal{O}(\log n)$ upper bound is 'essentially best possible'.

We prove a refinement of the upper bound of Enomoto and Miyauchi [72], in which the number of division vertices per edge depends on the stack-number or queue-number of the given graph. In particular, every graph G has a 3-stack subdivision with $\mathcal{O}(\log \min\{\mathsf{sn}(G), \mathsf{qn}(G)\})$ division vertices per edge. Since $\mathsf{sn}(G)$ and $\mathsf{qn}(G)$ are both no more than n, our bound is at most the $\mathcal{O}(\log n)$ bound of Enomoto and Miyauchi [72] (ignoring constant factors). This result has a significant implication for Open Problem 1.2. Namely that queue-number is bounded by stack-number if and only if 3-stack graphs have bounded queue-number (Theorem 7.8). For this corollary to hold, it is essential that the number of division vertices per edge is some function of $\mathsf{sn}(G)$, thus emphasizing the significance of our bound in comparison with the previous results. As described in Table 7.1 our result for 3-stack subdivisions generalizes to s-stack subdivisions in a similar fashion to the result of Enomoto and Miyauchi [73].

We prove an analogous result for queue layouts. In particular, every graph G has a 2-queue subdivision with $\mathcal{O}(\log \mathsf{qn}(G))$ division vertices per edge. Thus, at least for the representation of graph subdivisions, two queues suffice rather than three stacks. In this sense, queues are more powerful than stacks. Moreover, our bound on the number of division vertices per edge is optimal up to a constant factor for all graphs. Unfortunately, no such universal lower bound is known for stack layouts of subdivisions.

Stack and queue layouts are generalized through the notion of a *mixed* layout, as defined in Section 2.2.2. Observe that the proof of Theorem 7.1 implies that every graph has a 2-stack 1-queue subdivision, since the first stack is also a queue, whereas we prove that every graph has a 1-stack 1-queue subdivision.

Our main result concerning track layouts highlights the trade-off between few tracks and few edge colours. We prove that every graph G has a subdivision D with $\mathcal{O}(\log \mathsf{qn}(G))$ division vertices per edge, such that (a) D has a $(1,4)$-track layout, (b) D has a $(2,3)$-track layout, and (c) D has a $(3,2)$-track layout. We shall see that all of these numeric values are best possible for any non-planar graph G. Moreover, the number of division vertices per edges is optimal, since any subdivision satisfying (a), (b) or (c) has an edge with $\Omega(\log \mathsf{qn}(G))$ division vertices. Note that for all $d \geq 2$, the result generalizes to $(1, d+2)$-, $(d,3)$-, and $(d+1,2)$-track layouts. Table 7.1 summarizes the bounds on stack-, queue-, mixed- and track-layouts of graph subdivisions established in this chapter.

TABLE 7.1: Layouts of a subdivision of a graph G.

graph	type of layout		# division vertices per edge	reference
arbitrary	s-stack	$(s \geq 3)$	$\mathcal{O}(\log_{s-1} \mathsf{sn}(G))$	Theorem 7.7
arbitrary	s-stack	$(s \geq 3)$	$\mathcal{O}(\log_{s-1} \mathsf{qn}(G))$	Theorem 7.9
planar	2-stack		1	[38, 127], Lemma 7.20
arbitrary	q-queue	$(q \geq 2)$	$\Theta(\log_q \mathsf{qn}(G))$	Theorems 7.5 and 7.6
planar	1-queue		$n-2$	Theorem 7.19
arbitrary	s-stack q-queue	$(s \geq 1,\, q \geq 1)$	$\mathcal{O}(\log_{(s+q)q} \mathsf{sn}(G))$	Theorem 7.10
arbitrary	s-stack q-queue	$(s \geq 1,\, q \geq 1)$	$\mathcal{O}(\log_{(s+q)q} \mathsf{qn}(G))$	Theorem 7.11
planar	1-stack 1-queue		4	Lemma 7.23
arbitrary	$(d+1,2)$-track	$(d \geq 2)$	$\Theta(\log_d \mathsf{qn}(G))$	Theorems 7.13 and 7.16
arbitrary	$(d,3)$-track	$(d \geq 2)$	$\Theta(\log_d \mathsf{qn}(G))$	Theorems 7.14 and 7.16
arbitrary	$(d+2)$-track	$(d \geq 2)$	$\Theta(\log_d \mathsf{qn}(G))$	Theorems 7.15 and 7.16
planar	3-track		$n-2$	Theorem 7.19

7.1.2 Thickness and topological parameters

The *thickness* of a graph G, denoted by $\theta(G)$, is the minimum number of subgraphs in a partition of $E(G)$ into planar subgraphs [123]. Thickness is not topological since $\theta(G') \leq 2$. Beineke [6] attributes this observation to Tutte. The proof is straightforward. Let $V(G) = \{v_1, v_2, \dots, v_n\}$. Denote by $x_{i,j}$ the division vertex of each edge v_iv_j with $i < j$. Then $\{v_ix_{i,j} : 1 \leq i < j \leq n\}$ and $\{v_ix_{j,i} : 1 \leq j < i \leq n\}$ is a partition of $E(G')$ in two (planar) forests.

The key difference between geometric thickness (as defined in Section 5.3) and (graph-theoretic) thickness is that geometric thickness requires the edges to be drawn as straight line-segments, whereas thickness allows edges to bend arbitrarily. Eppstein [75] proved that $\overline{\theta}(G') \leq 2$ for every graph G. Thus geometric thickness is not topological.

Stack-number (or book-thickness) is equivalent to geometric thickness with the additional requirement that the vertices are in convex position [7]. Thus

$$\forall \text{ graph } G,\ \theta(G) \leq \overline{\theta}(G) \leq \mathsf{sn}(G) \ . \tag{7.1}$$

Blankenship and Oporowski [10], Enomoto and Miyauchi [72], and Eppstein [75] independently proved that $\mathsf{sn}(K_n)$ is bounded by $\mathsf{sn}(K_n')$. The proofs by Blankenship and Oporowski [10] and Eppstein [75] use essentially the same Ramsey-theoretic argument. Since $\overline{\theta}(K_n') = 2$, Eppstein [75] observed that stack-number is not bounded by geometric thickness. Using a more elaborate Ramsey-theoretic argument, Eppstein [75] proved that geometric thickness is not bounded by thickness. In particular, for every t there exists a graph with thickness three and geometric thickness at least t. Blankenship and Oporowski [10] conjecture that their result for complete graphs extend to all graphs.

Conjecture 7.1. [10] There exists a function f such that for every subdivision D of a graph G with at most one division vertex per edge, $\mathsf{sn}(G) \leq f(\mathsf{sn}(D))$.

In Lemma 7.11 we prove that sub-sn is bounded by sn. Thus the truth of Conjecture 7.1 would imply that stack-number is topological, and as we now show, would also imply an affirmative solution to Open Problem 1.1.

Theorem 7.2. *If Conjecture 7.1 is true then stack-number is bounded by queue-number.*

Proof. Conjecture 7.1 implies that there exists a function f^* such that for any s-stack subdivision of a graph G with at most k division vertices per edge, G has a $f^*(s,k)$-stack layout. In Theorem 7.9 we prove that every graph G has a 3-stack subdivision with $\mathcal{O}(\log \mathsf{qn}(G))$ division vertices per edge. Thus $\mathsf{sn}(G) \leq f^*(3, c \cdot \log \mathsf{qn}(G))$ for some constant c, and stack-number is bounded by queue-number. $\square$

In Section 7.2.2 we prove that queue-number is topological (for all graphs), and that track-number is topological for any proper minor-closed graph family. We now relate queue-number to a new thickness parameter. Let the 2-*track thickness* of a bipartite graph G, denoted by $\theta_2(G)$, be the minimum k such that G has a $(k,2)$-track layout. By (7.1) and Lemma 5.10(c),

$$\forall \text{ bipartite graphs } G,\ \theta(G) \leq \overline{\theta}(G) \leq \mathsf{sn}(G) \leq \theta_2(G)\ .$$

Note that Theorem 7.13 implies that for all $d \geq 2$, every graph G has a subdivision D with $\mathcal{O}(\log_d \mathsf{qn}(G))$ division vertices per edge, such that D has 2-track thickness $\theta_2(D) \leq d+1$. Let the 2-*track sub-thickness* of a graph G, denoted by $\mathsf{sub}\text{-}\theta_2(G)$, be the 2-track thickness of G'. This is well-defined since G' is bipartite. The first part of the next theorem follows from Lemmas 5.10 and 5.12, while we prove the second part in Section 7.2.1.

Theorem 7.3. (Section 7.2.1) *Queue-number is tied to 2-track thickness for bipartite graphs, and queue-number is tied to 2-track sub-thickness (for all graphs).*

The immediate implication of Theorem 7.3 for Open Problem 1.1 is that stack-number is bounded by queue-number if and only if stack-number is bounded by 2-track sub-thickness.

7.2 Small subdivisions

In this section we consider layouts of G' and G'', the subdivisions of a graph G with one and two division vertices per edge, respectively.

7.2.1 Track layouts

Lemma 7.1. *For every q-queue graph G, the subdivision G' has a $(q+1,2)$-track layout. That is, 2-track sub-thickness is bounded by queue-number. In particular,* $\mathsf{sub}\text{-}\theta_2(G) \leq \mathsf{qn}(G)+1$.

Proof. Let σ be the vertex ordering in a q-queue layout of G with queues $\{E_\ell : 1 \leq \ell \leq q\}$. Recall that $L(e)$ and $R(e)$ denote the left and right endpoints in σ of each edge e. Let $X(e)$ denote the division vertex of e in G'. Let $\prec$ be the total order on $\{X(e) : e \in E(G)\}$ such that $X(e) \prec X(f)$ whenever $L(e) <_\sigma L(f)$, or $L(e) = L(f)$ and $R(e) <_\sigma R(f)$. $(V(G),\sigma)$ and $(\{X(e) : e \in E(G)\},\prec)$ define a 2-track assignment of G'. Colour the edges of G' as follows. For all edges $e \in E_\ell$, let $\mathsf{col}(L(e)X(e)) = 0$ and $\mathsf{col}(X(e)R(e)) = \ell$. Since in $\prec$, division vertices are ordered primarily by the left endpoint of the corresponding edge, no two edges $L(e)X(e)$ and $L(f)X(f)$ form an X-crossing. Suppose $e' = X(e)R(e)$ and $f' = X(f)R(f)$ form an X-crossing. Without loss of generality $R(e) <_\sigma R(f)$ and $X(f) \prec X(e)$. By construction $L(f) <_\sigma L(e)$, and e is nested inside f in σ. Thus e and f are in distinct queues, and $\mathsf{col}(e') \neq \mathsf{col}(f')$. Hence there is no monochromatic X-crossing. The number of edge colours is $q+1$. Therefore we have a $(q+1,2)$-track layout of G'. □

Lemma 7.1 is best possible in the following (weak) sense. Let G be a 2-queue subdivision of a non-planar graph, which exists by Theorem 7.5. If G' has a $(k,2)$-track layout, then $k \geq 3$ since G' is non-planar, and by Theorem 7.21, only planar graphs have $(2,2)$-track layouts.

Open Problem 7.1. Is 2-track sub-thickness $\mathsf{sub}\text{-}\theta_2(G) \in o(\mathsf{qn}(G))$?

We have the following complimentary result to Lemma 7.1.

Lemma 7.2. *Queue-number is bounded by 2-track sub-thickness. In particular,* $\mathsf{qn}(G) \leq \mathsf{sub}\text{-}\theta_2(G)^2$ *for every graph G.*

Proof. Let $k = \mathsf{sub}\text{-}\theta_2(G)$. Clearly we can assume that G is connected. Thus in the given $(k,2)$-track layout of G', the vertices of G are on one track and the division vertices are on the other track. Let σ be the ordering of the vertices of G on the first track. Let $\{e_1, e_2, \dots, e_q\}$ be a maximum rainbow in σ, where e_i is nested inside e_{i+1} for all $i < q$. Let x_i be the division vertex of G' corresponding to each edge e_i. By the Erdös-Szekeres Theorem [77], the permutation of $\{x_1, \dots, x_q\}$ in the second track has an increasing or decreasing subsequence (with respect to the indices) of at least $\sqrt{q}$ vertices. Depending on whether it is increasing or decreasing, the left or right endpoints of $e_1, \dots, e_q$ along with $x_1, \dots, x_q$ determine a set of at least $\sqrt{q}$ edges that are pairwise X-crossing in the $(k,2)$-track of G'. Thus $\sqrt{q} \leq k$ and $q \leq k^2$. By Lemma 2.2, σ admits a k^2-queue layout of G. Hence $\mathsf{qn}(G) \leq k^2$. □

Lemmas 7.1 and 7.2 imply that queue-number and bipartite sub-thickness are tied (the second part of Theorem 7.3 in Section 7.1).

Lemma 7.3. *Every c-colourable q-queue graph G has:*
(a) $\mathsf{tn}_2(G') \leq q+1$, (b) $\mathsf{tn}(G') \leq c(q+1)$, *and* (c) $\mathsf{tn}(G'') \leq q+2$.

Proof. Let σ be the vertex ordering in a q-queue layout of G with queues $\{E_\ell : 1 \leq \ell \leq q\}$. Let $X(e)$ denote the division vertex of e in G'. Let $X_\ell = \{X(e) : e \in E_\ell\}$ for each $1 \leq \ell \leq q$. Let $<_\ell$ denote the queue order of each E_ℓ. Consider $<_\ell$ to also order X_ℓ. That is, for all edges $e, f \in E_\ell$,

$$X(e) \leq_\ell X(f) \iff L(e) \leq_\sigma L(f) \text{ and } R(e) \leq_\sigma R(f) \ . \tag{7.2}$$

First we prove (a). The set $\{(X_\ell, <_\ell) : 1 \leq \ell \leq q\} \cup \{(V(G), \sigma)\}$ defines a $(q+1)$-track assignment of G'. Colour edges $L(e)X(e)$ of G' blue, and colour edges $R(e)X(e)$ of G' red.

We claim that there is no monochromatic X-crossing. All edges of G' are between a vertex of G and a division vertex. Thus an X-crossing must involve two division vertices on the same track. Consider two edges e and f with $X(e) <_\ell X(f)$ for some $1 \leq \ell \leq q$. By (7.2), each of the pairs of edges $\{L(e)X(e), L(f)X(f)\}$ and $\{R(e)X(e), R(f)X(f)\}$ do not form an X-crossing. For each pair of edges $\{L(e)X(e), R(f)X(f)\}$ and $\{R(e)X(e), L(f)X(f)\}$ the edges are coloured differently. Thus there is no monochromatic X-crossing and we have a $(2, q+1)$-track layout of G'.

Now we prove (b). Let $\{V_i : 1 \leq i \leq c\}$ be a vertex c-colouring of G. Let $X_{i,\ell} = \{X(e) : e \in E_\ell,\, L(e) \in V_i\}$ for all $1 \leq \ell \leq q$ and $1 \leq i \leq c$. Thus $\{(X_{i,\ell}, <_\ell) : 1 \leq i \leq c,\, 1 \leq \ell \leq q\} \cup \{(V_i, <_\sigma) : 1 \leq i \leq c\}$ defines a $(qc+c)$-track assignment of G'. Consider division vertices $X(e), X(f) \in X_{i,\ell}$ such that $X(e) <_\ell X(f)$. By (7.2), $L(e) \leq L(f)$ in the ordering on V_i. Thus the pair of edges $\{L(e)X(e), L(f)X(f)\}$ do not form an X-crossing. Since neither $R(e)$ and $R(f)$ are in V_i, the pairs of edges $\{L(e)X(e), R(f)X(f)\}$ and $\{R(e)X(e), L(f)X(f)\}$ do not form an X-crossing. If both $R(e)$ and $R(f)$ are in the same colour class V_j, then $R(e) \leq_j R(f)$ by (7.2), and the pair of edges $\{R(e)X(e), R(f)X(f)\}$ do not form an X-crossing. Thus we have a $(qc+c)$-track layout of G'.

Finally we prove (c). Let $(L(e), X(e), Y(e), R(e))$ be the path replacing each edge e in G''. The first track consists of $\{(V(G), \sigma)\}$. The second track consists of $\{X(e) : e \in E(G)\}$, ordered so that $X(e) < X(f)$ whenever $L(e) <_\sigma L(f)$, or $L(e) = L(f)$ and $R(e) <_\sigma R(f)$. Edges between the first and the second track are of the form $L(e)X(e)$. Since vertices $X(e)$ in the second track are primarily ordered by $L(e)$, there is no X-crossing between the first and second track. Now define and order Y_ℓ as with X_ℓ. Then $\{(Y_\ell, <_\ell) : 1 \leq \ell \leq q\}$ comprises the final q tracks. An X-crossing involving vertices on these tracks can only be between pairs of edges $\{X(e)Y(e), X(f)Y(f)\}$ and $\{Y(e)R(e), Y(f)R(f)\}$, where e and f are in the same queue. By (7.2), such pairs of edges do not form an X-crossing. Thus we have $(q+2)$-track layout of G''. $\square$

We now describe how to produce a track layout of G' given a track layout of a graph G.

Lemma 7.4. *Let G be a (k,t)-track graph with maximum span s $(\leq t-1)$. Then G' has:* (a) $\mathsf{tn}_{ks+1}(G') \leq 2$, (b) $\mathsf{tn}_k(G') \leq 2t-1$, *and* (c) $\mathsf{tn}(G') \leq k(t-1)+t$.

Proof. Let $\{V_i : 1 \leq i \leq t\}$ be a (k,t)-track layout of G with span s. Let $\{E_\ell : 1 \leq \ell \leq k\}$ be the corresponding edge-colouring. By Lemma 5.14, G has a ks-queue layout. By Lemma 7.3(a), G' has a $(ks+1, 2)$-track layout. This proves part (a).

For each edge vw of G, let both edges in G' corresponding to vw be coloured by the colour assigned to vw. Now we prove part (b). For each $1 \leq i \leq t-1$, let $X_i \subseteq V(G') \setminus V(G)$ be the set consisting of the division vertices of edges $vw \in E(G)$ such that $v \in V_i$, $w \in V_j$, and $i < j$. Order the vertices in X_i with respect to the order of the corresponding vertices in V_i, breaking ties by the order in some V_j where applicable. Clearly there is no monochromatic X-crossing, where vertices of $G \setminus G'$ remain in the given track layout. The number of tracks is $2t-1$.

Finally we prove part (c). For each $1 \leq i \leq t-1$ and $1 \leq \ell \leq k$, let $X_{i,\ell} \subseteq X_i$ be the set consisting of the division vertices of edges $vw \in E_\ell$ such that $v \in V_i$, $w \in V_j$, and $i < j$. Order each $X_{i,\ell}$ as in X_i. All edges of G' incident to a vertex in $X_{i,\ell}$ are monochromatic. Thus there is no X-crossing regardless of the edge colours. The number of tracks is $t + k(t-1)$. □

We now describe how to produce a track layout of a graph G given a track layout of G'.

Lemma 7.5. *If a graph G is vertex c-colourable and $\mathsf{tn}_k(G') \leq t$ then $\mathsf{tn}_{tk^2}(G) \leq ct$.*

Proof. Let $\{V_i : 1 \leq i \leq c\}$ be a vertex c-colouring of G, and for each vertex $v \in V(G)$, let $\mathsf{col}(v) = i$ where $v \in V_i$. Let $\{(W_j, <_j) : 1 \leq j \leq t\}$ be a (k,t)-track layout of G' with edge colouring $\{E_\ell : 1 \leq \ell \leq k\}$. Let $V_{i,j} = V_i \cap W_j$ for each $1 \leq i \leq c$ and $1 \leq j \leq t$. Then $\{(V_{i,j}, <_j) : 1 \leq i \leq c, 1 \leq j \leq t\}$ is a track assignment of G. We now colour each edge vw of G. Without loss of generality $\mathsf{col}(v) < \mathsf{col}(w)$. Let x be the division vertex of vw in G', and say $x \in W_j$, $vx \in E_{\ell_1}$, and $wx \in E_{\ell_2}$. Then colour vw by the ordered triple (j, ℓ_1, ℓ_2). Note that the number of edge colours is tk^2. We claim that there is no monochromatic X-crossing in the track assignment of G. Suppose for the sake of contradiction, that there are monochromatic edges vw and pq in G that form an X-crossing. Without loss of generality, $\mathsf{col}(v) = \mathsf{col}(p) < \mathsf{col}(w) = \mathsf{col}(q)$, and in the given track layout of G', $v <_{j_1} p$ and $q <_{j_2} w$ for some $1 \leq j_1, j_2 \leq t$. Let x and y be the division vertices of vw and pq, respectively. Since vw and pq are monochromatic, x and y are in the same track W_{j_3}. If $x <_{j_3} y$ then wx and qy form a monochromatic X-crossing in the given track layout, and if $y <_{j_3} x$ then vx and py form a monochromatic X-crossing in the given track layout. In both cases we have the desired contradiction. Thus there is no monochromatic X-crossing in the track assignment of G, and we have a (tk^2, ct)-track layout of G. □

Lemma 7.6. *Let G be a graph with chromatic number $\chi(G) \leq c$ and star chromatic number $\chi_{\mathsf{st}}(G) \leq d$. If G' has a (k,t)-track layout then G has track-number $\mathsf{tn}(G) \leq d(2(ct-1)tk^2 + 1)^{d-1}$.*

Proof. By Lemma 7.5, G has a (tk^2, ct)-track layout. By Corollary 5.2 with span $s = ct - 1$, G has track-number $\mathsf{tn}(G) \leq d(2(ct-1)tk^2+1)^{d-1}$. □

Lemmas 2.1, 7.6 and 7.4(c) imply:

Theorem 7.4. *Track-number is topological for any proper minor-closed family.* □

7.2.2 Queue layouts

In this section we study the relationship between the queue-number of a graph G and the queue-number of G'. First note that Lemmas 5.14 and 7.1 imply:

Lemma 7.7. *The subdivision G' of a q-queue graph G has a $(q+1)$-queue layout.* $\square$

Lemmas 5.12 and 7.2 imply that if G' has a q-queue layout then G has a $4q^2$-queue layout. This bound can be improved as follows.

