






# Linear Layouts of Graphs with Priority Queues

Emilio Di Giacomo   

University of Perugia, Italy

Walter Didimo   

University of Perugia, Italy

Henry Förster   

Technical University of Munich, Heilbronn, Germany

Torsten Ueckerdt   

Karlsruhe Institute of Technology, Germany

Johannes Zink   

Technical University of Munich, Heilbronn, Germany

---

## Abstract

A *linear layout* of a graph consists of a linear ordering of its vertices and a partition of its edges into *pages* such that the edges assigned to the same page obey some constraint. The two most prominent and widely studied types of linear layouts are *stack* and *queue layouts*, in which any two edges assigned to the same page are forbidden to *cross* and *nest*, respectively. The names of these two layouts derive from the fact that, when parsing the graph according to the linear vertex ordering, the edges in a single page can be stored using a single stack or queue, respectively. Recently, the concepts of stack and queue layouts have been extended by using a double-ended queue or a restricted-input queue for storing the edges of a page. We extend this line of study to *edge-weighted* graphs by introducing *priority queue layouts*, that is, the edges on each page are stored in a priority queue whose keys are the edge weights. First, we show that there are edge-weighted graphs that require a linear number of priority queues. Second, we characterize the graphs that admit a priority queue layout with a single queue, regardless of the edge-weight function, and we provide an efficient recognition algorithm. Third, we show that the number of priority queues required independently of the edge-weight function is bounded by the pathwidth of the graph, but can be arbitrarily large already for graphs of treewidth two. Finally, we prove that determining the minimum number of priority queues is NP-complete if the linear ordering of the vertices is fixed.

**2012 ACM Subject Classification** Mathematics of computing → Graph algorithms; Mathematics of computing → Graph theory

**Keywords and phrases** linear layouts, recognition and characterization, priority queue layouts

**Digital Object Identifier** 10.4230/LIPIcs.WADS.2025.29

**Related Version** *Full Version*: <https://arxiv.org/abs/2506.23943> [20]

**Funding** Research of E. Di Giacomo and W. Didimo partially supported by: (i) Italian MUR PRIN Proj. 2022TS4Y3N – “EXPAND: scalable algorithms for EXPLoratory Analyses of heterogeneous and dynamic Networked Data”; (ii) Italian MUR PRIN Proj. 2022ME9Z78 – “NextGRAAL: Next-generation algorithms for constrained GRAph visuALization”; (iii) Italian MUR PON Proj. ARS01 00540 – “RASTA: Realtà Aumentata e Story-Telling Automatizzato per la valorizzazione di Beni Culturali ed Itinerari”.

**Acknowledgements** This work was initiated at the Annual Workshop on Graph and Network Visualization (GNV 2023), Chania, Greece, June 2023.



© Emilio Di Giacomo, Walter Didimo, Henry Förster, Torsten Ueckerdt, and Johannes Zink; licensed under Creative Commons License CC-BY 4.0

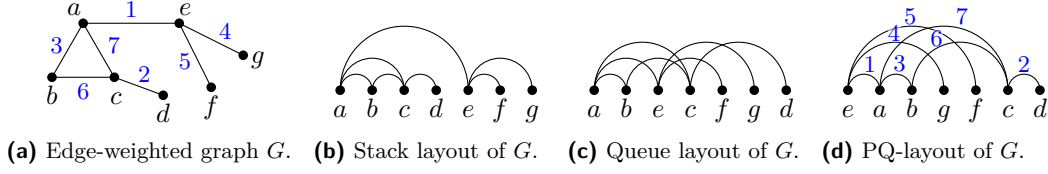
19th International Symposium on Algorithms and Data Structures (WADS 2025).

Editors: Pat Morin and Eunjin Oh; Article No. 29; pp. 29:1–29:17



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** (a)–(c) Examples of stack and queue layouts on one page for the same graph  $G$  ignoring edge weights. (d) A PQ-layout on one page of  $G$  (edge weights are in blue).