Lemma 7.8. *For every graph G, if G' has a q-queue layout with vertex ordering σ, then σ restricted to $V(G)$ admits a $q(2q+1)$-queue layout of G.*

Proof. Let X be the set of division vertices of G'. The vertex sets $V(G)$ and X partition $V(G')$ and define a vertex 2-colouring of G'. By Lemma 5.12, G' has a $(2q,2)$-track layout with tracks $(V(G),\sigma)$ and (X,σ). Let $1 \leq \mathsf{col}(e) \leq 2q$ be the colour assigned to each edge e of G'. Consider a layout of G in which the vertices are ordered by σ and the edges are partitioned into queues as follows. For each edge $vw \in E(G)$ divided by vertex x in G', let $\mathsf{queue}(vw) = \{\mathsf{col}(vx), \mathsf{col}(wx)\}$. We now prove that this layout is a queue layout of G. Say vw is nested inside ab in G and without loss of generality $a <_\sigma v <_\sigma w <_\sigma b$. Let vw be divided by x in G', and let ab be divided by c in G'. First suppose that $x <_\sigma c$. Then each of xw and xv form an X-crossing with ac. Thus $\mathsf{col}(xw) \neq \mathsf{col}(ac)$ and $\mathsf{col}(xv) \neq \mathsf{col}(ac)$. Hence $\mathsf{queue}(vw) \neq \mathsf{queue}(ab)$. Now suppose $c <_\sigma x$. Then bc forms an X-crossing with each of xw and xv. Thus $\mathsf{col}(bc) \neq \mathsf{col}(xw)$ and $\mathsf{col}(bc) \neq \mathsf{col}(xv)$. Hence $\mathsf{queue}(vw) \neq \mathsf{queue}(ab)$. The number of queues in the queue layout of G is $q(2q+1)$. $\square$

Lemmas 7.7 and 7.8 imply that queue-number is topological, as mentioned in Section 7.1.2. We now prove a slightly more general result than Lemma 7.8 that will be used in Section 7.3.3. Here we start with a subdivision with at most one division vertex per edge rather than exactly one division vertex per edge.

Lemma 7.9. *Let D be a q-queue subdivision of a graph G with at most one division vertex per edge. Then G has a $2q(q+1)$-queue layout.*

Proof. Let σ be the vertex ordering in a q-queue layout of D. Let A be the set of edges of G that are subdivided in D, and let B the set of edges of G that are not subdivided in D. By Lemma 7.8, $G[A]$ has a $q(2q+1)$-queue layout with vertex ordering σ. By assumption, $G[B]$ has a q-queue layout with vertex ordering σ. Thus G has a $2q(q+1)$-queue layout with vertex ordering σ. $\square$

7.2.3 Stack layouts

We now describe how to produce a stack layout of G' from a queue, stack or track layout of G. By Lemmas 7.1 and 5.10(c) we have:

Lemma 7.10. *The subdivision G' of a q-queue graph G has a $(q+1)$-stack layout. That is,* $\mathsf{sn}(G') \leq \mathsf{qn}(G) + 1$.

Lemma 7.11. *The subdivision G' of an s-stack graph G has an $(s+1)$-stack layout. That is,* $\mathsf{sn}(G') \leq \mathsf{sn}(G) + 1$.

Proof. Consider an s-stack layout of G with vertex ordering σ. Denote the division vertex of e in G' by $X(e)$. We now create a stack layout of G'. For each vertex v of G, let $e_1, e_2, \dots, e_d$ be all the edges incident to v such that each $L(e_i) = v$, and $R(e_d) <_\sigma R(e_{d-1}) <_\sigma \cdots <_\sigma R(e_1)$. Add the division vertices $X(e_1), X(e_2), \dots, X(e_d)$ immediately to the right of v in this order. Clearly for all edges e and f of G, the edges $L(e)X(e)$ and $L(f)X(f)$ of G' do not cross. Thus all these 'left' edges can be assigned to a single stack. Each 'right' edge $X(e)R(e)$ of G' inherits the stack assigned to e in G. Clearly no two right edges in the same stack cross. Thus G' has a $(s+1)$-stack layout. $\square$

Lemma 7.12. *Let G be a (k,t)-track graph with maximum span s $(\leq t-1)$. Then the subdivision G' of G with one division vertex per edge has a $s(k+1)$-stack layout.*

Proof. Let $\{(V_i, <_i) : 1 \leq i \leq t\}$ be a (k,t)-track layout of G with maximum span s, and with edge colouring $\{E_\ell : 1 \leq \ell \leq k\}$. Denote by $L(e)$ and $R(e)$ the endpoints of each edge e of G where $L(e) \in V_i$ and $R(e) \in V_j$ with $i < j$. Denote by $X(e)$ the division vertex in G' of e. For each $1 \leq i \leq t-1$ and $1 \leq \alpha \leq s$, let

$$X_{i,\alpha} \;=\; \{X(e) \,:\, e \in E(G),\, L(e) \in V_i,\, R(e) \in V_{i+\alpha}\} \ .$$

Since the maximum span is s, every division vertex of G' is in some $X_{i,\alpha}$. Order each $X_{i,\alpha}$ such that for all $X(e), X(f) \in X_{i,\alpha}$, we have $X(e) < X(f)$ whenever $L(f) <_i L(e)$, or $L(e) = L(f)$ and $R(f) <_{i+\alpha} R(e)$. Let σ be the vertex ordering of G' defined by

$$\big(V_1, X_{1,s}, X_{1,s-1}, \dots, X_{1,1};\, V_2, X_{2,s}, X_{2,s-1}, \dots, X_{2,1};\, \dots;\, V_t\big) \ .$$

Note that $L(e) <_\sigma X(e) <_\sigma R(e)$ for every edge e of G. For all $1 \leq \alpha \leq s$ let

$$E_\alpha = \{L(e)X(e) \,:\, L(e) \in V_i,\, X(e) \in X_{i,\alpha}\} \ .$$

For all $1 \leq \ell \leq k$ and $0 \leq \beta \leq s-1$, let

$$E_{\ell,\beta} = \{X(e)R(e) \,:\, e \in E_\ell,\, L(e) \in V_i,\, i \equiv \beta \pmod{s}\} \ .$$

This partitions the edges of G' into $s(k+1)$ sets. We claim that no two edges in a single set cross in σ. Consider two edges e and f of G. Say $L(e) \in V_{i_1}$ and $L(f) \in V_{i_2}$.

Consider edges $L(e)X(e)$ and $L(f)X(f)$ both in some E_α. Without loss of generality $i_1 \leq i_2$, and if $L(e) = L(f)$ then $R(e) <_\sigma R(f)$. If $i_1 < i_2$ then $L(e) <_\sigma X(e) <_\sigma L(f) <_\sigma X(f)$, and $L(e)X(e)$ and $L(f)X(f)$ do not cross. If $i_1 = i_2$ then without loss of generality $L(e) \leq_\sigma L(f)$. Since $L(e)X(e)$ and $L(f)X(f)$ are in E_α, both $X(e)$ and $X(f)$ are in $X_{i_1,\alpha}$. Thus $L(e) \leq_\sigma L(f) <_\sigma X(f) <_\sigma X(e)$, and $L(f)X(f)$ does not cross $L(e)X(e)$. Thus each set E_α is a valid stack in σ.

Now suppose the edges $X(e)R(e)$ and $X(f)R(f)$ cross in σ. Without loss of generality $X(e) <_\sigma X(f) <_\sigma R(e) <_\sigma R(f)$. Say $R(e) \in V_{i_3}$ and $R(f) \in V_{i_4}$. Then $i_1 \leq i_2 < i_3 \leq i_4$. If $i_1 < i_2$ then $i_2 - i_1 < i_3 - i_1 \leq s$. Thus $i_1 \not\equiv i_2 \pmod s$, and $X(e)R(e)$ and $X(f)R(f)$ are not in the same $E_{\ell,\beta}$. Now suppose $i_1 = i_2$. Since $X(e) <_\sigma X(f)$, we have $i_3 = i_4$ and $L(f) \leq_{i_1} L(e)$. If $L(f) = L(e)$ then, since $X(e) <_\sigma X(f)$ we have $R(f) <_{i_3} R(e)$, and thus $R(f) <_\sigma R(e)$, a contradiction. If $L(f) <_{i_1} L(e)$ then $R(e) <_{i_3} R(f)$ since $R(e) <_\sigma R(f)$. That is, e and f form an X-crossing in the track layout, and are thus coloured differently. Hence $X(e)R(e)$ and $X(f)R(f)$ are not in the same $E_{\ell,\beta}$.

Thus each E_α and each $E_{\ell,\beta}$ is a valid stack, and G' has a $s(k+1)$-stack layout. □

7.3 Big subdivisions

In this section we prove the main results summarized in Table 7.1. That is, every graph G has a 3-stack subdivision, a 2-queue subdivision, a mixed 1-stack 1-queue subdivision, and a 4-track subdivision. In each case the number of division vertices per edge is $\mathcal{O}(\log \mathsf{sn}(G))$ or $\mathcal{O}(\log \mathsf{qn}(G))$. First of all we introduce the notion of a (k, H)-layout.

Let G and H be graphs. H is called a *host graph*, and its vertices are called *nodes*. An *H-partition of G* is a partition $\{H_x \subseteq V(G) : x \in V(H)\}$ of $V(G)$ into *bags* indexed by the nodes of H such that for all edges $vw \in E(G)$ either:

- $\exists$ node $x \in V(H)$ such that both $v, w \in H_x$ (vw is called an *intrabag edge mapped* to x), or

- $\exists$ edge $xy \in E(H)$ such that $v \in H_x$ and $w \in H_y$ (vw is called an *interbag edge mapped* to xy).

Recall that tree-partitions, that is a T-partition for some tree T, were instrumental in the results presented in Chapter 6. To obtain our main results for layouts of subdivisions we employ the following very general structure. A *(k, H)-layout of G* is a pair $(\{E_1, E_2, \dots, E_k\}, \{(H_x, <_x) : x \in V(H)\})$ such that:

- $\{H_x \subseteq V(G) : x \in V(H)\}$ is an H-partition of G.

- $\forall$ nodes $x \in V(H)$, $<_x$ is a total order on H_x.

- $\{E_1, \dots, E_k\}$ is a colouring of the interbag edges such that there is no monochromatic X-*crossing*, where an X-crossing consists of a pair of interbag edges vw and pq such that for some edge $xy \in E(H)$, $v <_x p$ and $q <_y w$.

For each edge $xy \in E(H)$, let k_{xy} denote the number of colours used in the edge colouring of the interbag edges of G that are mapped to xy. For each node $x \in V(H)$, let s_x denote the minimum number of stacks such that $<_x$ admits an s_x-stack layout of $G[H_x]$, and let q_x denote the minimum number of queues such that $<_x$ admits a q_x-queue layout of $G[H_x]$.

We say a (k, T)-layout of G is *simple* if T is a rooted tree, and for every non-leaf node $x \in V(T)$, the set T_x is an independent set of G. Thus for simple layouts, $q_x = s_x = 0$ for all non-leaf nodes. A (k, H)-layout with no intrabag edges is called a (k, H)-*track layout*. A $(1, H)$-track layout is called an H-*track layout*. Observe that a (k, K_t)-track layout is simply a (k, t)-*track layout* as defined in Section 2.2.3.

Our main results are proved using the following strategy. First a particular host tree T (or tree-like graph T) is defined. The vertices of our graph G are mapped to the root of T, and each edge vw of G is mapped to some node of T. At each non-root node of T on the path from the root to the node that vw is mapped to, we add two division vertices to vw. This process produces a (k, T)-layout of a subdivision D of G, as is described in Section 7.3.1. Then a stack, queue, mixed or track-layout of T is determined, as described in Section 5.4. Then in Section 7.3.2 we describe how to transform a given layout of T into the desired layout of D. This process is then carried out for queue, stack, mixed, and track layouts in Sections 7.3.3–7.3.6.

7.3.1 (k, T)-Layouts

Recall that we assume that rooted trees are oriented away from the root node and that when referring to the edges of a directed graph, xy means an edge oriented from x to y.

Lemma 7.13. *Let T be the tree comprised of a root node r and $d \geq 1$ leaves $v_1, v_2, \dots, v_d$ adjacent to r. Suppose that the nodes of T are labeled with non-negative integers $l(r), l(v_1), l(v_2), \dots, l(v_d)$. Let G be a graph with a k-queue (respectively, k-stack) layout with vertex ordering σ, where $k \leq l(r) + l(v_1) + l(v_2) + \cdots + l(v_d)$. Then G has a subdivision D with zero or two division vertices per edge such that D has a $(1, T)$-layout in which the division vertices are mapped to the leaves of T, and the original vertices are mapped to the root r and are ordered by σ. Furthermore, every node $x \in V(T)$ has $q_x \leq l(x)$ ($s_x \leq l(x)$).*

Proof. Say $\sigma = (v_1, v_2, \dots, v_n)$. Let l be an integer such that $k - l \leq l(v_1) + l(v_2) + \cdots + l(v_d)$. Let F be the set of edges of G in an arbitrary set of l queues (stacks). Subdivide every edge $e = vw \in E(G) \setminus F$ twice, and denote the resulting path (v, e_v, e_w, w). This defines a subdivision D of G with zero or two division vertices per edge. For each vertex $v \in V(G)$, let $N^+(v) = \{e_v : e \in E(G) \setminus F, v = L_\sigma(e)\}$ and $N^-(v) = \{e_v : e \in E(G) \setminus F, v = R_\sigma(e)\}$. Order the vertices of $N^+(v)$ and $N^-(v)$ with respect to the order of the neighbours of v in σ. In the case of a given queue layout, let π be the vertex ordering of $V(D) \setminus V(G)$ defined by

$$\pi = \left(N^+(v_1), N^-(v_2), N^+(v_2), N^-(v_3), N^+(v_3), \ldots, N^-(v_{n-1}), N^+(v_{n-1}), N^-(v_n) \right) .$$

For a given stack layout, let π be the vertex ordering of $V(D) \setminus V(G)$ defined by

$$\pi = \left(\overleftarrow{N^+(v_1)}, \overleftarrow{N^-(v_2)}, \overleftarrow{N^+(v_2)}, \overleftarrow{N^-(v_3)}, \overleftarrow{N^+(v_3)}, \ldots, \overleftarrow{N^-(v_{n-1})}, \overleftarrow{N^+(v_{n-1})}, \overleftarrow{N^-(v_n)} \right) .$$

Partition the remaining $k - l$ queues (stacks) of G into sets $A_1, A_2, \ldots, A_d$ so that each A_i has at most $l(v_i)$ queues (stacks). Create $(1, T)$-layout of D as follows. Map the original vertices ordered by σ to r. By construction, the intrabag edge F of D mapped to r form l queues (stacks) with respect to σ. Thus $q_r \leq l$ $(s_r \leq l)$. For each edge $vw \in E(G) \setminus F$ that is in a queue (stack) in A_i, map e_v and e_w to v_i. Order each bag T_{v_i} by π. Since π is ordered primarily with respect to σ, there is no X-crossing in the layout. That is, we have a $(1, T)$-layout of D. In this layout, the edges $e_v e_w$ of D are intrabag edges mapped to the leaves of T. Consider each such edge $e_v e_w$ to be assigned to the same queue (stack) as vw in the given layout of G. Consider two edges $e = vw$ and $f = xy$ in $E(G) \setminus F$ that have no common endpoint. Since π is ordered primarily with respect to σ, the edges $e_v e_w$ and $f_x f_y$ nest/cross in π if and only if e and f nest/cross in σ. Now consider two edges $e = vx$ and $f = vy$ in $E(G) \setminus F$ (that have a common endpoint). In the case of queues, $e_v e_x$ and $f_v f_y$ are either crossing or disjoint. For stacks, $e_v e_x$ and $f_v f_y$ are either nested or disjoint. Thus the queue (stack) assignment for intrabag edges is valid, and $q_{v_i} \leq l(v_i)$ $(s_{v_i} \leq l(v_i))$ for each $1 \leq i \leq d$. □

For the next result we will need the following construction. Let G be a graph with a (k_1, T_1)-layout for some tree T_1. Let x be a node of T_1, and suppose that the subgraph $G[T_{1_x}]$ has a subdivision D_x where D_x has a (k_2, T_2)-layout, for some tree T_2 such that all the original vertices of D_x are mapped to the root r of T_2 ordered as per $<_x$. Let *merge-at-*x be a binary operation on the layouts (k_2, T_2) and (k_1, T_1) defined as follows. First replace $(T_x, <_x)$ by $(T_r, <_r)$, and rename x to y. Delete r from T_2 and make its children point to y. Each node $z \neq y$ in the new tree T_3 inherits $(T_z, <_z)$ from the node it originated from. It follows from the definition that merging (k_2, T_2) and (k_1, T_2) at x results in a (k_3, T_3)-layout of the subdivision D of G where $k_3 \leq \max\{k_1, k_2\}$ and where $q_y = q_r$ $(s_y = s_r)$, and each node $z \neq y$ in $V(T_3)$ has q_z (s_z) equal to that of the node it originated from.

Lemma 7.14. *Let T be a rooted tree of height h. Suppose that each node $x \in V(T)$ is labeled by a non-negative integer $l(v)$ such that $\sum_{v \in V(T)} l(v) \geq k$. Let G be a k-queue (respectively, k-stack) graph. Then G has a subdivision D with an even number of division vertices per edge, such that D has a $(1,T)$-layout in which every node $x \in V(T)$ has $q_x \leq l(x)$ $(s_x \leq l(x))$. Every edge of G has at most $2h$ division vertices in D, and if all the non-leaf nodes of T are labeled 0 and if all its leaves are at depth h, then every edge of G has exactly $2h$ division vertices in D.*

Proof. We proceed by induction on h. If $h = 0$ then the result follows trivially. Assume the result holds for all trees with height less than h, and let T be a tree of height h rooted at r. Let T' be the subtree of T induced by the nodes at depth at most $h-1$. Define a labeling on the nodes of T' as follows. For each node $v \in V(T')$ at depth $h-1$, let $l'(v) = l(v) + l(v_1) + l(v_2) + \cdots + l(v_d)$ where $v_1, v_2, \ldots, v_d$ are the children of v in T. For all remaining nodes $v \in V(T')$, let $l'(v) = l(v)$. Now $\sum_{v \in V(T')} l'(v) = \sum_{v \in V(T)} l(v) \geq k$. Thus by induction, G has a subdivision D' with at most $2(h-1)$ division vertices per edge, and D has a $(1,T')$-layout such that $q_x \leq l'(x)$ $(s_x \leq l'(x))$ for all nodes $x \in V(T')$. For each node $x \in V(T)$ at depth $h-1$, let $T(x)$ denote the subtree of T induced by x and its children, and let each node of $T(x)$ inherit its label from T. For every leaf node $x \in V(T')$ at depth $h-1$, apply Lemma 7.13 to the $l'(x)$-queue (stack) layout $(D'[T'_x], <_x)$ and the labeled tree $T(x)$. Merging(-at-x) the resulting $(1,T(x))$-layout of $D'[T'_x]$ with the $(1,T')$-layout of D' (for every leaf node x) gives rise to the desired $(1,T)$-layout of a subdivision D of G. Since only the intrabag edges in the leaf nodes of T' are subdivided and they are subdivided either zero or two times, D is a subdivision of G with an even number of division vertices per edge. Moreover, D has at most $2h$ division vertices per edge. The final claim of the lemma is immediate from the construction. Figure 7.1 illustrates the main concepts of the proof. □

For all integers $d_1, d_2 > 0$, a *complete* (d_1, d_2)*-ary tree* is a rooted tree in which all the leaves are at the same depth, every non-leaf node at even depth has outdegree d_1 and every non-leaf node at odd depth has outdegree d_2. If $d_1 = d_2 = d$ then we speak of a *complete* d*-ary tree*. The following special case of Lemma 7.14 will be useful.

Lemma 7.15. *Let T be a subdivision of the complete (d_1, d_2)-ary tree of height h. Let h' be the height of T. Let $\alpha = (d_1)^{\lceil h/2 \rceil}(d_2)^{\lfloor h/2 \rfloor}$. Then every k-queue (respectively, k-stack) graph G has a subdivision D with an even number of division vertices per edge, and D has a simple $(1,T)$-layout in which $q_x \leq \lceil k/\alpha \rceil$ $(s_x \leq \lceil k/\alpha \rceil)$ for every node $x \in V(T)$. Moreover, the number of division vertices per edge is at most $2h'$, or exactly $2h'$ if all the leaves of T are at depth h'.*

Proof. Let $l(x) = 0$ for each non-leaf node $x \in V(T)$. Let $l(x) = \lceil k/\alpha \rceil$ for each leaf node $x \in V(T)$. The number of leaves in the complete (d_1, d_2)-ary tree of height h is α.