## 1 Introduction

*Stack layouts* were introduced by Bernhart and Kainen in 1979 [13] with the motivation to study a “quite natural” edge decomposition technique. A stack layout is a *linear layout*, that is, the vertices are positioned on a line  $\ell$  with the edges being embedded on several half-planes, called *pages*, delimited by  $\ell$ . In a stack layout, edges embedded on the same page are forbidden to cross, a rule that contributed to their original designation as *book embeddings*. An example of stack layout on one page is depicted in Figure 1b. The minimum number of pages required to obtain a stack layout of a graph is called the *stack number*. After their introduction, stack layouts attracted notable interest in the field of theoretical computer science due to their capacity to model certain aspects in several domains, including VLSI design [17, 18], fault-tolerant multiprocessing [18, 41], RNA folding [28], and traffic-light control [33]. This led, in particular, to a series of papers investigating upper bounds on the stack number of planar graphs [16, 29, 32] and culminating at an upper bound of four [43]. Bekos et al. [11] and Yannakakis [44] independently proved that this upper bound is tight.

The name stack layout derives from the fact that the edges assigned to the same page can be stored using a stack as follows [17]. Assume that  $\ell$  is a horizontal line and consider the left-to-right ordering  $v_1, \dots, v_n$  of the vertices. Now, consider a left-to-right sweep of  $\ell$  and the set  $E_p$  of edges assigned to a page  $p$ . Initially, the stack storing  $E_p$  is empty. At vertex  $v_i$ , all edges that have  $v_i$  as their right endpoint are popped from the stack. Since all vertices are at the same side of the half-plane delimited by  $\ell$ , the edges ending at  $v_i$  are necessarily the last ones that have started prior to encountering  $v_i$ , so they are precisely the edges on top of the stack. Afterwards, edges starting at  $v_i$  are pushed onto the stack. (Note that the order of the pushes is determined by the right endpoints of the corresponding edges.)

Based on this latter interpretation of stack layouts, Heath, Leighton and Rosenberg [30, 31] proposed to study *queue layouts* of graphs where the edges occurring on the same page of the linear layout can be represented by a queue. More precisely, edges ending at  $v_i$  are first dequeued from a queue, after which edges beginning at  $v_i$  are enqueued during the left-to-right sweep of  $\ell$ . Thus, edges on the same page are allowed to cross but forbidden to properly nest. An example of a queue layout on one page is shown in Figure 1c. Surprisingly, these quite non-planar representations still found many practical applications, including scheduling [14], VLSI [35], and, a decade after their inception, 3D crossing-free graph drawing [21, 24]. Similarly to stack layouts, one is interested in minimizing the number of pages required to obtain a queue layout, which is known as the *queue number*. However, in contrast to the stack number, the queue number of planar graphs remained elusive for several decades during which only sublinear (but non-constant) upper bounds have been found [1, 5, 6, 22]. The first significant difference between planar and non-planar graphs was only demonstrated in 2019 when it was shown that bounded degree planar graphs have bounded queue number [9], whereas non-planar bounded degree graphs were already known to not have this property [42]. Recently, Dujmović et al. [23] proved that the queue number of planar graphs is bounded by a constant. Minor improvements have since been made [10, 26].

Another interesting recent direction in investigating linear layouts is to expand beyond stack and queue layouts by using other data structures for partitioning edges. For instance, *mixed linear layouts* allow the usage of both stack and queue pages [2, 19, 38], whereas *deque layouts* use *double-ended queues* which allow to insert and pull edges from the head and the tail [3, 4, 12], and *riple layouts* use *restricted-input queues* which are double-ended queues where insertions are allowed to occur only at the head [8].

**Our contribution.** The wide range of applications of linear layouts and the fact that they have been restricted so far to unweighted graphs, naturally motivate extensions of the notion of linear layout to edge-weighted graphs. For example, in scheduling applications, the edges of a graph can represent processes, each with an associated priority; each vertex  $v$  of the graph enforces that all processes corresponding to edges incident to  $v$  begin simultaneously; among the running processes, those with higher priority must be completed first. Edge weights can also model constraints in other classical applications of linear layouts, such as logistics networks modelled as switchyard networks [34, 37], where adjacent edges represent containers that must be pushed or popped simultaneously across different storage locations. The weight of an edge corresponds to the weight of its container, and it may be necessary to ensure that containers stacked on top of others are lighter.

We introduce a model called *priority queue layout*, or simply *PQ-layout*. This model utilizes *priority queues* for the storage of edges assigned to the same page, where the edge weights correspond to the priorities (i.e., keys) for the data structure. As for stack and queue layouts, in a PQ-layout all vertices of the graph lie on a horizontal line  $\ell$ , and the edges are assigned to different priority queues according to a left-to-right sweep of  $\ell$ . When an edge  $e$  is dequeued, it must have the minimum priority among the edges stored in the same priority queue as  $e$ . See Figure 1d for an example, and refer to Section 2 for a formal definition of our model. Note that, unlike stack and queue layouts, our model allows edges in the same page to cross or properly nest, provided that they have suitable weights.

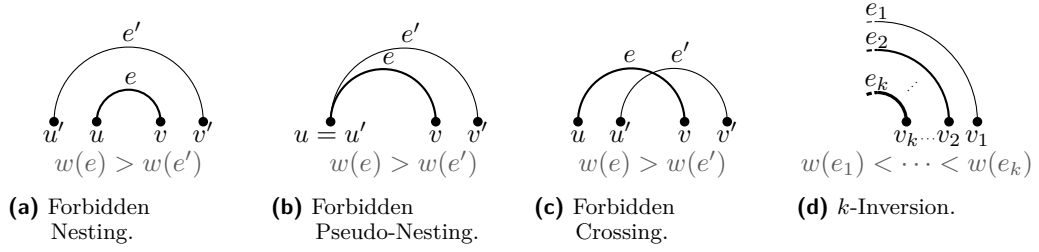
Following a current focus of research on linear layouts, we mainly study the *priority queue number* of edge-weighted graphs, that is, the number of pages required by their PQ-layouts:

(i) We provide non-trivial upper and lower bounds for the priority queue number of complete graphs (Section 3). (ii) We characterize the graphs that have priority queue number 1, regardless of the edge-weight function (Section 4). (iii) We investigate the priority queue number of graphs with bounded pathwidth and bounded treewidth (Section 5). (iv) We prove that deciding whether an edge-weighted graph has priority queue number  $k$  (for a non-fixed integer  $k$ ) is NP-complete if the linear ordering of the vertices is fixed (Section 6).

We remark that, as far as we know, there is only one previous study in the literature about using linear layouts of edge-weighted graphs [7]; however, this study uses edge weights to further restrict stack layouts. For space restrictions, proofs of statements marked with a (clickable)  $(\star)$  are omitted or sketched. We provide the missing proofs in the full version [20].

## 2 PQ-Layouts

Let  $\langle G, w \rangle$  be an *edge-weighted* graph, that is,  $G = (V, E)$  is an undirected graph and  $w: E \rightarrow \mathbb{R}$  is an *edge-weight function*. A *priority queue layout*, or simply a *PQ-layout*,  $\Gamma$  of  $\langle G, w \rangle$  consists of a linear (left-to-right) ordering  $v_1 \prec \dots \prec v_n$  of the vertices of  $G$  and a partitioning of the edges of  $G$  into  $k$  sets  $\mathcal{P}_1, \dots, \mathcal{P}_k$ , where each set is called a *page* and is associated with a priority queue. For each page  $\mathcal{P}_i$  with  $i \in \{1, \dots, k\}$ , the following must



■ **Figure 2** (a)–(c) Arrangements in a PQ-layout that correspond to the Forbidden Configuration F.1. (d) A  $k$ -inversion, a forbidden arrangement in linear layouts with priority queue number  $k - 1$ .

hold: when traversing the linear ordering  $\prec$  (from left to right) and performing, at each encountered vertex  $v \in V$ , the operations O.1 and O.2 (in this order), condition C.1 must always hold.

- O.1** Each edge  $e \in \mathcal{P}_i$  having  $v$  as its right endpoint is pulled from the priority queue. We do not count  $e$  as active any more.
- O.2** Each edge  $e \in \mathcal{P}_i$  having  $v$  as its left endpoint is inserted into the priority queue of  $\mathcal{P}_i$  with key  $w(e)$ . We say that  $e$  becomes *active*. Two edges being active at the same time (even though they may have become active in different steps) are called *co-active*.
- C.1** There is no pair of co-active edges  $e$  and  $e'$  such that  $e$  has  $v$  as its right endpoint,  $e'$  has a distinct right endpoint, and  $w(e) > w(e')$ . In other words, among the active edges of  $\mathcal{P}_i$ , the edges having  $v$  as their right endpoint have the smallest edge weights.

PQ-layouts can be equivalently defined by the absence of the forbidden configuration F.1.

**F.1** For some  $i \in \{1, \dots, k\}$ , there are two edges  $e = uv$  and  $e' = u'v'$  such that  $e, e' \in \mathcal{P}_i$ ,  $w(e) > w(e')$ , and  $u, u' \prec v \prec v'$ .