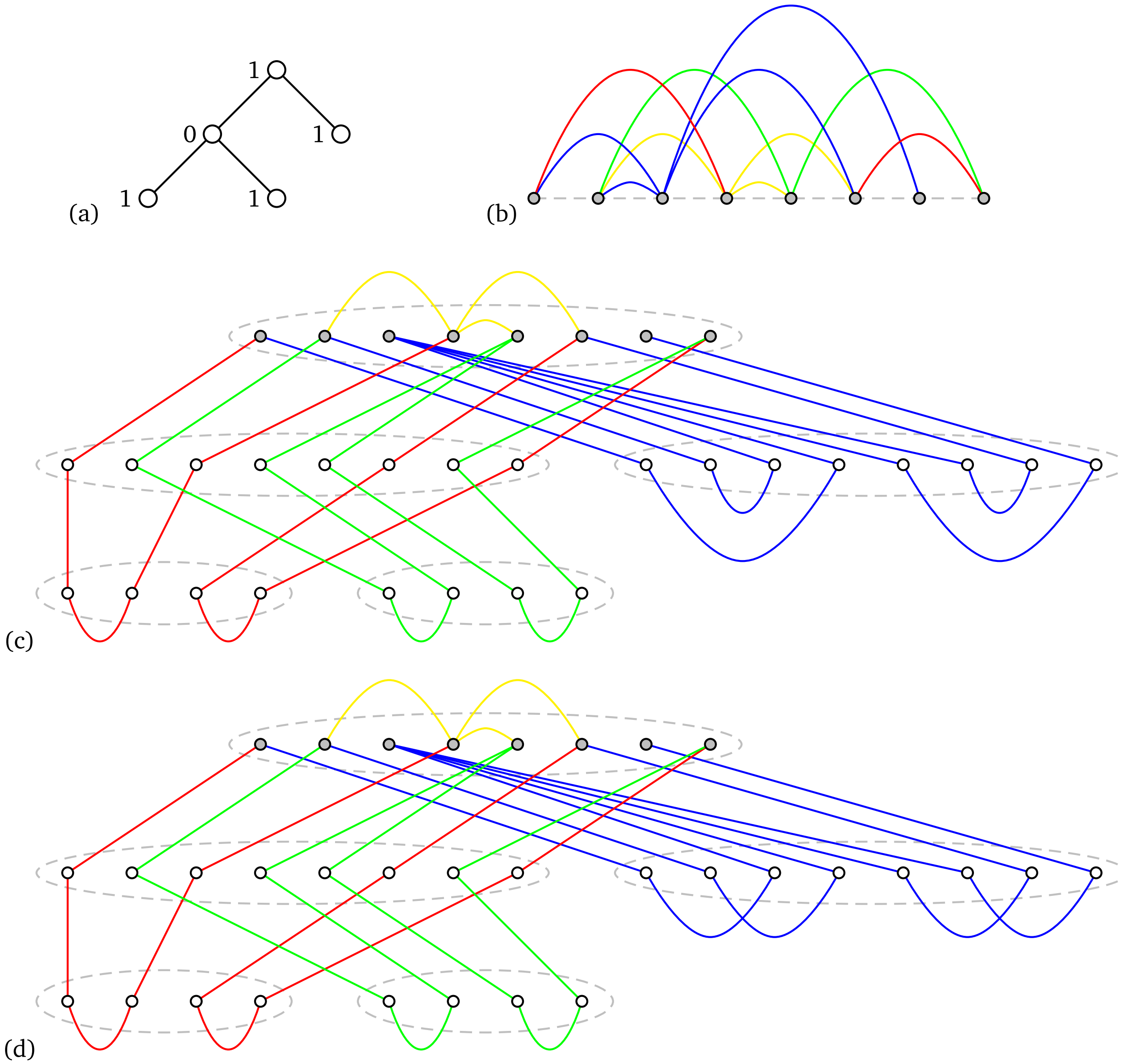


FIGURE 7.1: Illustration for Lemma 7.14. Given (a) a labelled tree T and (b) a 4-stack layout of G (that is also a 4-queue layout), the algorithm produces a $(1, T)$-layout of a subdivision of G with (c) $s_x \leq l(x)$ or (d) $q_x \leq l(x)$.

Subdividing the edges of a tree does not change the number of leaves. Thus T also has α leaves. Therefore $\sum_{x \in V(T)} l(x) \geq k$. Since the non-leaf nodes are labeled 0, by Lemma 7.14, G has a subdivision D with a $(1, T)$-layout such that for each leaf node $x \in V(T)$, $q_x \leq l(x) = \lceil k/\alpha \rceil$ ($s_x \leq l(x) = \lceil k/\alpha \rceil$), and for each non-leaf node $x \in V(T)$, $q_x \leq l(x) = 0$ ($s_x \leq l(x) = 0$). Thus the $(1, T)$-layout is simple. The claim about the number of division vertices per edge follows immediately from Lemma 7.14. □

7.3.2 (k, H)-Layout into layout of G

For a graph G with a (k, H)-layout, we now show how to convert a layout of H into a layout of G. First consider a (k, T)-layout in which T is a rooted tree. We will often define a 2-colouring of the edges of T using colours red and black. The edges of G mapped to red edges of T will be associated with stacks, and those mapped to black edges of T will be associated with queues. Let $E^r(T)$ and $E^b(T)$ denote the sets of red and black edges of T.

Lemma 7.16. *Let G be a graph with a (k, T)-layout for some rooted tree T. Suppose that each edge and node of T is coloured red or black such that T has a topological vertex ordering σ where the red edges form a stack and the black edges form a queue. For each node $x \in V(T)$, let $s'_x = s_x$ if x is red, and $s'_x = 0$ if x is black. Similarly, let $q'_x = q_x$ if x is black, and $q'_x = 0$ if x is red. Let*

$$\lambda_s = \max_{x \in V(T)} \left\{ s'_x + \sum_{xy \in E^r(T)} k_{xy} + \sum_{yx \in E^r(T)} k_{yx} \right\} ,$$

and

$$\lambda_q = \max_{x \in V(T)} \left\{ q'_x + \max_{y \in V(T)\,:\, y \leq_\sigma x} \sum_{yz \in E^b(T)\,:\, x \leq_\sigma z} k_{yz} \right\} .$$

Then G has an λ_s-stack λ_q-queue mixed layout, such that the edges of G that are mapped to red nodes or edges of T are in stacks, and the edges of G that are mapped to black nodes or edges of T are in queues.

Proof. First we label the nodes of T as *forward* or *backward*. Consider the nodes of T in the order of their appearance in σ. Label the root node as forward or backward arbitrarily. Now consider a non-root node x with incoming edge yx. Since σ is topological, y has already been labelled. If yx is black then label x with the same label as that given to y. If yx is red then label x with the opposite label to that given to y. Now create a vertex ordering π of G by replacing each node x in σ by T_x if x is forward, and by $\overleftarrow{T_x}$ if x is backward. (Recall that $\overleftarrow{T_x}$ denotes the reverse ordering of T_x to that in the given (k, T)-layout.)

Let $E^r(G)$ and $E^b(G)$ denote the sets of edges of G that are mapped to red edges/nodes and black edges/nodes of T, respectively. We first prove that there is an edge λ_q-colouring of $E^b(G)$ such that no two monochromatic edges in $E^b(G)$ are nested in π.

Let R be a rainbow in π formed from by the edges of $E^b(G)$ and with the maximum number of edges. Let the set of intrabag edges in R be denoted by R_{intra}, and the set of interbag edges be denoted by R_{inter}. Then $|R| = |R_{\text{intra}}| + |R_{\text{inter}}|$. Suppose the left endpoint of the innermost edge of R is mapped to node x. Then the right endpoint of each edge in R is mapped to a node z such that $x \leq_\sigma z$. Intrabag edges mapped to distinct nodes of T are not nested (and not crossing). Thus all the edges in R_{intra} are mapped to the same node of T. Hence all the edges of R_{intra} (if any) are mapped to x. Thus $|R_{\text{intra}}| \leq q'_x$. At least one of the endpoints of each edge in R_{inter} is not mapped to x. Thus by the construction of π, such endpoints appear in π either before or after all the endpoints of the edges in R_{intra}. Therefore the edges of R_{intra} are all nested inside the innermost edge of R_{inter}. Since the black edges in T are not nested in σ, all the edges of R_{inter} have an endpoint mapped to the same node $y \in T$. Since the edges in R_{intra} are nested inside the edges of R_{inter}, $y \leq_\sigma x$. Furthermore, since σ is a topological vertex-ordering of T, each edge of R_{inter} is mapped to some outgoing edge of y. If two edges of R_{inter} are mapped to the same edge incident to y, then by Lemma 5.10(b) they may be nested only if their edge colours in the (k,T)-layout are different. Therefore, $|R_{\text{inter}}| \leq \sum_{z \in V(T)\,:\, x \leq_\sigma z} k_{yz}$ and thus $|R| \leq q'_x + \sum_{z \in V(T)\,:\, x \leq_\sigma z} k_{yz}$. By considering all choices of x and $y \leq_\sigma x$ in $V(T)$, we conclude that a rainbow in π formed by the edges of $E^b(G)$ may have at most λ_q edges. By Lemma 2.2, the edges of $E^b(G)$ can be coloured with λ_q colours such that no two monochromatic edges are nested.

We now define an edge λ_s-colouring of $E^r(G)$. We then prove that no two monochromatic edges in $E^r(G)$ cross. Consider the nodes of T in the order of their appearance in σ. For each node x, colour the edges of G that are mapped to the red edges incident to x as follows. Two interbag edges of G that are mapped to the same outgoing red edge of x receive the same colour if and only if they belong to the same colour class $E_i \in \{E_1, E_2, \dots E_k\}$ in the (k,T)-layout of G. Two interbag edges of G mapped to two distinct red edges incident to x always receive distinct colours (regardless of whether they are incoming or outgoing). If x is red, colour the intrabag edges mapped to x with distinct colours to those used on the interbag edges mapped to the red edges incident to x, and so that $<_x$ admits an s_x-stack layout of $G[T_x]$. We now show that λ_s colours suffices for such a colouring. If the incoming edge yx of x is red the edges of G mapped to yx use k_{yx} colours out of λ_s colours, otherwise 0 out of λ_s colours are used. Thus we have either $\lambda_s - k_{yx}$ or λ_s colours available for colouring the edges of G mapped to x and the red outgoing edges $xy_1, xy_2, \dots, xy_p$ incident to x. Clearly we can colour the edges of G mapped to $xy_1, xy_2, \dots, xy_p$ and x as described above with $k_{xy_1} + k_{xy_2} + \cdots + k_{xy_p} + s'_x$ distinct colours. Thus the number of colours used is at most λ_s.

We now show that no two monochromatic edges $e_1, e_2 \in E^r(G)$ cross in π. That is, monochromatic edges in $E^r(G)$ can be in the same stack. From the description of the edge colouring, it is clear that if either e_1 or e_2 is an intrabag edge then the pair does not form a monochromatic crossing. Thus it suffices to consider pairs of interbag edges. Since the red edges in T are not crossing in σ, the only pairs of interbag edges that can create a monochromatic crossing are those with endpoints in the same bag T_x. In that case, if e_1 and e_2 are mapped to the same edge incident to x then e_1 and e_2 do not cross by Lemma 5.10(c). If e_1 and e_2 are mapped to two distinct edges incident to x then e_1 and e_2

are not monochromatic. □

Lemma 7.17. *Let H be a graph with a t-track layout $\{V_i : 1 \leq i \leq t\}$ such that each node in track V_i, $1 \leq i \leq t$, has at most one neighbour in each track V_j, $1 \leq j \leq i-1$. Let G be a graph with a (k,H)-track layout. Let*

$$p = \max_{x \in V(H)} \max_{1 \leq \ell \leq t} \sum_{xy \in E(H)\,:\,\mathsf{track}(y)=\ell} k_{xy}. \tag{7.3}$$

Then replacing each node x in the t-track layout of H by $(H_x, <_x)$ from the (k,H)-track layout of G, gives a (p,t)-track layout of G.

Proof. Define an edge colouring of G as follows. For each node x of T in V_i, and for each ℓ, $i < \ell \leq t$, consider the set of edges E_ℓ incident to x that have their other endpoint in V_ℓ. Colour the edges of G that are mapped to the edges of E_ℓ with p colours such that any two edges $e_1, e_2 \in E(G)$ receive the same colour if and only if they are mapped to the same edge $xy \in E_\ell$ and they belong to the same colour class in the (k,H)-layout of G. This is possible with at most p colours by (7.3).

We now prove that there are no monochromatic X-crossings with this edge p-colouring. Consider two monochromatic edges $e_1, e_2 \in E(G)$. If e_1 and e_2 are mapped to the same edge of H then by the above colouring procedure and by the properties of the edge colouring in the (k,H)-track layout of G, edges e_1 and e_2 do not form a monochromatic X-crossing. If e_1 and e_2 are mapped to two edges $xy, zq \in E(H)$ that have no endpoint in common, then e_1 and e_2 do not form a monochromatic X-crossing since xy and zq do not form a monochromatic X-crossing in the t-track layout of H. Finally, if e_1 and e_2 are mapped to two edges $xy, xz \in E(H)$ that share an endpoint x, then e_1 and e_2 can only form a monochromatic X-crossing if y and z are in the same track V_ℓ. Say $x \in V_i$. Since x has at most one neighbour in $V_1, V_2, \dots, V_{i-1}$, we have that $\ell > i$. Therefore, by the above colouring procedure e_1 and e_2 do not have the same colour. □

Lemma 7.18. *Let $d \geq 2$ be an integer. Let G be a graph with a simple $(1,T_0)$-layout for some tree T_0, such that every leaf node x has $q_x \leq 1$, and every non-leaf node x has $q_x = 0$ and $\deg^+(x) = d$. Then there is a tree T, such that the subdivision D obtained from G by subdividing each intrabag edge once has a $(2,T)$-track layout in which every node $x \in V(T)$ has*

$$\sum_{xy \in E(T)} k_{xy} + \sum_{yx \in E(T)} k_{yx} \leq d+1, \quad \text{and} \quad \sum_{xy \in E(T)} k_{xy} \leq d\ . \tag{7.4}$$

Proof. For every leaf node $x \in V(T_0)$, let D_x be the subdivision of $G[T_{0x}]$ obtained by subdividing each edge of $G[T_{0x}]$ once. By the proof of Lemma 7.1, D_x has a $(2,T^*)$-track layout where T^* is a single edge comprised of a root node adjacent to one leaf, such that all the original vertices of $G[T_{0x}]$ are mapped to the root and are ordered by $<_x$, and all the division vertices are mapped to the leaf node in T^*. For each leaf node $x \in V(T_0)$ merge-at-x the $(1, T_0)$-layout of G and the $(2, T^*)$-track layout of $G[T_{0x}]$. In the resulting $(2, T)$-layout of D there are no intrabag edges. Thus we have a $(2, T)$-track layout where T is the subdivision of T_0 with each leaf-edge of T_0 subdivided once. Let V_ℓ be the set of leaves in T. Let E_ℓ be the set of edges of T with an endpoint in V_ℓ. All the interbag edges of D that are mapped to the edges in $E \setminus E_\ell$ are coloured with one colour. All the interbag edges of D that are mapped to the edges in E_ℓ are coloured with at most two colours. Thus, each node $x \in V(T)$ that has no neighbour in V_ℓ satisfies (7.4). Each node $x \in V(T)$ that has a neighbour in V_ℓ has degree at most 2. Since the incoming edge yx of x has $k_{yx} \leq 1$ and its outgoing edge xv has $k_{xv} \leq 2$, x satisfies (7.4) since $d \geq 2$. Finally, each leaf node x has $k_{yx} \leq 2$ where yx is the incoming edge of x. Thus x satisfies (7.4) since $d \geq 2$. □

7.3.3 Queue layouts

Theorem 7.5. *For every integer $d \geq 2$, every graph G has a d-queue subdivision with $2\lceil \log_d \mathsf{qn}(G) \rceil + 1$ division vertices per edge.*

Proof. Let T_0 be the complete d-ary tree of height $h = \lceil \log_d \mathsf{qn}(G) \rceil$. By Lemma 7.15 with $d_1 = d_2 = d$, G has a subdivision D_0 with $2h$ division vertices per edge, such that D_0 has a simple $(1, T_0)$-layout in which every non-leaf node $x \in V(T_0)$ has $q_x = 0$, and every leaf node $x \in V(T_0)$ has $q_x \leq 1$. Let D be the subdivision of G obtained from D_0 by subdividing each intrabag edge (in the $(1, T_0)$-layout of D_0) once. Clearly D has $2\lceil \log_d \mathsf{qn}(G) \rceil + 1$ division vertices per edge of G. By Lemma 7.18 applied to D_0, there exists a tree T such that D has a $(2, T)$-track layout in which every node $x \in V(T)$ has

$$\sum_{xy \in E(T)} k_{xy} \;\leq\; d \;. \tag{7.5}$$

Let all the edges and nodes of T be coloured black. By Lemma 5.16, T has a topological vertex ordering σ that admits a 1-queue layout. By (7.5) and since every node x in T has $q_x = 0$, we have

$$\lambda_q \;=\; \max_{x \in V(T)} \left\{ q_x + \max_{y \in V(T)\,:\, y \leq_\sigma x} \sum_{yz \in E(T)\,:\, x \leq_\sigma z} k_{yz} \right\} \;\leq\; \max_{x \in V(T)} \left\{ \sum_{xv \in E(T)} k_{xv} \right\} \;\leq\; d \;. \tag{7.6}$$

Therefore, by Lemma 7.16, D has a d-queue layout, as illustrated in Figure 7.2 for $d = 2$. □

We now prove that the number of division vertices per edge in Theorem 7.5 is optimal

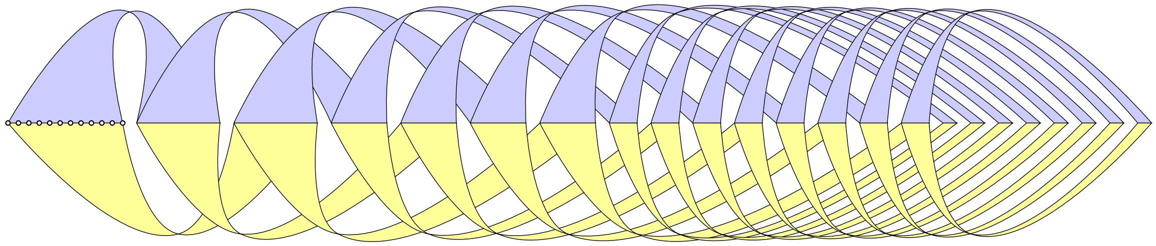


FIGURE 7.2: 2-queue subdivision of an 8-queue graph.

up to a constant factor.

Lemma 7.19. *Let D be a q-queue subdivision of a graph G with at most k division vertices per edge. Then G has a $(\frac{1}{2}(2q+2)^{2k}-1)$-queue layout.*

Proof. Let $q_i = \frac{1}{2}(2q+2)^{2^i} - 1$, and $k_i = k/2^i$. We proceed by induction on $i \geq 0$ with the hypothesis: *there exists a subdivision D_i of G with at most k_i division vertices per edge, and D_i has a q_i-queue layout*. Consider the base case with $i = 0$. Let $D_0 = D$. Then D_0 is a subdivision of G with $k_0 = k$ division vertices per edge, and D_0 has a q_0-queue layout, since $q_0 = q$.

Suppose that there exists a subdivision D_i of G with at most k_i division vertices per edge, and D_i has a q_i-queue layout. By contracting every second division vertex on the path representing each edge of G in D_i, we obtain a graph D_{i+1} such that D_i is a subdivision of D_{i+1} with at most one division vertex per edge, and D_{i+1} is a subdivision of G with at most $k_i/2$ division vertices per edge. By Lemma 7.9, D_{i+1} has a $2q_i(q_i+1)$-queue layout. Now $k_i/2 = k_{i+1}$, and $2q_i(q_i+1) \leq 2(q_i+1)^2 - 1 = \frac{1}{2}(2q+2)^{2^{i+1}} - 1 = q_{i+1}$. Thus the inductive hypothesis holds for all i.

With $i^* = \lfloor \log_2 k \rfloor + 1$, we have $k_{i^*} < 1$. The only subdivision of G with less than one division vertex per edge is G itself. Thus G has a q_{i^*}-queue layout, and $q_{i^*} = \frac{1}{2}(2q+2)^{(2^{\lfloor \log_2 k \rfloor + 1})} - 1 \leq \frac{1}{2}(2q+2)^{(2^{1+\log_2 k})} - 1 \leq \frac{1}{2}(2q+2)^{2k} - 1$. □

Theorem 7.6. *Let D be a d-queue subdivision of a graph G for some $d \geq 2$. Then there is an edge of G with at least $\frac{1}{6}\log_d \mathsf{qn}(G)$ division vertices in D.*

Proof. By Lemma 7.19, G has $(\frac{1}{2}(2d+2)^{2k}-1)$-queue layout. Thus $\mathsf{qn}(G) \leq \frac{1}{2}(2d+2)^{2k} - 1$, and $\mathsf{qn}(G) \leq \frac{1}{2}(3d)^{2k} - 1$ since $d \geq 2$. That is, $k \geq \frac{1}{2}\log_{3d} 2(\mathsf{qn}(G)+1) = \frac{1}{2}(\log_{3d} d)(\log_d 2(\mathsf{qn}(G)+1)) \geq \frac{1}{6}\log_d 2(\mathsf{qn}(G)+1)$ since $d \geq 2$. Therefore $k \geq \frac{1}{6}\log_d \mathsf{qn}(G)$, as claimed. Note that $\log_{3d} d \to 1$ for large d, and the lower bound on k tends to $\frac{1}{2}\log_d 2(\mathsf{qn}(G)+1)$. □

7.3.4 Stack layouts

Theorem 7.7. *For every integer $d \geq 2$, every graph G has a $(d+1)$-stack subdivision with $2\lceil \log_d \mathsf{sn}(G) \rceil - 2$ division vertices per edge.*

Proof. Let $k = \mathsf{sn}(G)$. Apply Lemma 7.15 with T the complete d-ary tree of height $h = \lceil \log_d k \rceil - 1$. Then $\alpha = d^{\lfloor h/2 \rfloor + \lceil h/2 \rceil} = d^h \geq d^{(\log_d k)-1} = k/d$. By Lemma 7.15, G has a subdivision D with $2h$ division vertices per edge, such that D has a simple $(1, T)$-layout in which every non-leaf node $x \in V(T)$ has $\deg^+(x) = d$ and $s_x = 0$, and every leaf node $x \in V(T)$ has $s_x \leq \lceil k/\alpha \rceil \leq d$. Let all the edges and nodes of T be coloured red. Define λ_s as in Lemma 7.16. That is, λ_s is the maximum, taken over all nodes $x \in V(T)$, of

$$s_x + \sum_{xy \in E(T)} k_{xy} + \sum_{yx \in E(T)} k_{yx} \tag{7.7}$$

For leaf nodes x, (7.7) is at most $d + 0 + 1 = d + 1$. For non-leaf nodes x, (7.7) is $0 + d + 1 = d + 1$. Thus $\lambda_s = d + 1$. By Lemma 5.17, T has a topological ordering that admits a 1-stack layout, and by Lemma 7.16, D has a $(d+1)$-stack layout. The stack layout of D is illustrated in Figure 7.3 for $d = 2$. □

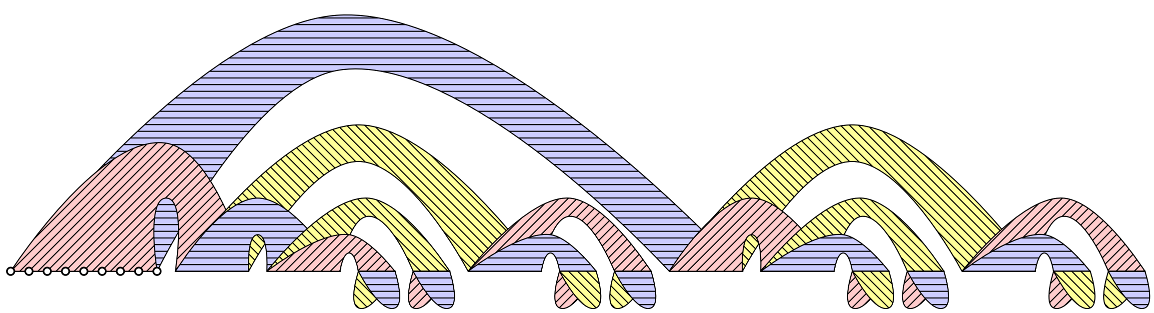


FIGURE 7.3: 3-stack subdivision of a 16-stack graph.

Recall that Open Problem 1.2 asks whether queue-number is bounded by stack-number. We make the following contribution to the study of this problem.

Theorem 7.8. *The following are equivalent:*

(1) *queue-number is bounded by stack-number,*

(2) *bipartite 3-stack graphs have bounded queue-number,*

(3) *bipartite 3-stack graphs have bounded 2-track thickness.*

Moreover, if queue-number is bounded by stack-number then queue-number is bounded by a polynomial function of stack-number.

Proof. That (1) implies (2) is immediate. Theorem 7.3 proves that (2) and (3) are equivalent. It remains to prove that (2) implies (1). Suppose that every bipartite 3-stack graph has queue-number at most some constant q. Consider an arbitrary graph G. By Lemma 7.11, G' has a $(\mathsf{sn}(G)+1)$-stack layout. Thus, by Theorem 7.7, G' has a 3-stack subdivision D with $2(2\lceil\log_2(\mathsf{sn}(G)+1)\rceil-2)+1 = 4\lceil\log_2(\mathsf{sn}(G)+1)\rceil-3$ division vertices per edge. That is, G has a 3-stack subdivision with $2\lceil\log_2(\mathsf{sn}(G)+1)\rceil-1$ division vertices per edge. Since each edge of G is subdivided an odd number of times, D is bipartite. By assumption, D has queue-number at most q. By Lemma 7.19, G has queue-number $\frac{1}{2}(2q+2)^{8\lceil\log_2(\mathsf{sn}(G)+1)\rceil-6}-1$. Since q is constant, queue-number is bounded by a polynomial function of stack-number. □

Theorem 7.9. *For every integer $d \geq 2$, every graph G has a $(d+1)$-stack subdivision with $1+2\lceil\log_d \mathsf{qn}(G)\rceil$ division vertices per edge.*

Proof. Apply Lemma 7.15 with $d_1 = d_2 = d$, $h = \lceil\log_d \mathsf{qn}(G)\rceil$, and T_0 a complete d-ary tree of height h. Then G has a subdivision D_0 with $2\lceil\log_d \mathsf{qn}(G)\rceil$ division vertices per edge such that D_0 has a simple $(1,T_0)$-layout in which every non-leaf node $x \in V(T)$ has $s_x = 0$, and every leaf node has $x \in V(T)$ has $q_x \leq 1$. Let D be the subdivision of G obtained from D_0 by subdividing each intrabag edge (in the $(1,T_0)$-layout of D_0) once. Clearly D has $2\lceil\log_d \mathsf{qn}(G)\rceil+1$ division vertices per edge of G. By Lemma 7.18, there exists a tree T such that D has a $(2,T)$-track layout in which every node $x \in V(T)$ has

$$\max_{x\in V(T)} \left\{ \sum_{xy\in E(T)} k_{xy} + \sum_{yx\in E(T)} k_{yx} \right\} \leq d+1 \enspace .$$

Colour all the edges and nodes of T red. Since every node $x \in V(T)$ has $s_x = 0$,

$$\max_{x\in V(T)} \left\{ s_x + \sum_{xy\in E(T)} k_{xy} + \sum_{yx\in E(T)} k_{yx} \right\} \leq d+1 \enspace .$$

By Lemma 5.17, T has a 1-stack layout, and by Lemma 7.16, D has a $(d+1)$-stack layout. □

7.3.5 Mixed layouts

Theorem 7.10. *For all integers $s \geq 1$ and $q \geq 1$, every graph G has an s-stack q-queue subdivision with $4\lceil\log_{(s+q)\,q} \mathsf{sn}(G)\rceil$ division vertices per edge.*

Proof. Apply Lemma 7.15 with $d_1 = s+q$, $d_2 = q$, $h = 2\lceil\log_{(s+q)q} \mathsf{sn}(G)\rceil$, and T a complete (d_1,d_2)-ary tree of height h. Then G has a subdivision D with $4\lceil\log_{(s+q)\,q} \mathsf{sn}(G)\rceil$ division vertices per edge, and D has a simple $(1,T)$-layout where $\max_{x\in V(T)}\{s_x\} \leq 1$ and where every node $v \in V(T)$ at even depth has $\deg^+(v) \leq s+q$ and every node $v \in V(T)$ at odd depth has $\deg^+(v) \leq q$. Colour the edges of T as follows. For each non-leaf node $v \in V(T)$ at even depth, colour its outgoing edges red or black so that at most s outgoing edges are red and at most q are black. For nodes $v \in V(T)$ at odd depth, colour the outgoing edges of v black. Clearly this edge colouring is good (see page 73 to recall the definition of good edge colouring). By Lemma 5.18, T has a topological ordering that admits a 1-queue layout of $T[E^b]$ and a 1-stack layout of $T[E^r]$.

Colour all the vertices of T red. Consequently, every node x in T has $q'_x = 0$. (See Lemma 7.16 to recall the definitions of q'_x and s'_x.) For each node $x \in V(T)$, let $\deg^+_{\text{black}}(x)$ denote the outdegree of x in $T[E^b]$. Define λ_s and λ_q as in Lemma 7.16. Then

$$\begin{aligned}
\lambda_q = & \max_{x \in V(T)} \left\{ q'_x + \max_{y \in V(T) \,:\, y \leq_\sigma x} \sum_{yz \in E^b(T) \,:\, x \leq_\sigma z} k_{yz} \right\} \\
\leq & \max_{x \in V(T)} \left\{ \sum_{xv \in E^b(T)} k_{xv} \right\} \\
\leq & \max_{x \in V(T)} \deg^+_{\text{black}}(x) \\
\leq & \; q \; .
\end{aligned}$$

By the properties of the simple $(1,T)$-layout of D every non-leaf node x of T has $s'_x = 0$ and every leaf node x of T has $s'_x \leq 1$. For a node x in T, let $\deg_{\text{red}}(x)$ denote the degree of x in $T[E^r]$. Since h is even, the height of T is even and thus all the edges incident to leaves of T are black. For every leaf node $x \in V(T)$ that implies that $\deg_{\text{red}}(x) = 0$. Therefore,

$$\begin{aligned}
\lambda_s = & \max_{x \in V(T)} \left\{ s'_x + \sum_{xy \in E^r(T)} k_{xy} + \sum_{yx \in E^r(T)} k_{yx} \right\} \\
\leq & \max \left\{ \max_{x \in V(T) \,:\, \deg(x) = 1} s'_x \; , \; \max_{x \in V(T) \,:\, \deg(x) \neq 1} \deg_{\text{red}}(x) \right\} \\
\leq & \; s \; .
\end{aligned}$$

By Lemma 7.16, the subdivision D of G has an s-stack q-queue mixed layout. □

Theorem 7.11. *For all $s \geq 1$ and $q \geq 1$, every graph G has an s-stack q-queue subdivision with $2 + 4\lceil \log_{(s+q)\,q} \mathsf{qn}(G) \rceil$ division vertices per edge.*

Proof. Apply Lemma 7.15 with $d_1 = s + q$, $d_2 = q$, $h = 2\lceil \log_{(s+q)q} \mathsf{qn}(G) \rceil$, and T a tree obtained from a complete (d_1, d_2)-ary tree of height h by subdividing each of its leaf-edges once. The height of T is $h+1$ and all of its leaves are at depth $h+1$. Then G has a subdivision D with $2+4\lceil \log_{(s+q)\,q} \mathsf{qn}(G) \rceil$ division vertices per edge, and D has a simple $(1,T)$-layout in which every non-leaf node $x \in V(T)$ has $q_x = 0$, and every leaf node $x \in V(T)$ has $q_x \leq 1$.

Colour the edges of T as follows. For each node $x \in V(T)$ at odd depth, colour all its outgoing edges black. For each node $x \in V(T)$ at even depth, if depth$(x) < h$ colour each of its outgoing edges red or black such that s are red and q are black, otherwise, depth$(x) = h$, colour its only outgoing edge red. Clearly this edge colouring of T is good. Thus by Lemma 5.18, has a topological vertex ordering, such that the black edges form a queue, and the red edges form a stack.

Colour all the vertices of T black. Consequently, every node $x \in V(T)$ has $s'_x = 0$. (See Lemma 7.16 to recall the definitions of q'_x and s'_x). For each node $x \in V(T)$, let $\text{deg}_{\text{red}}(x)$ denote the degree of x in $T[E^r]$. Define λ_s and λ_q as in Lemma 7.16. Then

$$\lambda_s = \max_{x \in V(T)} \left\{ s'_x + \sum_{xy \in E^r(T)} k_{xy} + \sum_{yx \in E^r(T)} k_{yx} \right\} \leq \max_{x \in V(T)} \{\text{deg}_{\text{red}}(x)\} \leq s \ .$$

By the properties of the simple $(1,T)$-layout of D every non-leaf node x of T has $q'_x = 0$ and every leaf node x of T has $q'_x \leq 1$. By construction, the edges incident to leaves of T are red. Thus every leaf node $x \in V(T)$ has degree zero in $T[E^b]$. Now λ_q is the maximum, taken over all nodes $x \in V(T)$, of

$$q'_x + \max_{y \in V(T)\,:\,y \leq_\sigma x} \sum_{yz \in E^b(T)\,:\,x \leq_\sigma z} k_{yz} \ . \tag{7.8}$$

Since nodes of T appear in σ according to nondecreasing depth, for each node $x \in V(T)$ at depth i, the summation in (7.8) may be nonzero only for nodes $y \in V(T)$ at depth $i-1$ and i. Since the nodes at depth h and $h+1$ have outdegrees zero in $T[E^b]$, for leaf nodes x, (7.8) is $1+0=1$. Since the nodes at depth less than h have outdegrees q in $T[E^b]$, for non-leaf nodes x, (7.8) is $0 + \max\{q, 0\} = q$. Since $q \geq 1$, by Lemma 7.16, the subdivision D of G has an s-stack q-queue mixed layout. □

Theorems 7.10 and 7.11 with $s = 1$ and $q = 1$ imply:

Theorem 7.12. *Every graph G has a 1-stack 1-queue subdivision with* $\min\{4\lceil\log_2 \mathsf{sn}(G)\rceil, 2 + 4\lceil\log_2 \mathsf{qn}(G)\rceil\}$ *division vertices per edge.* □

Corollary 7.1. *Let $\mathcal{G}$ be a graph family with bounded stack-number and/or bounded queue-number. Then every graph in $\mathcal{G}$ has a 1-stack 1-queue subdivision with a bounded number of division vertices per edge.* □

Since the stack-number of a proper minor-closed graph family is bounded [11], Corollary 7.1 implies that every graph from such a family has a 1-stack 1-queue subdivision with a constant number of division vertices per edge.

7.3.6 Track layouts

First we consider layouts of subdivisions on two tracks.

Theorem 7.13. *For every integer $d \geq 2$, every graph G has a $(d{+}1, 2)$-track subdivision D with $4\lceil \log_d \mathsf{qn}(G)\rceil + 3$ division vertices per edge. That is, D has 2-track thickness $\theta_2(D) \leq d + 1$.*

Proof. By Theorem 7.5, G has a d-queue subdivision D_0 with $2\lceil \log_d \mathsf{qn}(G)\rceil + 1$ division vertices per edge. By Lemma 7.1, $D = D_0'$ has a $(d + 1, 2)$-track layout. □

We now consider 3-track layouts of subdivisions.

Theorem 7.14. *For every integer $d \geq 2$, every graph G has a $(d, 3)$-track subdivision with $1 + 2\lceil \log_d \mathsf{qn}(G)\rceil$ division vertices per edge.*

Proof. Let T_0 be the complete d-ary tree of height $h = \lceil \log_d \mathsf{qn}(G)\rceil$. By Lemma 7.15, G has a subdivision D_0 with $2\lceil \log_d \mathsf{qn}(G)\rceil$ division vertices per edge such that D_0 has a simple $(1, T_0)$-layout in which every non-leaf node $x \in V(T_0)$ has $\deg^+(x) = d$ and $q_x = 0$, and every leaf node $x \in V(T_0)$ has $q_x \leq 1$. By Lemma 7.18, there is a tree T, such that the subdivision $D = D_0'$ obtained by subdividing each intrabag edge of D_0 once has a $(2, T)$-track layout in which every node $x \in V(T)$ has $\sum_{xy\in E(T)} k_{xy} \leq d$ and $\deg^+(x) \leq d$. Consider the (edge-monochromatic) track layout of T produced by Lemma 5.19. By Lemma 7.17 with $p = d$, for some t, D has a (d, t)-track layout with every edge having span one, as illustrated in Figure 7.4 for $d = 2$. By Lemma 5.7(b) with $s = 1$ and $k = d$, D has a $(d, 3)$-track layout □

Finally we consider layouts of subdivisions on four or more tracks, and with no X-crossings.

Theorem 7.15. *For every integer $d \geq 2$, every graph G has a bipartite $(d{+}2)$-track subdivision with at most $8\lceil \log_d \mathsf{qn}(G)\rceil + 1$ division vertices per edge.*

Proof. Let T_0 be the complete d-ary tree of height $h = \lceil \log_d \mathsf{qn}(G)\rceil$. Let T be the subdivision of T_0 obtained as follows. For each node $x \in V(T_0)$ at depth at most $h - 2$, subdivide its rightmost outgoing edge twice, and subdivide the remaining $d - 1$ outgoing edges three times. For each non-leaf node $x \in V(T_0)$ that is incident to a leaf-edge, subdivide its rightmost outgoing edge once, and subdivide the remaining $d - 1$ outgoing edges twice. The resulting tree T has height $h + 3h - 1 = 4\lceil \log_d \mathsf{qn}(G)\rceil - 1$. By Lemma 7.15, G has a subdivision D_0 with at most $8\lceil \log_d \mathsf{qn}(G)\rceil - 2$ division vertices per edge and a simple $(1, T)$-layout, such that every non-leaf node $x \in V(T)$ has $q_x = 0$, and every leaf node $x \in V(T)$ has $q_x \leq 1$. Moreover, every edge of G has an even number of division vertices in D.

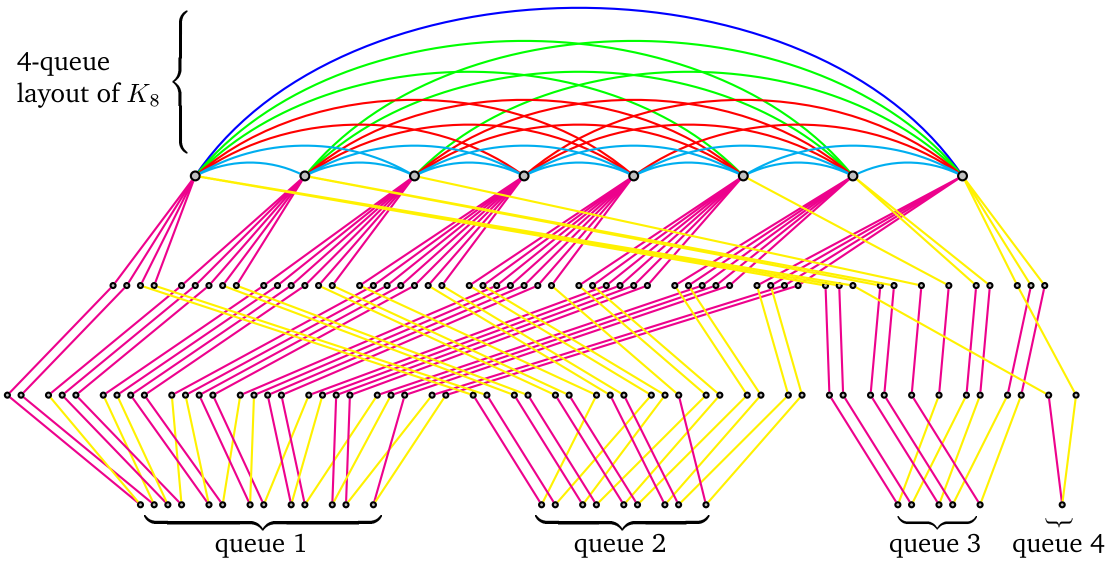


FIGURE 7.4: $(2, 4)$-Track layout of subdivision of K_8 before wrapping.

Let H the graph obtained from T by adding a 4-cycle (x, a_x, b_x, c_x) to each leaf-node $x \in V(T)$, as illustrated in Figure 7.5. Now subdivide every intrabag edge vw of D_0 three times. We obtain a subdivision D of G in which every edge of G has an odd number of division vertices in D. Thus D is bipartite, and has at most $8\lceil \log_d \mathsf{qn}(G) \rceil + 1$ division vertices per edge.

Create a $(1, H)$-layout of D from the simple $(1, T)$-layout of D_0 as follows. For each intrabag edge $vw \in E(D_0)$ mapped to a leaf-node $x \in E(T)$ such that $v <_x w$ in the $(1, T)$-layout, place the division vertex a_{vw} incident to v in the bag H_{a_x}, place the middle division vertex b_{vw} in the bag H_{b_x}, and place the division vertex c_{vw} incident to w in the bag H_{c_x}. Since the intrabag edges mapped to x in the $(1, T)$-layout of D_0 induce a 1-queue layout, we can order the division vertices in H_{a_x}, H_{b_x} and H_{c_x} by the queue order of the edges they subdivide (recall equation (2.1)). As in Lemma 7.3(c), there is no X-crossing in the resulting layout. Thus we have an H-track layout of D.

Now create a track layout of H indexed by

$$\{(i, j) : 0 \leq i \leq 3h, 1 \leq j \leq d\} \cup \{(3h + 1, 1)\} \ .$$

Nodes are ordered in the obvious way so that there are no X-crossings, as illustrated in Figure 7.5.