The order of  $u$  and  $u'$  is irrelevant for F.1. For completeness, the resulting three arrangements of the involved vertices are illustrated in Figures 2a–2c. Note that the forbidden pseudo-nesting in Figure 2b requires  $u = u'$ . In contrast, the symmetric case where  $v = v'$  is never forbidden. This is a direct consequence of the asymmetry between the insert and pull operations of priority queues. Thus, unlike for stack and queue layouts, the reverse vertex ordering does not necessarily provide a PQ-layout.

We refer to the linear ordering  $\prec$  of vertices also by the ordering along the *spine*. For a PQ-layout  $\Gamma$  of any graph, let  $\text{pqn}(\Gamma)$  denote the number of priority queues in  $\Gamma$ . Now, the *priority queue number*  $\text{pqn}(G, w)$  of  $\langle G, w \rangle$  is the minimum integer  $k \geq 0$  for which there exists a PQ-layout  $\Gamma$  of  $\langle G, w \rangle$  with  $\text{pqn}(\Gamma) = k$ .

For any given graph  $G$ , there always exists an edge-weight function  $w: E \rightarrow \mathbb{R}$  such that  $\text{pqn}(G, w) = 1$ . It is enough to assign the same weights to all edges but it is also possible to use distinct edge weights. Namely, define any linear ordering of the vertices  $v_1 \prec \dots \prec v_n$  and visit the vertices in this ordering. When a vertex  $v$  with  $v \neq v_1$  is visited, assign consecutive (non-negative) integer weights to all edges for which  $v$  is the right endpoint such that the weights monotonically increase. Hence, any graph with any given linear ordering of its vertices can have priority queue number 1 if we can choose the edge-weight function. However, the edge-weight function may be provided as part of the input. Thus, it is interesting to study bounds on the priority queue number of families of graphs that hold independently of the edge-weight function. To this end, for a graph  $G = (V, E)$  without any prescribed edge-weight function, the *priority queue number*  $\text{pqn}(G)$  is the minimum integer  $k \geq 0$  such that for every possible edge-weight function  $w: E \rightarrow \mathbb{R}$  we have  $\text{pqn}(G, w) \leq k$ .

Some of our proofs utilize configurations where every pair of edges must be assigned to distinct priority queues. Let  $\langle G, w \rangle$  be an edge-weighted graph, let  $\prec$  be a vertex ordering of  $G$ , and let  $k$  be an integer such that  $k \geq 1$ . We define a  $k$ -inversion to be a sequence  $(e_1, \dots, e_k)$  of  $k$  edges in  $G$  with pairwise distinct right endpoints  $v_1, \dots, v_k$ , respectively, such that (i)  $v_k \prec \dots \prec v_1$ , where  $v_k$  and  $v_1$  are called the *left* and *right end*, respectively, (ii) the left endpoint of each of  $e_1, \dots, e_k$  appears to the left of  $v_k$  in  $\prec$ , and (iii)  $w(e_1) < \dots < w(e_k)$ . See Figure 2d for an illustration. Any two edges in a  $k$ -inversion form a forbidden configuration and thus must be assigned to distinct priority queues. In other words, if, for edge weights  $w$  of  $G$ , every vertex ordering of  $\langle G, w \rangle$  contains a  $k$ -inversion, then  $\text{pqn}(G, w) \geq k$  and hence  $\text{pqn}(G) \geq k$ .

### 3 Priority Queue Number of Complete Graphs

In this section, we concentrate on complete graphs and complete bipartite graphs. We start with a simple upper bound for the complete graph  $K_n$  on  $n$  vertices.

► **Observation 1.**  $\text{pqn}(K_n) \leq n - 1$ .