Firstly, consider a node $x \in V(H)$ that corresponds to a node of T_0 at depth $i \leq h - 2$ in T_0. Recall that the first $d - 1$ outgoing edges of x in T_0 are subdivided three times, and the rightmost outgoing edge in T_0 is subdivided twice. Denote the d outgoing paths at x in H

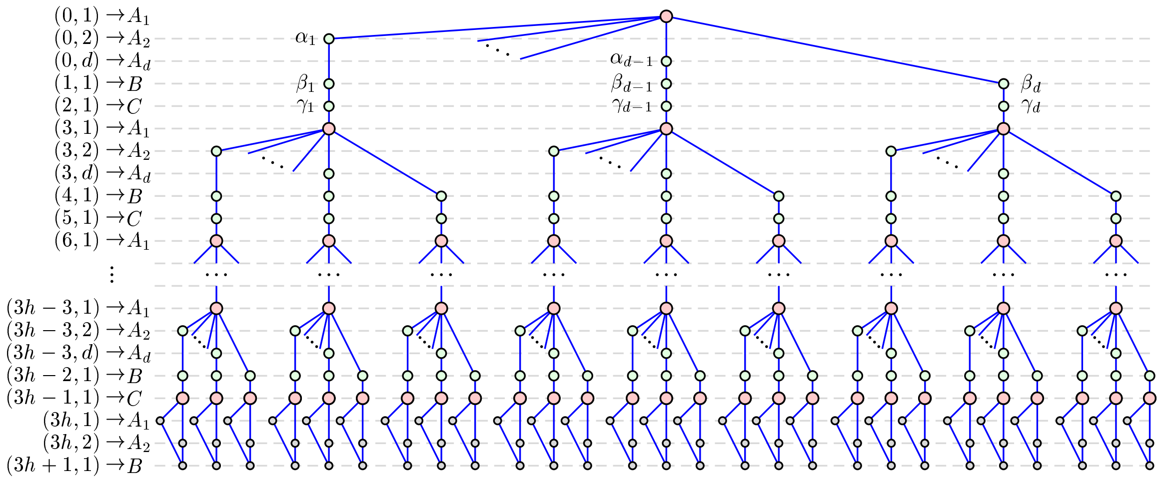


FIGURE 7.5: Track layout of H.

by

$$(x, \alpha_1, \beta_1, \gamma_1), (x, \alpha_2, \beta_2, \gamma_2), \dots, (x, \alpha_{d-1}, \beta_{d-1}, \gamma_{d-1}), (x, \beta_d, \gamma_d) \ .$$

Position x in track $(3i, 1)$. For each $1 \leq j \leq d-1$, position α_j in track $(3i, j+1)$. For each $1 \leq j \leq d$, position β_j in track $(3i+1, 1)$, and position γ_j in track $(3i+2, 1)$.

Now consider a node $x \in V(H)$ that corresponds to a node of T_0 at depth $h-1$ in T_0. Recall that the first $d-1$ outgoing edges of x in T_0 are subdivided twice, and the rightmost outgoing edge in T_0 is subdivided once. Denote the d outgoing paths at x in H by

$$(x, \alpha_1, \beta_1), (x, \alpha_2, \beta_2), \dots, (x, \alpha_{d-1}, \beta_{d-1}), (x, \beta_d) \ .$$

Position x in track $(3h-3, 1)$. Position each node α_j, $1 \leq j \leq d-1$, in track $(3h-3, j+1)$. Position each node β_j, $1 \leq j \leq d$, in track $(3h-2, 1)$.

Finally consider a node $x \in V(H)$ that corresponds to a leaf node of T_0 (at depth h in T_0). Position x in track $(3h-1, 1)$, position a_x in track $(3h, 1)$, position b_x in track $(3h+1, 1)$, and position c_x in track $(3h, 2)$.

Now wrap the track layout of H using Lemma 5.5(b) with $k = 1$. The partial span $s = 1$, so we are wrapping modulo $3 = 2s + 1$. Observe that the track layout of H is indexed by:

$$\begin{aligned} &\big\{(i, j) : i \equiv 0 \pmod 3,\ 0 \leq i \leq 3h,\ 1 \leq j \leq d\big\} \\ \cup\ &\big\{(i, 1) : i \equiv 1 \pmod 3,\ 0 \leq i \leq 3h+1\big\} \\ \cup\ &\big\{(i, 1) : i \equiv 2 \pmod 3\ 0 \leq i \leq 3h\big\} \ . \end{aligned}$$

Thus in Lemma 5.5(b), we have $t'_0 = d$, $t'_1 = 1$, and $t'_2 = 1$. Thus H has a $(d+2)$-track layout. In Figure 7.5 we indicate the new track assignment by $A_1, \dots, A_d, B, C$, where for each $0 \leq i \leq h$, the tracks $(3i, j)$ are mapped A_j, the track $(3i + 1, 1)$ is mapped to B, and the track $(3i + 2, 1)$ is mapped to C. Note that for $i = 3h$ we use the assumption that $d \geq 2$.

It is easily seen that in the $(d + 2)$-track layout of H, every node has at most one neighbour on any other track. Thus replacing each node x in the track layout of H by H_x, we obtain a $(d + 2)$-track layout of D, as in Lemma 7.17. □

Note that the bound on the number of division vertices per edge in Theorem 7.15 can be slightly improved, at the expense of D no longer being bipartite. We will need D to be bipartite in Section 8.3.

The following result proves that in each of Theorems 7.13, 7.14 and 7.15, the bound on the number of division vertices per edge is within a constant factor of optimal for any graph.

Theorem 7.16. *In every (k, t)-track subdivision D of a graph G there is an edge with at least $\frac{1}{2} \log_{2kt} 2\,\mathsf{qn}(G)$ division vertices.*

Proof. Let r be the maximum number of division vertices in an edge of G in the subdivision D. By Lemma 5.14, D has $k(t-1)$-queue layout. By Lemma 7.19, $\mathsf{qn}(G) \leq \frac{1}{2}(2k(t-1)+2)^{2r} - 1 \leq \frac{1}{2}(2kt)^{2r}$. Hence $2\,\mathsf{qn}(G) \leq (2kt)^{2r}$ and $r \geq \frac{1}{2}\log_{2kt} 2\,\mathsf{qn}(G)$. □

7.4 Planar subdivisions

We have seen that every graph has a 3-stack subdivision, a 2-queue subdivision, a 4-track subdivision, and a subdivision with bipartite thickness at most 3. It is interesting to consider which graphs have s-stack subdivisions for each $1 \leq s \leq 2$; which graphs have 1-queue subdivisions; which graphs have t-track subdivisions for $2 \leq t \leq 3$; and which graphs have subdivisions with 2-track thickness at most t for $1 \leq t \leq 2$. In this section we completely answer these questions. As the section title suggests, planar graphs will play a leading role in the characterizations.

7.4.1 Planar stack layouts

Theorem 7.17. *Every graph has a 3-stack subdivision. A graph has a 2-stack subdivision if and only if it is planar. A graph has a 1-stack subdivision if and only if it is outerplanar.*

Proof. By Theorem 7.1 with $d = 2$ every graph has a 3-stack subdivision. The 2-stack graphs are precisely the subgraphs of planar Hamiltonian graphs [7]. Thus a non-planar graph does not have a 2-stack subdivision. Many authors [38, 127, 159] have observed that every planar graph has a subdivision that is a subgraph of a planar Hamiltonian graph (see Lemma 7.20 below), and hence has a 2-stack layout. The 1-stack graphs are precisely the outerplanar graphs [7]. Thus, for any outerplanar graph, the graph itself is a 1-stack subdivision. Conversely, if a subdivision of a graph G is outerplanar then so is G. Thus only the outerplanar graphs have 1-stack subdivisions. □

We now consider how many division vertices per edge are needed in a 2-stack subdivision of any planar graph. Pach and Wenger [159] proved that the subdivision of a planar graph with two division vertices per edge is the subgraph of a Hamiltonian planar graph, and hence has a 2-stack layout. Kaufmann and Wiese [127] and Di Giacomo *et al.* [38] improve this result by showing that the subdivision G' of a planar graph G with one division vertex per edge is the subgraph of a Hamiltonian planar graph, and hence has a 2-stack layout. (Note that Pach and Wenger [159] were more interested in the total number of vertices in the Hamiltonian supergraph rather than the number of division vertices per edge. Di Giacomo *et al.* [38] also prove that the division vertex x of each edge vw is between v and w in the 2-stack layout.) Here we give a new proof of the above result in [38, 127], with the additional property that the Hamiltonian supergraph is bipartite.

Lemma 7.20. *For every planar graph G, the subdivision G' of G with one division vertex per edge is the subgraph of a bipartite Hamiltonian planar graph, and hence has a 2-stack layout.*

Proof. Without loss of generality G is a triangulation. Otherwise we can add edges to G so that every face is a 3-cycle. Let $V = V(G)$. Now subdivide every edge once. Let X be the set of these division vertices. Finally add a single vertex to each face adjacent to the six vertices on that face. Let Y be the set of these vertices. We obtain a planar triangulation H. Observe that $\{V, X, Y\}$ is a vertex 3-colouring of H. Thus every triangle of H contains one vertex from each of V, X and Y. Every such triangle forms a face of H. Therefore every triangle in H is a face, and H has no separating triangles. Since H is a triangulation, by the classical result of Whitney [200], H has a Hamiltonian cycle C.

The subgraph of H induced by $V \cup X$ is G'. Thus H and G' are 2-stack graphs. We now construct a bipartite Hamiltonian planar graph W from H such that G' is a subgraph of W. Consider a face f of G'. Let x be the vertex adjacent to every vertex of f in H. Exactly two edges incident to x are in C. Say $xv, xw \in C$, where $v, w \in f$. Delete all the edges incident to x except xv and xw. Clearly the resulting graph remains Hamiltonian. In the case that the distance from v to w along the boundary of f is odd, subdivide the edge xv. The resulting graph W is clearly Hamiltonian. It is easily verified that each face of W is an even cycle. Thus W is bipartite. □

7.4.2 Planar queue and track layouts

Recall that Open Problem 6.1 asks whether planar graphs have bounded queue-number. We make the following contribution to the study of this problem, which is analogous to

Theorem 7.8 for arbitrary graphs. Recall from Table 6.1 that the best known upper bound on the queue-number of planar graphs is $\mathcal{O}(\sqrt{n})$.

Theorem 7.18. *Let $\mathcal{F}(n)$ be the family of functions $\mathcal{O}(1)$ or $\mathcal{O}(\text{polylog}\, n)$. The following are equivalent:*

(1) *n-vertex planar graphs have queue-number in $\mathcal{F}(n)$,*

(2) *n-vertex bipartite Hamiltonian planar graphs have queue-number in $\mathcal{F}(n)$,*

(3) *n-vertex bipartite Hamiltonian planar graphs have 2-track thickness in $\mathcal{F}(n)$.*

Proof. That (1) implies (2) is immediate. Theorem 7.3 proves that (2) and (3) are equivalent. It remains to prove that (3) implies (1). Suppose that every n-vertex bipartite Hamiltonian planar graph has 2-track thickness at most some function $f(n) \in \mathcal{F}(n)$. Let G be an n-vertex planar graph. By Lemma 7.20, there is a bipartite Hamiltonian planar graph W containing G' as a subgraph. Observe that W has $n + (3n-6) + 2(2n-4) < 8n$ vertices. By assumption, W has 2-track thickness $\theta_2(W) \leq f(8n)$, and since G' is a subgraph of W, we have $\theta_2(G') \leq f(8n)$. By Lemma 7.2, G has queue-number at most $(f(8n))^2 \in \mathcal{F}(n)$. □

We now answer the questions discussed at the start of this section in the case of queue and track layouts.

Lemma 7.21. *Every n-vertex planar graph G has a subdivision D with at most $n-2$ division vertices per edge such that D admits an n-track layout with every edge having span one.*

Proof. By the classical result of Fáry [79] and Wagner [192], G has a straight-line plane drawing. Rotate such a drawing so that every vertex has a unique Y-coordinate. Draw n lines parallel to the X-axis, one through each vertex, and subdivide every edge at the point at which it crosses a line. The subdivision D obtained has at most $n-2$ division vertices per edge. Now consider each line to be a track. Since there are no crossings in the drawing, there are no X-crossings in the track assignment of D. Thus we have an n-track layout of D with every edge having span one. □

Theorem 7.19. *Every graph has a 2-queue subdivision. A graph has a 1-queue subdivision if and only if it is planar.*

Proof. By Theorem 7.5 with $d = 2$ every graph has a 2-queue subdivision. Since 1-queue graphs are planar [114], non-planar graphs do not have 1-queue subdivisions. For any planar graph G, the subdivision D of G from Lemma 7.21 has a 1-queue layout by Lemma 5.14. Note that this conclusion can also be reached by observing that D is arched levelled planar (see [114]). □

Theorem 7.20. *Every graph has a* 4*-track subdivision. A graph has a* 3*-track subdivision if and only if it is planar. A graph has a* 2*-track subdivision if and only if it is a forest of caterpillars.*

Proof. By Theorem 7.15 with $d = 2$ every graph has a 4-track subdivision. By Lemma 8.4, a 3-track graph is planar. Thus non-planar graphs do not have 3-track subdivisions. For any planar graph G, the subdivision of G from Lemma 7.21 can be wrapped into a 3-track layout by Lemma 5.7(b). By Lemma 2.4, a graph has a 2-track layout if and only if it is a forest of caterpillars. If a subdivision of a graph G is a forest of caterpillars then so is G. Thus a graph has a 2-track subdivision if and only if it is a forest of caterpillars. □

We expect that the bound on the number of division vertices per edge in Lemma 7.21 can be improved.

Open Problem 7.2. Is there a function f such that every planar graph G has a subdivision D with $f(\mathsf{qn}(G))$ division vertices per edge, and D has a 1-queue layout and/or a 3-track layout?

Theorem 7.21. *Every graph has a subdivision with* 2*-track thickness at most* 3*. A graph has a subdivision with* 2*-track thickness at most* 2 *if and only if it is planar. A graph has a subdivision with* 2*-track thickness at most* 1 *if and only if it is a forest of caterpillars.*

Proof. The first claim is Theorem 7.13 with $d = 2$. If the 2-track thickness of a graph G is at most 2, then $\mathsf{sn}(G) \leq 2$ by Lemma 5.10(c), and thus G is planar [7]. Thus no non-planar graph has a subdivision with 2-track thickness at most 2. By Lemma 7.21, every planar graph has a subdivision D that admits an (edge-monochromatic) track layout with every edge having span one. By Lemma 5.7(a), such a track layout can be wrapped into a $(2, 2)$-track layout. That is, $\theta_2(D) \leq 2$. This proves the second claim. By Lemma 2.4, a graph has 2-track thickness at most 1 if and only if it is a forest of caterpillars. If a subdivision of G is a forest of caterpillars then so is G. This proves the third claim. □

7.4.3 Planar mixed layouts

Since the stack-number of planar graphs is at most four [207], Theorem 7.12 implies that every planar graph has a 1-stack 1-queue subdivision with eight division vertices per edge. Although asymptoticly much weaker than Theorem 7.10, the following result gives a better bound on the number of division vertices per edge for graphs with small stack-number.

Lemma 7.22. *For every integer* $s \geq 1$*, every graph* G *has a* s*-stack* 1*-queue subdivision with at most* $\lceil \mathsf{sn}(G)/s \rceil$ *division vertices.*

Proof. Let $k = \lceil \mathsf{sn}(G)/s \rceil$. Let $h = \lfloor \frac{k}{2} \rfloor$. Let T be the path on $2h$ edges rooted at the 'middle' vertex r. Thus T has height h. Label each node $x \in V(T)$ by $l(x) = s$. Then

$\sum_x l(x) = (2h+1)s = (2\lfloor \frac{k}{2} \rfloor + 1)s \geq ks = \lceil \mathsf{sn}(G)/s \rceil s \geq \mathsf{sn}(G)$. By Lemma 7.14, G has a subdivision D with at most $2h \leq k$ division vertices per edge, and D has a $(1,T)$-layout such that $s_x \leq s$ for all nodes $x \in V(T)$.

Change the root of T from r to one of the two leaves of T and redirect the edges accordingly. Now every node in T has at most one outgoing edge. Colour all the edges of T black and all the nodes of T red. Since all the edges are black, by Lemma 5.18, T has a topological ordering σ that admits a 1-queue layout of T. Furthermore, since there are no red edges in T,

$$\max_{x \in V(T)} \left\{ s'_x + \sum_{xy \in E^r(T)} k_{xy} + \sum_{yx \in E^r(T)} k_{yx} \right\} \leq s \quad .$$

Since there are no black nodes and since every node has at most one black outgoing edge

$$\max_{x \in V(T)} \left\{ q'_x + \max_{y \in V(T)\,:\, y \leq_\sigma x} \sum_{yz \in E^b(T)\,:\, x \leq_\sigma z} k_{yz} \right\} \leq \max_{x \in V(T)} \sum_{xv \in E^b(T)} k_{xv} \leq 1 \quad .$$

(See Lemma 7.16 to recall the definitions of q'_x and s'_x). Therefore by Lemma 7.16, D has an s-stack 1-queue mixed layout. $\square$

By Lemma 7.22 with $s = 1$ and since planar graphs have 4-stack layouts [207] we have:

Lemma 7.23. *Every planar graph has a* 1*-stack* 1*-queue subdivision with four division vertices per edge.* $\square$

Similar bounds can be be obtained for the number of division vertices per edge in a 1-stack 1-queue subdivision of a graph with small stack-number (see Table 6.1).

Lemma 7.23 provides a partial solution to the following open problem by Heath and Rosenberg [114], who conjectured an affirmative answer.

Open Problem 7.3. [114] Does every planar graph have a 1-stack 1-queue mixed layout?

7.5 Bibliographic notes

The results of this chapter are the subject of [56].

Chapter 8

Three-Dimensional Graph Drawings

In this chapter we prove the following intrinsic relationship between 3D drawings and track layouts.

Theorem 8.1. *Every graph with track-number* $\mathsf{tn}(G) \leq t$ *has a* $2t \times 4t \times 4t\lfloor\frac{n}{t}\rfloor$ *3D drawing with* $\mathcal{O}(t^2 n)$ *volume. Conversely, if a graph* G *has an* $A \times B \times C$ *3D drawing then* G *has track-number* $\mathsf{tn}(G) \leq 2AB$.

This theorem and the bounds on track-number derived in Chapters 6 and 7 imply a number of results on 3D straight-line and polyline drawings. In particular, all graphs of bounded treewidth have $\mathcal{O}(n)$ volume 3D drawings, and all connected graphs have $\mathcal{O}(m \log n)$ volume 3D polyline drawings with $\mathcal{O}(\log n)$ bends per edge.

As Table 8.1 indicates, for bipartite graphs we obtain a better bound on the volume of 3D drawings in terms of track-number. Recently, Dujmović and Wood [58] further generalized this bound to $\mathcal{O}(c^7 tn)$ for c-colourable t-track graphs, resulting in $\mathcal{O}(n^{3/2})$ volume bounds for a variety of graph families, including planar graphs and graphs with bounded degree.

TABLE 8.1: Volume of 3D straight-line drawings and track-number of n-vertex graphs

graph family	volume	reference
t-track	$\mathcal{O}(t^2 n)$	Theorem 8.1
t-track c-colourable	$\mathcal{O}(c^7 tn)$	[58]
bipartite t-track	$2tn$	Lemma 8.3

Table 8.2 summarizes the best known upper bounds on the volume and bends per edge in 3D polyline drawings, including those established in this chapter. In general, there is a trade-off between few bends and small volume in such drawings, which is evident in Table 8.2. Our upper bound of $\mathcal{O}(m \log q)$ is within a factor of $\mathcal{O}(\log q)$ of being optimal for all q-queue graphs, since Bose *et al.* [16] proved that 3D polyline drawings have at least $\frac{1}{8}(n + m)$ volume.

TABLE 8.2: Volume of 3D straight and polyline drawings of graphs with n vertices and $m \geq n$ edges.

graph family	bends per edge	volume	reference
arbitrary	0	$\mathcal{O}(n^3)$	[26]
arbitrary	0	$\mathcal{O}(m^{4/3}n)$	[58]
maximum degree Δ	0	$\mathcal{O}(\Delta mn)$	[58]
maximum degree Δ	0	$\mathcal{O}(\Delta^{15/2}m^{1/2}n)$	[58]
c-colourable	0	$\mathcal{O}(c^2n^2)$	[158]
c-colourable	0	$\mathcal{O}(c^6m^{2/3}n)$	[58]
K_h-minor free	0	$\mathcal{O}(h^{17/2}\log^{7/2} h \cdot n^{3/2})$	[58]
genus γ	0	$\mathcal{O}(\gamma^4 n^{3/2})$	[58]
planar	0	$\mathcal{O}(n^{3/2})$	[58]
outerplanar	0	$\mathcal{O}(n)$	[80]
bounded treewidth	0	$\mathcal{O}(n)$	Corollary 8.1
c-colourable q-queue	1	$\mathcal{O}(cqm)$	Theorem 8.3
arbitrary	1	$\mathcal{O}(nm)$	Theorem 8.4
q-queue	2	$\mathcal{O}(qn)$	Theorem 8.5
q-queue (constant $\epsilon > 0$)	$\mathcal{O}(1)$	$\mathcal{O}(mq^{\epsilon})$	Theorem 8.6
q-queue	$\mathcal{O}(\log q)$	$\mathcal{O}(m \log q)$	Theorem 8.7

The remainder of the chapter is organized as follows. In Section 8.1 we prove Theorem 8.1, which is used in Section 8.2 to derive the above mentioned result on 3D drawings of bounded treewidth graphs. 3D polyline drawings are the focus of Section 8.3. Final remarks are given in Section 8.4.

8.1 Track layouts into 3D drawings

In this section we prove Theorem 8.1, which states that 3D drawings with small volume are closely related to track layouts with few tracks.

Lemma 8.1. *If a graph G has an $A \times B \times C$ 3D drawing, then G has a $2AB$-track layout.*

Proof. Let $V_{x,y}$ be the set of vertices of G with an X-coordinate of x and a Y-coordinate of y, where without loss of generality $1 \leq x \leq A$ and $1 \leq y \leq Y$. With each set $V_{x,y}$ ordered by the Z-coordinates of its elements, $\{V_{x,y} : 1 \leq x \leq A, 1 \leq y \leq Y\}$ is an improper AB-track assignment. There is no X-crossing, as otherwise there would be a crossing in the original drawing, and hence we have an improper AB-track layout. By Observation 2.1, G has a $2AB$-track layout. □

We now prove the converse of Lemma 8.1. The proof is inspired by the generalizations of the moment curve algorithm by Cohen *et al.* [26] and Pach *et al.* [158], described in

Section 1.1.2. Loosely speaking, Cohen *et al.* [26] allow three 'free' dimensions, whereas Pach *et al.* [158] use the assignment of vertices to colour classes to 'fix' one dimension with two dimensions free. We use an assignment of vertices to tracks to fix two dimensions with one dimension free. The style of 3D drawing produced by our algorithm, where tracks are drawn vertically, is illustrated in Figure 8.1.

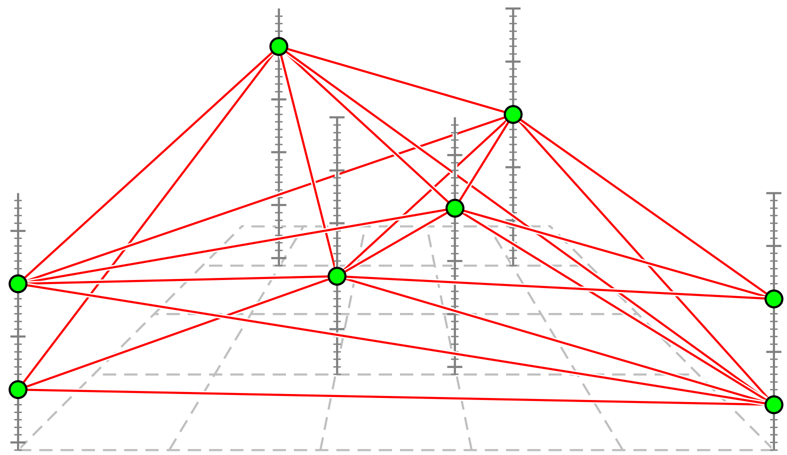


FIGURE 8.1: A 3D drawing produced from a monochromatic track layout.

Lemma 8.2. *If a graph G has a t-track layout, then G has a $t \times 2t \times 2t \cdot n'$ 3D drawing, where n' is the maximum number of vertices in a track.*

Proof. Suppose $\{(V_i, <_i) : 1 \leq i \leq t\}$ is the given t-track layout. Let p be the smallest prime such that $p > t$. Then $p \leq 2t$ by Bertrand's postulate. For each i, $1 \leq i \leq t$, represent the vertices in V_i by the grid-points

$$\{(i, i^2 \bmod p, t) : 1 \leq t \leq p \cdot |V_i|,\, t \equiv i^3 \pmod p\} \ ,$$

such that the Z-coordinates respect the given total order $<_i$. Draw each edge as a line-segment between its endpoints. Suppose two edges e and e' cross such that their endpoints are at distinct points $(i_\alpha, i_\alpha^2 \bmod p, t_\alpha)$, $1 \leq \alpha \leq 4$. Then these points are coplanar, and if M is the matrix

$$M = \begin{pmatrix} 1 & i_1 & i_1^2 \bmod p & t_1 \\ 1 & i_2 & i_2^2 \bmod p & t_2 \\ 1 & i_3 & i_3^2 \bmod p & t_3 \\ 1 & i_4 & i_4^2 \bmod p & t_4 \end{pmatrix}$$

then the determinant $\det(M) = 0$. We proceed by considering the number of distinct tracks $N = |\{i_1, i_2, i_3, i_4\}|$. By the definition of a track layout, $N \geq 2$.

- $N = 2$: Since there is no X-crossing, e and e' do not cross.

• $N = 3$: Without loss of generality $i_1 = i_2$. It follows that $\det(M) = (t_2 - t_1) \cdot \det(M')$, where

$$M' = \begin{pmatrix} 1 & i_2 & i_2^2 \bmod p \\ 1 & i_3 & i_3^2 \bmod p \\ 1 & i_4 & i_4^2 \bmod p \end{pmatrix} .$$

Since $t_1 \neq t_2$, $\det(M') = 0$. However, M' is a Vandermonde matrix modulo p, and thus

$$\det(M') \equiv (i_2 - i_3)(i_2 - i_4)(i_3 - i_4) \pmod{p},$$

which is non-zero since i_2, i_3 and i_4 are distinct integers smaller than the prime p, a contradiction.

• $N = 4$: Let M' be the matrix obtained from M by taking each entry modulo p. Then $\det(M') = 0$. Since $t_\alpha \equiv i_\alpha^3 \pmod{p}$, $1 \leq \alpha \leq 4$,

$$M' \equiv \begin{pmatrix} 1 & i_1 & i_1^2 & i_1^3 \\ 1 & i_2 & i_2^2 & i_2^3 \\ 1 & i_3 & i_3^2 & i_3^3 \\ 1 & i_4 & i_4^2 & i_4^3 \end{pmatrix} \pmod{p} .$$

Since each $i_\alpha < p$, M' is a Vandermonde matrix modulo p, and thus

$$\det(M') \equiv (i_1 - i_2)(i_1 - i_3)(i_1 - i_4)(i_2 - i_3)(i_2 - i_4)(i_3 - i_4) \pmod{p},$$

which is non-zero since $i_\alpha \neq i_\beta$ and p is a prime. This contradiction proves there are no edge crossings. The produced drawing is at most $t \times 2t \times 2t \cdot n'$. □

Proof of Theorem 8.1. Suppose G is a graph with a t-track layout. By Lemma 5.4 with $t' = t$, G has a $2t$-track layout with at most $\lceil \frac{n}{t} \rceil$ vertices in each track. By Lemma 8.2, G has a $2t \times 4t \times 4t \cdot \lceil \frac{n}{t} \rceil$ drawing. Conversely, if a graph G has a $A \times B \times C$ 3D drawing, then by Lemma 8.1, G has track-number $\mathsf{tn}(G) \leq 2AB$. □

For bipartite graphs the bound in Lemma 8.2 can be improved as follows.

Lemma 8.3. *Every t-track bipartite graph G with bipartition $\{A, B\}$ has a $2 \times t \times \max\{|A|, |B|\}$ 3D drawing.*

Proof. Let $\{T_i : 1 \leq i \leq t\}$ be a t-track layout of G. For each $1 \leq i \leq t$, let $A_i = T_i \cap A$ and $B_i = T_i \cap B$. Order each A_i and B_i as in T_i. Place the j^{th} vertex in A_i at $(0, t - i + 1, j + \sum_{k=1}^{i-1} |A_k|)$. Place the j^{th} vertex in B_i at $(1, i, j + \sum_{k=1}^{i-1} |B_k|)$. The drawing is thus $2 \times t \times \max\{|A|, |B|\}$. There is no crossing between edges in $G[A_i, B_j]$ as otherwise there would be an X-crossing in the track layout. Clearly there is no crossing between edges in

$G[A_i, B_j]$ and $G[A_i, B_k]$ for $j \neq k$. Suppose there is a crossing between edges in $G[A_i, B_j]$ and $G[A_k, B_\ell]$ with $i \neq k$ and $j \neq \ell$. Without loss of generality $i < k$. Then the projections of the edges in the XY-plane also cross, and thus $\ell < j$. This implies that the projections of the edges in the XZ-plane do not cross, and thus the edges do not cross. Figure 8.2 illustrates the construction used in this proof. □

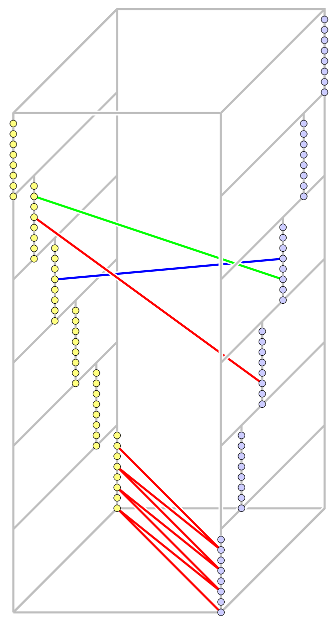


FIGURE 8.2: 3D drawing of a 6-track bipartite graph.

This improvement can be generalized to c-colourable graphs by the number theoretic analysis similar to, but much more involved than, the one used in the proof of Lemma 8.2. The constants in Lemma 8.2 can be dramatically improved in the case of a 3-track or 4-track layout. Here the vertices are positioned on the edges of a triangular or rectangular prism. These models of graph drawing were introduced by Felsner *et al.* [80].

Lemma 8.4. *Let $\{V_1, V_2, V_3\}$ be a 3-track layout of a graph G. Let $n' = \max\{|V_1|, |V_2|, |V_3|\}$. Then G has a $2 \times 2 \times n'$ straight-line drawing with the vertices on a triangular prism. In this case, G is necessarily planar.*

Proof. Position the i^{th} vertex in V_1 at $(0, 0, i)$. Position the i^{th} vertex in V_2 at $(1, 0, i)$. Position the i^{th} vertex in V_3 at $(0, 1, i)$. Since there is no X-crossing in the track layout, no two edges cross. Since G is embedded in a surface homeomorphic to the sphere, G is planar. □

Lemma 8.5. *Let $\{V_1, V_2, V_3, V_4\}$ be a 4-track layout of a graph G. Let $n' = \max\{|V_1|, |V_2|, |V_3|, |V_4|\}$. Then G has a $2 \times 2 \times 2n'$ straight-line drawing with the vertices on a rectangular prism.*

Proof. Position the i^{th} vertex in V_1 at $(0, 0, 2i)$. Position the i^{th} vertex in V_2 at $(1, 0, 2i)$. Position the i^{th} vertex in V_3 at $(0, 1, 2i)$. Position the i^{th} vertex in V_4 at $(1, 1, 2i + 1)$. Clearly the only possible crossing is between edges vw and xy with $v \in V_1$, $w \in V_4$, $x \in V_2$, and $y \in V_3$. Such a crossing point is on the line $L = \{(\frac{1}{2}, \frac{1}{2}, z) : z \in \mathbb{R}\}$. However, vw intersects L at $(\frac{1}{2}, \frac{1}{2}, \alpha + \frac{1}{2})$ for some integer α, and xy intersects L at $(\frac{1}{2}, \frac{1}{2}, \beta)$ for some integer β. Thus vw and xy do not intersect. □

Di Giacomo and Meijer [40] proved that a 4-track graph with n vertices has a $2 \times 2 \times n$ drawing. When $n' < \frac{n}{2}$ the above construction has less volume. For 4-track layouts we also have the following 'non-prism' constructions.

Lemma 8.6. *Let $\{V_1, V_2, V_3, V_4\}$ be a 4-track layout of a graph G. Let $n' = \max\{|V_1|, |V_2|, |V_3|, |V_4|\}$. Then G has a $3 \times 3 \times n'$ straight-line drawing.*

Proof. Position the i^{th} vertex in V_1 at $(0, 0, i)$. Position the i^{th} vertex in V_2 at $(1, 1, i)$. Position the i^{th} vertex in V_3 at $(2, 1, i)$. Position the i^{th} vertex in V_4 at $(0, 2, i)$. Since the 'footprint' of the drawing in the XY-plane is a plane K_4 there is no crossings in the 3D construction. □

8.2 Straight-line drawings

Theorems 6.3 and 8.1 imply the following theorem.

Theorem 8.2. *Let G be a graph with maximum degree $\Delta(G)$, pathwidth $\mathsf{pw}(G)$, tree-partition-width $\mathsf{tpw}(G)$, and treewidth $\mathsf{tw}(G)$. Then G has a 3D drawing with the following dimensions:*

(a) $\mathcal{O}(\mathsf{pw}(G)) \times \mathcal{O}(\mathsf{pw}(G)) \times \mathcal{O}(n)$, *which is* $\mathcal{O}(\mathsf{tw}(G)\ \log n) \times \mathcal{O}(\mathsf{tw}(G)\ \log n) \times \mathcal{O}(n)$,

(b) $\mathcal{O}(\mathsf{tpw}(G)) \times \mathcal{O}(\mathsf{tpw}(G)) \times \mathcal{O}(n)$, *which is* $\mathcal{O}(\Delta(G)\,\mathsf{tw}(G)) \times \mathcal{O}(\Delta(G)\,\mathsf{tw}(G)) \times \mathcal{O}(n)$,

(c) $\mathcal{O}(6^{4^{\mathsf{tw}(G)}}) \times \mathcal{O}(6^{4^{\mathsf{tw}(G)}}) \times \mathcal{O}(n)$. □

Most importantly, we have the following corollary of Theorem 8.2(c).

Corollary 8.1. *Every graph with bounded treewidth has a 3D drawing with $\mathcal{O}(n)$ volume*[1]. □

Note that bounded treewidth is not necessary for a graph to have a 3D drawing with $\mathcal{O}(n)$ volume. The $\sqrt{n} \times \sqrt{n}$ plane grid graph has $\Theta(\sqrt{n})$ treewidth, but is easily seen to have an improper 3-track layout, and thus, by Lemma 8.2, has a 3D drawing with $\mathcal{O}(n)$ volume.

[1]The large constant in the $\mathcal{O}(n)$ volume bound of Corollary 8.1 can be improved for graphs with small treewidth. In [59] we proved that every series-parallel graph has an 18-track layout, and thus has a $36 \times 37 \times 37\lceil \frac{n}{18} \rceil$ drawing.

8.3 Polyline drawings

We first prove results for 3D 1-bend drawings.

Theorem 8.3. *Every c-colourable q-queue graph G with n vertices and m edges has a $2 \times c(q+1) \times (n+m)$ polyline drawing with one bend per edge. The volume is $2c(q+1)(n+m)$.*

Proof. The subdivision G' of G with one division vertex per edge is bipartite and has $n+m$ vertices. By Lemma 7.3(b), $\mathsf{tn}(G') \leq c(q+1)$. Thus by Lemma 8.3, G' has a $2 \times c(q+1) \times (n+m)$ straight-line drawing, which is the desired 3D polyline drawing of G. □

The next result applies a construction of Calamoneri and Sterbini [19].

Theorem 8.4. *Every n-vertex m-edge graph G has an $n \times m \times 2$ polyline drawing with one bend per edge.*

Proof. Let $(v_1, v_2, \dots, v_n)$ be an arbitrary vertex ordering of G. Let $(x_1, x_2, \dots, x_m)$ be an arbitrary ordering of the division vertices of G'. Place each v_i at $(i, 0, 0)$ and each x_j at $(0, j, 1)$. Clearly the endpoints of any two disjoint edges of G' are not coplanar (see [19]). Thus no two edges cross, and we have an $n \times m \times 2$ straight-line drawing of G', which is a 3D 1-bend drawing of G. □

Now consider 3D 2-bend drawings.

Theorem 8.5. *Every n-vertex m-edge q-queue graph G has a $2 \times 2q \times (2n-3)$ polyline drawing with two bends per edge. The volume is at most $8qn \in \mathcal{O}(n\sqrt{m})$.*

Proof. Let $\sigma = (v_1, v_2, \dots, v_n)$ be the vertex ordering in a q-queue layout of G. Let $\{E_\ell : 1 \leq \ell \leq q\}$ be the queues. Order the edges in each queue E_ℓ according to the queue order (see Eq. (2.1)). Denote by $(L(e), X(e), Y(e), R(e))$ the path replacing e in G'', where $L(e) <_\sigma R(e)$. Put each vertex v_i at $(0, 0, i)$. If e is the j^{th} edge in the ordering of E_ℓ, put the division vertices $X(e)$ at $(1, 2\ell, j)$ and $Y(e)$ at $(1, 2\ell+1, j)$. Observe that the projection of the drawing onto the XY-plane is planar. Thus the only possible crossings occur between edges contained in a plane parallel with the Z-axis. Thus an X-crossing could only occur between pairs of edges $\{L(e)X(e), L(f)X(f)\}$, $\{X(e)Y(e), X(f)Y(f)\}$, or $\{Y(e)R(e), Y(f)R(f)\}$, where e and f are in a single queue E_ℓ. Suppose $e <_\ell f$. Then the Z-coordinates satisfy: $Z(L(e)) \leq Z(L(f))$, $Z(R(e)) \leq Z(R(f))$, $Z(X(e)) < Z(X(f))$, and $Z(Y(e)) < Z(Y(f))$. Thus there is no crossing. The drawing is at most $2 \times 2q \times (2n-3)$ since each queue has at most $2n-3$ edges [55, 114, 160]. The volume is at most $8qn$, which is $\mathcal{O}(n\sqrt{m})$ [55, 114, 178]. □

Theorem 8.6. *Let G be a q-queue graph with n vertices and m edges. For every $\epsilon > 0$, G has a*

$$2 \times \left(\lceil q^\epsilon \rceil + 2 \right) \times \left(n + (8 \left\lceil \tfrac{1}{\epsilon} \right\rceil + 1)m \right)$$

polyline drawing with at most $8\lceil\frac{1}{\epsilon}\rceil + 1$ bends per edge. The volume is $\mathcal{O}(q^\epsilon(n + \frac{m}{\epsilon}))$. For constant ϵ there are $\mathcal{O}(1)$ bends per edge and the volume is $\mathcal{O}(q^\epsilon(n+m))$, which is in $\mathcal{O}(n^\epsilon(n+m))$.

Proof. Let $d = \lceil q^\epsilon \rceil$. By Theorem 7.15, G has a bipartite subdivision D with at most $8\lceil \log_d q\rceil + 1$ division vertices per edge such that the track-number $\mathsf{tn}(D) \leq d + 2$. Now $\log_d q \leq \frac{1}{\epsilon}$. Thus D has at most $8\lceil\frac{1}{\epsilon}\rceil + 1$ division vertices per edge, and $\mathsf{tn}(D) \leq \lceil q^\epsilon \rceil + 2$. The number of vertices of D is at most $n + (8\lceil\frac{1}{\epsilon}\rceil + 1)m$. By Lemma 8.3, D has a $2 \times (\lceil q^\epsilon\rceil + 2) \times (n + (8\lceil\frac{1}{\epsilon}\rceil + 1)m)$ straight-line drawing, which is the desired 3D polyline drawing of G. The other claims immediately follow since $q \leq n$. ☐

Theorem 8.7. *Every q-queue graph G with n vertices and m edges has a $2 \times 2 \times \big(n + m(8\lceil \log_2 q\rceil + 1)\big)$ polyline drawing on a rectangular prism. There are $\mathcal{O}(\log q)$ bends per edge, and the volume is $\mathcal{O}(n + m\log q)$ volume, which is in $\mathcal{O}(n + m\log n)$.*

Proof. By Theorem 7.15, G has a 4-track subdivision D with at most $8\lceil\log_2 q\rceil + 1$ division vertices per edge. The number of vertices of D is at most $n+m(8\lceil\log_2 q\rceil+1)$. By Lemma 8.5, D has a $2 \times 2 \times \big(n + m(8\lceil\log_2 q\rceil + 1)\big)$ straight-line drawing, which is the desired polyline drawing of G. The volume is $\mathcal{O}(n + m\log n)$ since $q \leq n$. ☐

Since the queue-number of an n-vertex graph is at most n we have:

Corollary 8.2. *Every graph with n vertices and m edges has a polyline drawing with $\mathcal{O}(n + m\log n)$ volume and $\mathcal{O}(\log n)$ bends per edge.* ☐

8.4 Conclusion and bibliographic notes

Felsner *et al.* [80] asked the following question.

Open Problem 8.1. [80] Does every n-vertex planar graph have a 3D drawing with $\mathcal{O}(n)$ volume?

By Theorem 8.1, this question has an affirmative answer if every planar graph has $\mathcal{O}(1)$ track-number. Whether every planar graph has $\mathcal{O}(1)$ track-number is an open problem due to H. de Fraysseix [private communication, 2000].

Open Problem 8.2. [H. de Fraysseix (2000)] Do planar graphs have bounded track-number?

By Theorem 5.2, this question is equivalent to the Open Problem 6.1.

Theorem 8.1 (that is, Lemmas 8.1 and 8.2) have appeared in [53]. Lemma 8.3 has appeared in [58]. The results of Section 8.2 have appeared in [59]. Lemmas 8.4 and 8.5 and the results of Section 8.3 are a part of [56].