**Proof.** Choose an arbitrary ordering of the vertices  $v_1 \prec \dots \prec v_n$  and define the sets (priority queues)  $\mathcal{P}_1, \dots, \mathcal{P}_{n-1}$ . For each  $j \in \{2, \dots, n\}$ , assign all edges whose right endpoint is  $v_j$  to  $\mathcal{P}_{j-1}$ . Clearly, no forbidden configuration occurs, independent of the edge-weight function. ◀

Next, we consider bipartite graphs, that is, the graphs whose vertex sets are the union of two disjoint independent sets  $A$  and  $B$ . In the literature on linear layouts, *separated* linear layouts, where vertices of  $A$  precede the vertices of  $B$  or vice versa, have been considered for bipartite graphs [25]. The *separated priority queue number*  $\text{spqn}(G)$  of a bipartite graph  $G$  is the minimum  $k$  such that, for any edge-weight function, there is a PQ-layout  $\Gamma$  that is a separated linear layout and  $\text{pqn}(\Gamma) = k$ . We can easily determine the separated priority queue number of the complete bipartite graph  $K_{m,n}$ . At the end of this section, we will use the separated priority queue number to give a lower bound on the priority queue number of complete and complete bipartite graphs.

► **Theorem 2.**  $\text{spqn}(K_{m,n}) = \min\{m, n\}$ .

**Proof.** First observe that  $\text{spqn}(K_{m,n}) \leq \min\{m, n\}$ : place the vertices of the larger set, say  $A$ , before the vertices of the smaller set, say  $B$ . Then, independent of the edge weight function, for each vertex  $b \in B$ , all edges ending at  $b$  can be assigned to a separate page.

To show  $\text{spqn}(K_{m,n}) \geq \min\{m, n\}$ , assume w.l.o.g. that the vertices in  $A$  precede the vertices in  $B$ . Consider an edge weight function where, at every vertex  $b \in B$ , each edge weight in  $\{1, 2, \dots, \min\{m, n\}\}$  occurs at least once among  $b$ 's incident edges. In any bipartite PQ-layout  $\Gamma$  of  $K_{m,n}$ , denote the vertices of  $B$  by  $b_1, b_2, \dots$  in the order they appear along the spine. In  $\Gamma$ ,  $b_1$  is incident to an edge with weight  $\min\{m, n\}$ ,  $b_2$  is incident to an edge with weight  $\min\{m, n\} - 1$ , etc.,  $b_{\min\{m, n\}}$  is incident to an edge of weight 1. This is a  $\min\{m, n\}$ -inversion requiring at least  $\min\{m, n\}$  pages. ◀

Theorem 2 also provides an upper bound for the (non-separated) priority queue number of  $K_{n,n}$ . Next, we give a corresponding linear lower bound showing that  $\text{pqn}(K_{n,n}) \in \Theta(n)$ .

► **Theorem 3 (★).**  $\text{pqn}(K_{n,n}) \geq \left\lceil \frac{3-\sqrt{5}}{4}n \right\rceil \approx 0.191n$ .

**Proof Sketch.** For  $K_{n,n}$ , we define an edge-weight function  $w$  that has, at each vertex, for every  $i \in \{1, \dots, n\}$ , exactly one incident edge of weight  $i$ .<sup>1</sup> To this end, we partition the edge set into  $n$  perfect matchings  $M_1, \dots, M_n$ , and, for each edge  $e \in M_i$ , we set  $w(e) = i$ .

For any PQ-layout  $\Gamma$  of  $\langle K_{n,n}, w \rangle$ , we can find a sublayout  $\Gamma'$  that is a *bipartite* PQ-layout of  $\langle K_{\frac{n}{2}, \frac{n}{2}}, w \rangle$  – either take the first  $\frac{n}{2}$  vertices of  $A$  and the last  $\frac{n}{2}$  vertices of  $B$  or vice versa.<sup>2</sup>

Note that we cannot directly apply Theorem 2 to  $\Gamma'$  because we have already fixed an edge-weight function. Instead, we analyze the possible distribution of the edge weights incident to vertices in  $\Gamma'$  by a representation as a black-and-white grid;<sup>3</sup> see the full version [20] for an illustration and a more extensive description. Our grid has  $\frac{n}{2}$  columns that represent the vertices of the latter partition (in order from left to right as they appear in  $\Gamma'$ ) and it has  $n$  rows that represent the edge weights  $\{1, \dots, n\}$  (in order from bottom to top). A grid cell in column  $i$  and row  $j$  is colored black if the  $i$ -th vertex is incident to an edge of weight  $j$ , and it is colored white otherwise. Observe that, in this grid, a *strictly monotonically decreasing path* of black cells having length  $k$  is equivalent to a  $k$ -inversion in  $\Gamma'$ . We can show that there always exists such a path of length  $\left\lceil \frac{3-\sqrt{5}}{4}n \right\rceil$ : the candidates for being the first black cell of the path are those who do not have another black cell in their top left region of the grid. We find these cells, color them white and iteratively repeat this process to find the candidates for being the second, third, etc. black cell of the path. We can prove that we need at least  $\left\lceil \frac{3-\sqrt{5}}{4}n \right\rceil$  iterations until the grid is white. ◀

Since  $K_n$  contains  $K_{\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor}$  as a subgraph, Theorem 3 immediately implies a linear lower bound on  $\text{pqn}(K_n)$ :

► **Corollary 4.**  $\text{pqn}(K_n) \geq \left\lfloor \frac{3-\sqrt{5}}{8}n \right\rfloor \approx 0.0955n$ .