Chapter 9

Conclusion and Open Problems

In this thesis we investigated graph drawing problems from two complementary directions. In one, we studied parameterized analogues of well-known hard algorithmic problems. We applied two FPT techniques for their solution. In the other direction, we studied structural properties of graphs through their track layouts. The results of this study were applied to several well-known graph layout models.

The study of parameterized complexity of hard graph drawing problems has only just begun. A number of techniques useful for deriving FPT algorithms have not been considered in this thesis. One example is the popular use of dynamic programming on tree decompositions. Furthermore, the problems we studied were parameterized by some measure restricting the desired output. We may instead parameterize a problem by some measure restricting input graphs. For example, is the problem of testing *upward planarity* of planar directed graphs with treewidth k in $\mathcal{FPT}$? Here the treewidth of an input graph is the parameter of the problem. For input graphs excluding K_h minor, h may be an appropriate parameter for some problems. Here the techniques developed for the proof of the graph minors theorem may prove useful again. A general direction for future work would be to explore these avenues — they have the potential to provide new insights into a variety of problems arising in graph drawing.

We conclude this thesis by summarizing and relating the open problems considered or raised in this thesis.

Open problems

- Open Problem 1.1 [110]: Is stack-number bounded by queue-number?

Blankenship and Oporowski [10] conjectured that the next question has an affirmative answer (Conjecture 7.1). By Theorem 7.2 the truth of that conjecture would imply an affirmative solution to Open Problem 1.1.

- [10] Does there exist a function f such that for every subdivision D of a graph G with at most one division vertex per edge, $\mathsf{sn}(G) \leq f(\mathsf{sn}(D))$.

By Theorem 7.8 the following open problems are equivalent.

- Open Problem 1.2 [110]: Is queue-number bounded by stack-number?
- Do bipartite 3-stack graphs have bounded queue-number?
- Do bipartite 3-stack graphs have bounded 2-track thickness?

By Theorems 5.2 and 7.18 the following open problems are equivalent.

- Open Problem 6.1 [110]: Do planar graphs have bounded queue-number?
- Open Problem 8.2 [H. de Fraysseix (2000)]: Do planar graphs have bounded track-number?
- Do bipartite Hamiltonian planar graphs have bounded queue-number?
- Do bipartite Hamiltonian planar graphs have bounded 2-track thickness?

By Theorem 8.1 an affirmative answer to any of the last seven questions implies an affirmative answer to the following question.

- Open Problem 8.1 [80]: Does every n-vertex planar graph have a 3D drawing with $\mathcal{O}(n)$ volume?

By Theorem 5.2, queue-number and track-number are tied for any proper minor closed family of graphs. An affirmative solution to the following open problem would imply that queue-number and track-number are tied (for all graphs).

- Open Problem 5.2: Is star chromatic number bounded by queue-number?

The following are the remaining open problems raised in this thesis.

- Open Problem 3.1 [H. Fernau (2003)]: Is the 1-SIDED CROSSING MINIMIZATION problem in the class $\mathcal{FPT}$ when parameterized by the number of crossings by which a 2-layer drawing is allowed to exceed the lower bound?
- Open Problem 4.1: Is there a c-approximation algorithm for the optimization version of the 2-LAYER PLANARIZATION problem with $c < 2$? Is there an FPT algorithm for the 2-LAYER PLANARIZATION problem parameterized by the number of edge deletions k, with the exponential part of the running time better than 6^k?

- Open Problem 4.2: Is there a c-approximation algorithm for the optimization version of the 1-LAYER PLANARIZATION problem with $c < 3$? Is there an FPT algorithm for the 1-LAYER PLANARIZATION problem parameterized by the number of edge deletions k, with the exponential part of the running time better than 3^k? Is there a problem kernel of size $f(k)$ for the problem?

- Open Problem 5.1: What is the computational complexity of recognizing track graphs? Is it $\mathcal{NP}$-complete to recognize $(2,2)$-track graphs? Is it $\mathcal{NP}$-complete to recognize 3-track graphs?

- Open Problem 6.2: Is the queue-number of a graph bounded by a polynomial function of its treewidth?

- Open Problem 7.1: Is 2-track sub-thickness $\mathsf{sub}\text{-}\theta_2(G) \in o(\mathsf{qn}(G))$?

- Open Problem 7.2: Is there a function f such that every planar graph G has a subdivision D with $f(\mathsf{qn}(G))$ division vertices per edge, and D has a 1-queue layout and/or a 3-track layout?

- Open Problem 7.3: [114] Does every planar graph have a 1-stack 1-queue mixed layout?

Bibliography

[1] A. A. Ageev, A triangle-free circle graph with chromatic number 5. *Discrete Math.*, **152(1-3)**:295–298, 1996.

[2] M. Alzohairi and I. Rival, Series-parallel planar ordered sets have pagenumber two. In S. North, ed., *Proc. 4th International Symp. on Graph Drawing (GD '96)*, vol. 1190 of *Lecture Notes in Comput. Sci.*, pp. 11–24, Springer, 1997.

[3] M. Alzohairi, I. Rival, and A. Kostochka, The pagenumber of spherical lattices is unbounded. *Arab J. Math. Sci.*, **7(1)**:79–82, 2001.

[4] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Appl. Math.*, **23(1)**:11–24, 1989.

[5] G. Atneosen, *On the embeddability of compacta in n-books: intrinsic and extrinsic properties*. Ph.D. thesis, Michigan State University, 1968.

[6] L. W. Beineke, Biplanar graphs: a survey. *Comput. Math. Appl.*, **34(11)**:1–8, 1997.

[7] F. Bernhart and P. C. Kainen, The book thickness of a graph. *J. Combin. Theory Ser. B*, **27(3)**:320–331, 1979.

[8] S. N. Bhatt, F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg, Scheduling tree-dags using FIFO queues: A control-memory trade-off. *J. Parallel Distrib. Comput.*, **33**:55–68, 1996.

[9] T. C. Biedl, T. Shermer, S. Whitesides, and S. Wismath, Bounds for orthogonal 3-D graph drawing. *J. Graph Algorithms Appl.*, **3(4)**:63–79, 1999.

[10] R. Blankenship and B. Oporowski, Drawing subdivisions of complete and complete bipartite graphs on books. Tech. Rep. 1999-4, Department of Mathematics, Louisiana State University, 1999.

[11] R. Blankenship and B. Oporowski, Book embeddings of graphs and minor-closed classes. In *Proc. 32nd Southeastern International Conf. on Combinatorics, Graph Theory and Computing*, Department of Mathematics, Louisiana State University, 2001.

[12] H. Bodlaender, ed., *Proc. 29th Workshop on Graph Theoretic Concepts in Computer Science (WG'03)*, Lecture Notes in Comput. Sci., Springer, to appear.

[13] H. L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, **25(6)**:1305–1317, 1996.

[14] H. L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci.*, **209(1-2)**:1–45, 1998.

[15] H. L. Bodlaender and J. Engelfriet, Domino treewidth. *J. Algorithms*, **24(1)**:94–123, 1997.

[16] P. Bose, J. Czyzowicz, P. Morin, and D. R. Wood, The maximum number of edges in a three-dimensional grid-drawing. In *Proc. 19th European Workshop on Computational Geometry*, pp. 101–103, University of Bonn, Germany, 2003.

[17] F. J. Brandenburg, ed., *Proc. International Symp. on Graph Drawing (GD '95)*, vol. 1027 of *Lecture Notes in Comput. Sci.*, Springer, 1996.

[18] I. Bruß and A. Frick, Fast interactive 3-D graph visualization. In [17], pp. 99–110.

[19] T. Calamoneri and A. Sterbini, 3D straight-line grid drawing of 4-colorable graphs. *Inform. Process. Lett.*, **63(2)**:97–102, 1997.

[20] M. J. Carpano, Automatic display of hierarchized graphs for computer aided decision analysis. *IEEE Trans. Syst. Man Cybern.*, **SMC-10(11)**:705– 715, 1980.

[21] C. Catarci, The assignment heuristic for crossing reduction. *IEEE Trans. on Systems, Man, and Cybernetics*, **25(3)**, 1995.

[22] J. Chen, I. Kanj, and W. Jia, Vertex cover: Further observations and further improvements. In P. Widmayer, G. Neyer, and S. Eidenbenz, eds., *Proc. 25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '99)*, vol. 1665 of *Lecture Notes in Comput. Sci.*, pp. 313–324, Springer, 1999.

[23] K. Chilakamarri, N. Dean, and M. Littman, Three-dimensional Tutte embedding. In *Proc. 26th Southeastern International Conf. on Combinatorics, Graph Theory and Computing*, vol. 107 of *Cong. Numer.*, pp. 129–140, 1995.

[24] M. Chrobak, M. Goodrich, and R. Tamassia, Convex drawings of graphs in two and three dimensions. In *Proc. 12th Annual ACM Symp. on Comput. Geom.*, pp. 319–328, 1996.

[25] F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg, Embedding graphs in books: a layout problem with applications to VLSI design. *SIAM J. Algebraic Discrete Methods*, **8(1)**:33–58, 1987.

[26] R. F. Cohen, P. Eades, T. Lin, and F. Ruskey, Three-dimensional graph drawing. *Algorithmica*, **17(2)**:199–208, 1996.

[27] T. Corman, R. R. C. Leiserson, and C. Stein, *Introduction to Algorithms - Second Edition*. MIT Press, 2nd edn., 2001.

[28] G. A. Cottafava and O. D'Antona, The fixed outerthickness of graphs. In H. W. Hale and A. N. Michel, eds., *Proc. 21st Midwest Symp. on Circuits and Systems*, pp. 32–35, Western Periodicals, California, 1978.

[29] G. A. COTTAFAVA AND O. D'ANTONA, Book-thickness and book-coarseness of graphs. In *Proc. 5th International Symp. on Network Theory (Sarajevo)*, pp. 337–340, 1984.

[30] B. COURCELLE, Graph rewriting: An algebraic and logic approach. *Handbook of Theoretical Computer Science*, **B**:193–242, 1990.

[31] P. R. CROMWELL AND I. J. NUTT, Embedding knots and links in an open book. II. Bounds on arc index. *Math. Proc. Cambridge Philos. Soc.*, **119(2)**:309–319, 1996.

[32] I. F. CRUZ AND J. P. TWAROG, 3D graph drawing with simulated annealing. In [17], pp. 162–165.

[33] H. DE FRAYSSEIX, P. OSSONA DE MENDEZ, AND J. PACH, A left-first search algorithm for planar graphs. *Discrete Comput. Geom.*, **13(3-4)**:459–468, 1995.

[34] H. DE FRAYSSEIX, J. PACH, AND R. POLLACK, How to draw a planar graph on a grid. *Combinatorica*, **10(1)**:41–51, 1990.

[35] G. DI BATTISTA, P. EADES, R. TAMASSIA, AND I. G. TOLLIS, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

[36] E. DI GIACOMO, Drawing series-parallel graphs on restricted integer 3D grids. In [139].

[37] E. DI GIACOMO, W. DIDIMO, G. LIOTTA, AND S. WISMATH, Book embeddings and point-set embeddings of series-parallel digraphs. In [93], pp. 162–173.

[38] E. DI GIACOMO, W. DIDIMO, G. LIOTTA, AND S. K. WISMATH, Drawing planar graphs on a curve. In [12].

[39] E. DI GIACOMO, G. LIOTTA, AND S. WISMATH, Drawing series-parallel graphs on a box. In *Proc. 14th Canadian Conf. on Computational Geometry (CCCG '02)*, pp. 149–153, The University of Lethbridge, Canada, 2002.

[40] E. DI GIACOMO AND H. MEIJER, Track drawings of graphs with constant queue number. In [139].

[41] R. DIESTEL, *Graph theory*, vol. 173 of *Graduate Texts in Mathematics*. Springer, 2nd edn., 2000.

[42] M. B. DILLENCOURT, D. EPPSTEIN, AND D. S. HIRSCHBERG, Geometric thickness of complete graphs. *J. Graph Algorithms Appl.*, **4(3)**:5–17, 2000.

[43] R. P. DILWORTH, A decomposition theorem for partially ordered sets. *Ann. of Math. (2)*, **51**:161–166, 1950.

[44] G. DING AND B. OPOROWSKI, Some results on tree decomposition of graphs. *J. Graph Theory*, **20(4)**:481–499, 1995.

[45] G. DING AND B. OPOROWSKI, On tree-partitions of graphs. *Discrete Math.*, **149(1-3)**:45–58, 1996.

[46] R. DOWNEY, M. FELLOWS, AND U. STEGE, Parameterized complexity: A framework for systematically confronting computational intractability. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 49, pp. 49–99, 1999.

[47] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized complexity*. Springer, 1999.

[48] S. DRESBACH, A new heuristic layout algorithm for directed acyclic graphs. In U. DERIGS, A. BACHEM, AND A. B. A. DREXL, eds., *Operations Research Proceedings (1994)*, pp. 121–126, Springer, 1995.

[49] V. DUJMOVIĆ, M. FELLOWS, M. HALLETT, M. KITCHING, G. LIOTTA, C. MCCARTIN, N. NISHIMURA, P. RAGDE, F. ROSEMAND, M. SUDERMAN, S. WHITESIDES, AND D. R. WOOD, On the parameterized complexity of layered graph drawing. In F. MEYER AUF DER HEIDE, ed., *Proc. 5th Annual European Symp. on Algorithms (ESA '01)*, vol. 2161 of *Lecture Notes in Comput. Sci.*, pp. 488–499, Springer, 2001.

[50] V. DUJMOVIĆ, M. FELLOWS, M. HALLETT, M. KITCHING, G. LIOTTA, C. MCCARTIN, N. NISHIMURA, P. RAGDE, F. ROSEMAND, M. SUDERMAN, S. WHITESIDES, AND D. R. WOOD, A fixed-parameter approach to two-layer planarization. In [150], pp. 1–15.

[51] V. DUJMOVIĆ, H. FERNAU, AND M. KAUFMANN, Fixed parameter algorithms for 1-sided crossing minimization revisited. In [139].

[52] V. DUJMOVIĆ, P. MORIN, AND D. R. WOOD, Layout of graphs with bounded tree-width, 2002, submitted.

[53] V. DUJMOVIĆ, P. MORIN, AND D. R. WOOD, Path-width and three-dimensional straight-line grid drawings of graphs. In [93], pp. 42–53.

[54] V. DUJMOVIĆ AND S. WHITESIDES, An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. In [93], pp. 118–130.

[55] V. DUJMOVIĆ AND D. R. WOOD, On linear layouts of graphs. Tech. Rep. TR-2003-05, School of Computer Science, Carleton University, Ottawa, Canada, 2003.

[56] V. DUJMOVIĆ AND D. R. WOOD, Stacks, queue and tracks: Layouts of graph subdivisions. Tech. Rep. TR-2003-07, School of Computer Science, Carleton University, Ottawa, Canada, 2003.

[57] V. DUJMOVIĆ AND D. R. WOOD, Track layouts of graphs. Tech. Rep. TR-2003-06, School of Computer Science, Carleton University, Ottawa, Canada, 2003.

[58] V. DUJMOVIĆ AND D. R. WOOD, Three-dimensional grid drawings with sub-quadratic volume. In [139], Also in J. PACH, ed., Towards a Theory of Geometric Graphs, *Contemporary Mathematics*, Amer. Math. Soc., to appear.

[59] V. DUJMOVIĆ AND D. R. WOOD, Tree-partitions of k-trees with applications in graph layout. In [12], Also in Tech. Rep. TR-2002-03, School of Computer Science, Carleton University, 2002.

[60] I. A. Dynnikov, Three-page representation of links. *Uspekhi Mat. Nauk*, **53(5(323))**:237–238, 1998.

[61] I. A. Dynnikov, Three-page approach to knot theory. Coding and local motions. *Funktsional. Anal. i Prilozhen.*, **33(4)**:25–37, 96, 1999.

[62] I. A. Dynnikov, A three-page approach to knot theory. The universal semigroup. *Funktsional. Anal. i Prilozhen.*, **34(1)**:29–40, 96, 2000.

[63] I. A. Dynnikov, A new way to represent links, one-dimensional formalism and untangling technology. *Acta Appl. Math.*, **69(3)**:243–283, 2001.

[64] P. Eades and P. Garvan, Drawing stressed planar graphs in three dimensions. In [17], pp. 212–223.

[65] P. Eades and D. Kelly, Heuristics for drawing 2-layered networks. *Ars Combin.*, **21(A)**:89–98, 1986.

[66] P. Eades, B. D. McKay, and N. C. Wormald, On an edge crossing problem. In *Proc. 9th Australian Computer Science Conference*, pp. 327–334, Australian National University, 1986.

[67] P. Eades, C. Stirk, and S. Whitesides, The techniques of Kolmogorov and Barzdin for three dimensional orthogonal graph drawings. *Inform. Proc. Lett.*, **60(2)**:97–103, 1996.

[68] P. Eades, A. Symvonis, and S. Whitesides, Three dimensional orthogonal graph drawing algorithms. *Discrete Applied Math.*, **103**:55–87, 2000.

[69] P. Eades and S. Whitesides, Drawing graphs in two layers. *Theoret. Comput. Sci.*, **131(2)**:361–374, 1994.

[70] P. Eades and N. C. Wormald, Edge crossings in drawings of bipartite graphs. *Algorithmica*, **11(4)**:379–403, 1994.

[71] T. Endo, The pagenumber of toroidal graphs is at most seven. *Discrete Math.*, **175(1-3)**:87–96, 1997.

[72] H. Enomoto and M. S. Miyauchi, Embedding graphs into a three page book with $O(M \log N)$ crossings of edges over the spine. *SIAM J. Discrete Math.*, **12(3)**:337–341, 1999.

[73] H. Enomoto and M. S. Miyauchi, Embedding a graph into a $d + 1$-page book with $\lceil m \log_d n \rceil$ edge-crossings over the spine. *IPSJ SIGNotes ALgorithms*, **051**, 2001. Abstract No. 008.

[74] H. Enomoto, M. S. Miyauchi, and K. Ota, Lower bounds for the number of edge-crossings over the spine in a topological book embedding of a graph. *Discrete Appl. Math.*, **92(2-3)**:149–155, 1999.

[75] D. Eppstein, Separating thickness from geometric thickness. In [93], pp. 150–161.

[76] P. Erdös, Appendix. In K. F. Roth, On a problem of Heilbronn. *J. London Math. Soc.*, **26**:198–204, 1951.

[77] P. Erdös and G. Szekeres, A combinatorial problem in geometry. *Composito Math.*, **2**:464–470, 1935.

[78] S. Even and A. Itai, Queues, stacks, and graphs. In Z. Kohavi and A. Paz, eds., *Proc. International Symp. on Theory of Machines and Computations*, pp. 71–86, Academic Press, 1971.

[79] I. Fáry, On straight line representation of planar graphs. *Acta Univ. Szeged. Sect. Sci. Math.*, **11**:229–233, 1948.

[80] S. Felsner, G. Liotta, and S. Wismath, Straight-line drawings on restricted integer grids in two and three dimensions. In [150], pp. 328–342.

[81] D. R. Fulkerson and O. A. Gross, Incidence matrices and interval graphs. *Pacific J. Math.*, **15**:835–855, 1965.

[82] Z. Galil, R. Kannan, and E. Szemerédi, On 3-pushdown graphs with large separators. *Combinatorica*, **9(1)**:9–19, 1989.

[83] Z. Galil, R. Kannan, and E. Szemerédi, On nontrivial separators for k-page graphs and simulations by nondeterministic one-tape Turing machines. *J. Comput. System Sci.*, **38(1)**:134–149, 1989.

[84] J. L. Ganley, Stack and queue layouts of Halin graphs, 1995, manuscript.

[85] J. L. Ganley and L. S. Heath, The pagenumber of k-trees is $O(k)$. *Discrete Appl. Math.*, **109(3)**:215–221, 2001.

[86] M. R. Garey and D. S. Johnson, *Computers And Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[87] M. R. Garey and D. S. Johnson, Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods*, **4(3)**:312–316, 1983.

[88] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou, The complexity of coloring circular arcs and chords. *SIAM J. Algebraic Discrete Methods*, **1(2)**:216–227, 1980.

[89] A. Garg and R. Tamassia, On the computational complexity of upward and rectilinear planarity testing. In R. Tamassia and I. G. Tollis, eds., *Proc. DIMACS International Workshop on Graph Drawing (GD '94)*, vol. 894 of *Lecture Notes in Comput. Sci.*, pp. 286–297, Springer, 1995.

[90] A. Garg, R. Tamassia, and P. Vocca, Drawing with colors. In J. Diaz and M. Serna, eds., *Proc. 4th Annual European Symp. on Algorithms (ESA '96)*, vol. 1136 of *Lecture Notes in Comput. Sci.*, pp. 12–26, Springer, 1996.

[91] C. GAVOILLE AND N. HANUSSE, Compact routing tables for graphs of bounded genus. In J. WIEDERMANN, P. VAN EMDE BOAS, AND M. NIELSEN, eds., *Proc. 26th International Colloquium on Automata, Languages and Programming (ICALP '99)*, vol. 1644 of *Lecture Notes in Computer Science*, pp. 351–360, Springer, 1999.

[92] M. C. GOLUMBIC, *Algorithmic graph theory and perfect graphs*. Academic Press, 1980.

[93] M. T. GOODRICH AND S. G. KOBOUROV, eds., *Proc. 10th International Symp. on Graph Drawing (GD '02)*, vol. 2528 of *Lecture Notes in Comput. Sci.*, Springer, 2002.

[94] M. GROHE, Computing crossing numbers in quadratic time. In *Proc. 32nd Annual ACM Symp. on Theory of Computing (STOC'01)*, pp. 231–236, 2001.

[95] A. GUPTA AND N. NISHIMURA, Sequential and parallel algorithms for embedding problems on classes of partial k-trees. In *Proc. 4th Scandinavian Workshop on Algorithm Theory (SWAT '94)*, vol. 824 of *Lecture Notes in Comput. Sci.*, pp. 172–182, Springer, 1984.

[96] A. GUPTA, N. NISHIMURA, A. PROSKUROWSKI, AND P. RAGDE, Embeddings of k-connected graphs of pathwidth k. In M. M. HALLDORSSON, ed., *Proc. 7th Scandinavian Workshop on Algorithm Theory (SWAT '00)*, vol. 1851 of *Lecture Notes in Comput. Sci.*, pp. 111–124, Springer, 2000.

[97] A. GYÁRFÁS, On the chromatic number of multiple interval graphs and overlap graphs. *Discrete Math.*, **55(2)**:161–166, 1985.

[98] A. GYÁRFÁS, Corrigendum: "On the chromatic number of multiple interval graphs and overlap graphs". *Discrete Math.*, **62(3)**:333, 1986.

[99] A. GYÁRFÁS, Problems from the world surrounding perfect graphs. *Zastos. Mat.*, **19(3-4)**:413–441, 1987.

[100] R. HALIN, S-functions for graphs. *J. Geometry*, **8(1-2)**:171–186, 1976.

[101] R. HALIN, Tree-partitions of infinite graphs. *Discrete Math.*, **97**:203–217, 1991.

[102] F. HARARY, *Graph theory*. Addison-Wesley, 1969.

[103] F. HARARY AND A. SCHWENK, A new crossing number for bipartite graphs. *Utilitas Math.*, **1**:203–209, 1972.

[104] T. HARJU AND L. ILIE, Forbidden subsequences and permutations sortable on two parallel stacks. In *Where mathematics, computer science, linguistics and biology meet*, pp. 267–275, Kluwer, 2001.

[105] T. HASUNUMA, Embedding iterated line digraphs in books. *Networks*, **40(2)**:51–62, 2002.

[106] T. HASUNUMA, Laying out iterated line digraphs using queues. In [139].

[107] T. HASUNUMA AND Y. SHIBATA, Embedding de Bruijn, Kautz and shuffle-exchange networks in books. *Discrete Appl. Math.*, **78(1-3)**:103–116, 1997.

[108] P. HEALY AND A. KUUSIK, The vertex-exchange graph and its use in multi-level graph layout. In [131], pp. 205–216.

[109] P. HEALY, A. KUUSIK, AND S. LEIPERT, Characterization of level non-planar graphs by minimal patterns. In D.-Z. DU, P. EADES, V. ESTIVILL-CASTRO, X. LIN, AND A. SHARMA, eds., *Proc. Computing and Combinatorics (COCOON '00)*, Lecture Notes in Comput. Sci., pp. 74–84, Springer, 2000.

[110] L. S. HEATH, F. T. LEIGHTON, AND A. L. ROSENBERG, Comparing queues and stacks as mechanisms for laying out graphs. *SIAM J. Discrete Math.*, **5(3)**:398–412, 1992.

[111] L. S. HEATH AND S. V. PEMMARAJU, Stack and queue layouts of posets. *SIAM J. Discrete Math.*, **10(4)**:599–625, 1997.

[112] L. S. HEATH AND S. V. PEMMARAJU, Stack and queue layouts of directed acyclic graphs. II. *SIAM J. Comput.*, **28(5)**:1588–1626, 1999.

[113] L. S. HEATH, S. V. PEMMARAJU, AND A. N. TRENK, Stack and queue layouts of directed acyclic graphs. I. *SIAM J. Comput.*, **28(4)**:1510–1539, 1999.

[114] L. S. HEATH AND A. L. ROSENBERG, Laying out graphs using queues. *SIAM J. Comput.*, **21(5)**:927–958, 1992.

[115] S.-H. HONG, Drawing graphs symmetrically in three dimensions. In [150], pp. 189–204.

[116] S.-H. HONG AND P. EADES, An algorithm for finding three dimensional symmetry in series parallel digraphs. In D. LEE AND S.-H. TENG, eds., *Proc. 11th International Conf. on Algorithms and Computation (ISAAC '00)*, vol. 1969 of *Lecture Notes in Comput. Sci.*, pp. 266–277, Springer, 2000.

[117] S.-H. HONG AND P. EADES, An algorithm for finding three dimensional symmetry in trees. In J. MARKS, ed., *Proc. 8th International Symp. on Graph Drawing (GD '00)*, vol. 1984 of *Lecture Notes in Comput. Sci.*, pp. 360–371, Springer, 2001.

[118] S.-H. HONG, P. EADES, A. QUIGLEY, AND S.-H. LEE, Drawing algorithms for series-parallel digraphs in two and three dimensions. In [199], pp. 198–209.

[119] A. IMAMIYA AND A. NOZAKI, Generating and sorting permutations using restricted-deques. *Information Processing in Japan*, **17**:80–86, 1977.

[120] G. JACOBSON, Space-efficient static trees and graphs. In *Proc. 30th Annual Symp. Foundations of Comput. Sci. (FOCS '89)*, pp. 549–554, IEEE, 1989.

[121] M. JÜNGER, S. LEIPERT, AND P. MUTZEL, Level planarity testing in linear time. In [199], pp. 224–237.

[122] M. JÜNGER AND P. MUTZEL, 2-layer straightline crossing minimization: performance of exact and heuristic algorithms. *J. Graph Algorithms Appl.*, **1(1)**:1–25, 1997.

[123] P. C. KAINEN, Thickness and coarseness of graphs. *Abh. Math. Sem. Univ. Hamburg*, **39**:88–95, 1973.

[124] P. C. KAINEN AND S. OVERBAY, Book embeddings of graphs and a theorem of Whitney. Submitted.

[125] R. KANNAN, Unraveling k-page graphs. *Inform. and Control*, **66(1-2)**:1–5, 1985.

[126] M. KAUFMANN AND D. WAGNER, eds., *Drawing Graphs: Methods and Models*, vol. 2025 of *Lecture Notes in Comput. Sci.* Springer, 2001.

[127] M. KAUFMANN AND R. WIESE, Embedding vertices at points: Few bends suffice for planar graphs. In [131], pp. 165–174.

[128] A. KOSTOCHKA AND J. KRATOCHVÍL, Covering and coloring polygon-circle graphs. *Discrete Math.*, **163(1-3)**:299–305, 1997.

[129] A. V. KOSTOCHKA, Upper bounds on the chromatic number of graphs. *Trudy Inst. Mat.*, **10**:204–226, 1988.

[130] M. R. KRAMER AND J. VAN LEEUWEN, The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits. In F. P. PREPARATA, ed., *Advances in Computing Research — Volume 2: VLSI Theory*, pp. 129–146, JAI Press, Connecticut, 1985.

[131] J. KRATOCHVIL, ed., *Proc. 7th International Symp. on Graph Drawing (GD '99)*, vol. 1731 of *Lecture Notes in Comput. Sci.*, Springer, 1999.

[132] V. A. KURLIN, Three-page Dynnikov diagrams of linked 3-valent graphs. *Funktsional. Anal. i Prilozhen.*, **35(3)**:84–88, 2001.

[133] A. KUUSIK, *Integer Linear Programming Approaches to Hierarchical Graph Drawing*. Ph.D. thesis, University of Limerick, 2000.

[134] Q. H. LE TU, A planar poset which requires four pages. *Ars Combin.*, **35**:291–302, 1993.

[135] F. T. LEIGHTON AND A. L. ROSENBERG, Three-dimensional circuit layouts. *SIAM J. Comput.*, **15(3)**:793–813, 1986.

[136] T. LENGAUER, *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley, 1990.

[137] X. LI AND M. STALLMANN, New bounds on the barycenter heuristic for bipartite graph drawing. *Information Processing Letter*, **82**:293–298, 2002.

[138] Y. LIN AND X. LI, Pagenumber and treewidth. *Disc. Applied Math.*, to appear.

[139] G. LIOTTA, ed., *Proc. 11th International Symp. on Graph Drawing (GD '03)*, Lecture Notes in Comput. Sci., Springer, to appear.

[140] S. M. Malitz, Genus g graphs have pagenumber $O(\sqrt{g})$. *J. Algorithms*, **17(1)**:85–109, 1994.

[141] S. M. Malitz, Graphs with E edges have pagenumber $O(\sqrt{E})$. *J. Algorithms*, **17(1)**:71–84, 1994.

[142] M. S. Miyauchi, An $O(nm)$ algorithm for embedding graphs into a 3-page book. *Trans. IEICE*, **E77-A(3)**:521–526, 1994.

[143] B. Mohar, A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discrete Math.*, **12(1)**:6–26, 1999.

[144] B. Monien, F. Ramme, and H. Salmen, A parallel simulated annealing algorithm for generating 3D layouts of undirected graphs. In [17], pp. 396–408.

[145] H. R. Morton and E. Beltrami, Arc index and the Kauffman polynomial. *Math. Proc. Cambridge Philos. Soc.*, **123(1)**:41–48, 1998.

[146] D. J. Muder, M. L. Weaver, and D. B. West, Pagenumber of complete bipartite graphs. *J. Graph Theory*, **12(4)**:469–489, 1988.

[147] X. Muñoz, U. Unger, and I. Vrťo, One sided crossing minimization is NP-hard for sparse graphs. In [150], pp. 115–123.

[148] J. I. Munro and V. Raman, Succinct representation of balanced parentheses and static trees. *SIAM J. Comput.*, **31(3)**:762–776, 2001.

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

[150] P. Mutzel, M. Jünger, and S. Leipert, eds., *Proc. 9th International Symp. on Graph Drawing (GD '01)*, vol. 2265 of *Lecture Notes in Comput. Sci.*, Springer, 2002.

[151] P. Mutzel and R. Weiskircher, Two-layer planarization in graph drawing. In K. Y. Chwa and O. H. Ibarra, eds., *Proc. 9th Internat. Symp. on Algorithms and Computation (ISAAC '98)*, vol. 1533 of *Lecture Notes in Comput. Sci.*, pp. 69–78, Springer, 1998.

[152] H. Nagamochi, An improved approximation to the one-sided bilayer drawing. In [139].

[153] J. Nešetřil and P. Ossona de Mendez, Colorings and homomorphisms of minor closed classes. In B. Aronov, S. Basu, J. Pach, and M. Sharir, eds., *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*, vol. 25 of *Algorithms and Combinatorics*, Springer, 2003.

[154] R. Nowakowski and A. Parker, Ordered sets, pagenumbers and planarity. *Order*, **6(3)**:209–218, 1989.

[155] E. T. Ordman and W. Schmitt, Permutations using stacks and queues. In *Proc. of 24th Southeastern International Conference on Combinatorics, Graph Theory, and Computing*, vol. 96 of *Congr. Numer.*, pp. 57–64, 1993.

[156] D. I. OSTRY, *Some Three-Dimensional Graph Drawing Algorithms*. Master's thesis, Department of Computer Science and Software Engineering, The University of Newcastle, Australia, 1996.

[157] S. OVERBAY, *Generalized Book Embeddings*. Ph.D. thesis, Colorado State University, USA, 1998.

[158] J. PACH, T. THIELE, AND G. TÓTH, Three-dimensional grid drawings of graphs. In G. DI BATTISTA, ed., *Proc. 5th International Symp. on Graph Drawing (GD '97)*, vol. 1353 of *Lecture Notes in Comput. Sci.*, pp. 47–51, Springer, 1997. Also in BERNARD CHAZELLE, JACOB E. GOODMAN, and RICHARD POLLACK, eds., Advances in discrete and computational geometry, vol. 223 of *Contemporary Mathematics*, pp. 251–255, Amer. Math. Soc., 1999.

[159] J. PACH AND R. WENGER, Embedding planar graphs at fixed vertex locations. *Graphs Combin.*, **17(4)**:717–728, 2001.

[160] S. V. PEMMARAJU, *Exploring the Powers of Stacks and Queues via Graph Layouts*. Ph.D. thesis, Virginia Polytechnic Institute and State University, Virginia, U.S.A., 1992.

[161] T. PORANEN, A new algorithm for drawing series-parallel digraphs in 3D. Tech. Rep. A-2000-16, Dept. of Computer and Information Sciences, University of Tampere, Finland, 2000.

[162] V. R. PRATT, Computing permutations with double-ended queues. Parallel stacks and parallel queues. In *Proc. 5th Annual ACM Symp. on Theory of Computing (STOC '73)*, pp. 268–277, ACM, 1973.

[163] F. P. PREPARATA, Optimal three-dimensional VLSI layouts. *Math. Systems Theory*, **16**:1–8, 1983.

[164] D. RAUTENBACH AND B. REED, Tree width and graph minors. In *Topics in Graph Theory*, Cambridge University Press, to appear.

[165] S. RENGARAJAN AND C. E. VENI MADHAVAN, Stack and queue number of 2-trees. In D. DING-ZHU AND L. MING, eds., *Proc. 1st Annual International Conf. on Computing and Combinatorics (COCOON '95)*, vol. 959 of *Lecture Notes in Comput. Sci.*, pp. 203–212, Springer, 1995.

[166] N. ROBERTSON AND P. SEYMOUR, Graph minors XX: Wagner's conjecture, manuscript.

[167] N. ROBERTSON AND P. SEYMOUR, Graph minors I-XVIII. *J. Combin. Theory Ser. B*, 1983-2003.

[168] N. ROBERTSON, P. SEYMOUR, AND R. THOMAS, Sachs' linkless embedding conjecture. *J. Combin. Theory Series B*, **64**:185–227, 1995.

[169] N. ROBERTSON AND P. D. SEYMOUR, Graph minors. II. Algorithmic aspects of treewidth. *J. Algorithms*, **7(3)**:309–322, 1986.

[170] D. J. ROSE, R. E. TARJAN, AND G. S. LEUKER, Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, **5**:266–283, 1976.

[171] A. L. ROSENBERG, The DIOGENES approach to testable fault-tolerant arrays of processors. *IEEE Trans. Comput.*, **C-32**:902–910, 1983.

[172] A. L. ROSENBERG, Three-dimensional VLSI: A case study. *J. Assoc. Comput. Mach.*, **30(2)**:397–416, 1983.

[173] A. L. ROSENBERG, Book embeddings and wafer-scale integration. In *Proc. 17th Southeastern International Conference on Combinatorics, Graph Theory, and Computing*, vol. 54 of *Congr. Numer.*, pp. 217–224, 1986.

[174] A. L. ROSENBERG, DIOGENES, circa 1986. In *Proc. VLSI Algorithms and Architectures*, vol. 227 of *Lecture Notes in Comput. Sci.*, pp. 96–107, Springer, 1986.

[175] P. SCHEFFLER, *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme*. Ph.D. thesis, Akademie der Wissenschaften der DDR, Berlin, 1989.

[176] W. SCHNYDER, Planar graphs and poset dimension. *Order*, **5(4)**:323–343, 1989.

[177] D. SEESE, Tree-partite graphs and the complexity of algorithms. In L. BUDACH, ed., *Proc. International Conf. on Fundamentals of Computation Theory*, vol. 199 of *Lecture Notes in Comput. Sci.*, pp. 412–421, Springer, 1985.

[178] F. SHAHROKHI AND W. SHI, On crossing sets, disjoint sets, and pagenumber. *J. Algorithms*, **34(1)**:40–53, 2000.

[179] F. SHAHROKHI, O. SÝKORA, L. A. SZÉKELY, AND I. VRŤO, On bipartite drawings and the linear arrangement problem. *SIAM J. Comput.*, **30(6)**:1773–1789, 2001.

[180] E. STEINITZ, Polyeder und Raumeinteilungen. *Encyclopädie der Mathematischen Wissenschaften*, **3AB12**:1–139, 1922.

[181] M. SUDERMAN, Pathwidth and layered drawings of trees. Tech. Rep. SOCS-02.8, School of Computer Science, McGill University, Montreal, Canada, 2002.

[182] M. SUDERMAN AND S. WHITESIDES, Experiments with the fixed-parameter approach for two-layer planarization. In [139].

[183] K. SUGIYAMA, S. TAGAWA, AND M. TODA, Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems Man Cybernet.*, **11(2)**:109–125, 1981.

[184] R. P. SWAMINATHAN, D. GIRIRAJ, AND D. K. BHATIA, The pagenumber of the class of bandwidth-k graphs is $k-1$. *Inform. Process. Lett.*, **55(2)**:71–74, 1995.

[185] M. M. SYSŁO, Bounds to the page number of partially ordered sets. In M. NAGL, ed., *Proc. 15th International Workshop in Graph-Theoretic Concepts in Computer Science (WG '89)*, vol. 411 of *Lecture Notes in Comput. Sci.*, pp. 181–195, Springer, 1990.

[186] R. TARJAN, Sorting using networks of queues and stacks. *J. Assoc. Comput. Mach.*, **19**:341–346, 1972.

[187] M. THORUP, All structured programs have small tree-width and good register allocation. *Information and Computation*, **142(2)**:159–181, 1998.

[188] N. TOMII, Y. KAMBAYASHI, AND S. YAJIMA, On planarization algorithms of 2-level graphs. *Papers of tech. group on elect. comp., IECEJ*, **EC77-38**:1–12, 1977.

[189] W. UNGER, On the k-colouring of circle-graphs. In M. W. R. CORI, ed., *Proc. 5th International Symp. on Theoretical Aspects of Computer Science (STACS '88)*, vol. 294 of *Lecture Notes in Comput. Sci.*, pp. 61–72, Springer, 1988.

[190] W. UNGER, The complexity of colouring circle graphs. In A. FINKEL AND M. JANTZEN, eds., *Proc. 9th International Symp. on Theoretical Aspects of Computer Science (STACS '92)*, vol. 577 of *Lecture Notes in Comput. Sci.*, pp. 389–400, Springer, 1992.

[191] V. VALLS, R. MARTI, AND P. LINO, A branch and bound algorithm for minimizing the number of crossing arcs in bipartite graphs. *J. Operational Res.*, **90**:303–319, 1996.

[192] K. WAGNER, Bemerkung zum Vierfarbenproblem. *Jber. Deutsch. Math.-Verein.*, **46**:26–32, 1936.

[193] C. WARE AND G. FRANCK, Viewing a graph in a virtual reality display is three times as good as a 2D diagram. In A. L. AMBLER AND T. D. KIMURA, eds., *Proc. IEEE Symp. Visual Languages (VL '94)*, pp. 182–183, IEEE, 1994.

[194] C. WARE AND G. FRANCK, Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Trans. Graphics*, **15(2)**:121–140, 1996.

[195] C. WARE, D. HUI, AND G. FRANCK, Visualizing object oriented software in three dimensions. In *Proc. IBM Centre for Advanced Studies Conf. (CASCON '93)*, pp. 1–11, 1993.

[196] J. N. WARFIELD, Crossing theory and hierarchy mapping. *IEEE Trans. Systems Man Cybernet.*, **SMC-7(7)**:505–523, 1977.

[197] M. S. WATERMAN AND J. R. GRIGGS, Interval graphs and maps of DNA. *Bull. Math. Biol.*, **48(2)**:189–195, 1986.

[198] M. E. WATKINS, A special crossing number for bipartite graphs: A research problem. *Annals of New York Academy of Sciences*, **175**:405–410, 1970.

[199] S. WHITESIDES, ed., *Proc. 6th International Symp. on Graph Drawing (GD '98)*, vol. 1547 of *Lecture Notes in Comput. Sci.*, Springer, 1998.

[200] H. WHITNEY, A theorem on graphs. *Ann. of Math. (2)*, **32(2)**:378–390, 1931.

[201] A. WIGDERSON, The complexity of the Hamiltonian circuit problem for maximal planar graphs. Tech. Rep. EECS 198, Princeton University, USA, 1982.

[202] D. R. WOOD, Bounded degree book embeddings and three-dimensional orthogonal graph drawing. In [150], pp. 312–327.

[203] D. R. WOOD, Degree constrained book embeddings. *J. Algorithms*, **45(2)**:144–154, 2002.

[204] D. R. WOOD, Queue layouts, tree-width, and three-dimensional graph drawing. In M. AGRAWAL AND A. SETH, eds., *Proc. 22nd Foundations of Software Technology and Theoretical Computer Science (FST TCS '02)*, vol. 2556 of *Lecture Notes in Comput. Sci.*, pp. 348–359, Springer, 2002.

[205] D. R. WOOD, Optimal three-dimensional orthogonal graph drawing in the general position model. *Theoret. Comput. Sci.*, **299(1-3)**:151–178, 2003.

[206] D. R. WOOD, Grid drawings of k-colourable graphs. Submitted, 2003. See Tech. Rep. TR-2003-03, School of Computer Science, Carleton University, Ottawa, Canada.

[207] M. YANNAKAKIS, Embedding planar graphs in four pages. *J. Comput. System Sci.*, **38**:36–67, 1986.