## 4 Characterizing Graphs with Priority Queue Number 1

We study which graphs have priority queue number 1. We will provide a characterization of the graphs with priority queue number 1 in terms of forbidden minors (Section 4.4). We first show that the family of graphs with priority queue number 1 is minor-closed (Section 4.1). Then, we describe families of graphs with priority queue number 1 (Section 4.2) and a set of forbidden minors (Section 4.3). The characterization of Section 4.4 follows by proving that every graph that does not contain any of the forbidden minors of Section 4.3 falls in one of the families of Section 4.2. This characterization leads to an efficient recognition algorithm.

### 4.1 PQ-Layouts of Minors

A *minor*  $G'$  of a graph  $G = (V, E)$  is obtained from  $G$  by a series of edge contractions and by removing vertices and edges. Formally, for an edge  $e = uv$  in  $G$ , the *contraction of edge  $e$*  yields the graph  $G/e$ , obtained by replacing vertices  $u$  and  $v$  by a single vertex  $x_{uv}$  with incident edges  $\{x_{uv}y \mid uy \in E \text{ or } vy \in E\}$ . Many important graph classes (e.g., planar graphs) are *minor-closed*, that is, if we take a minor of a graph lying in such a class, the resulting graph belongs to this class as well. Every minor-closed graph class is defined by a finite set of forbidden minors and can be recognized efficiently [39, 40].

<sup>1</sup> In contrast to the proof of Theorem 2, we cannot arbitrarily assign weights to edges incident to each vertex  $b \in B$  because this may give twice the same edge weight at a vertex of partition  $A$ .

<sup>2</sup> For simplicity, we assume that  $n$  is even.

<sup>3</sup> A similar, although not identical, representation is used by Alam et al. [36] to prove the mixed page number of  $K_{n,n}$ .



► **Lemma 5** (\*). *Let  $G = (V, E)$  be a graph with  $\text{pqn}(G) = 1$  and let  $G' = G/e_0$  be obtained from  $G$  by contracting an edge  $e_0 = uv \in E$ . Then,  $\text{pqn}(G') = 1$ .*

**Proof Sketch.** We prove  $\text{pqn}(G', w') = 1$  for any given edge-weight function  $w'$ . We construct a PQ-layout  $\Gamma$  of  $\langle G, w \rangle$  using one priority queue where  $w(e) = w'(e)$  except for  $w(e_0) = W < \min_{e \in E \setminus \{e_0\}} w(e)$ . Assuming  $u \prec v$  in  $\Gamma$ , we construct a PQ-layout  $\Gamma'$  of  $\langle G', w' \rangle$ : We place vertex  $x_{uv}$  obtained from contracting  $e_0$  at the position of  $u$  in  $\Gamma$ . Vertices preceding  $u$  or succeeding  $v$  in  $\Gamma$  retain their positions in  $\Gamma'$ , and vertices between  $u$  and  $v$  in  $\Gamma$  are sorted according to the weight of their edges shared with  $x_{uv}$ . See the full version [20] for details. ◀

► **Corollary 6.** *The family of graphs with priority queue number 1 is minor-closed.*

## 4.2 Graphs with Priority Queue Number 1

We begin by showing how to construct PQ-layouts of trees:

► **Lemma 7.** *Let  $\langle T = (V, E), w \rangle$  be an  $n$ -vertex edge-weighted tree with root  $r \in V$ . Then, a PQ-layout  $\Gamma$  of  $\langle T, w \rangle$  with  $\text{pqn}(\Gamma) = 1$ , where for each vertex  $v \neq r$  its parent  $p(v)$  precedes  $v$  in the linear ordering  $\prec$  along the spine, can be constructed in  $\mathcal{O}(n \log n)$  time.*

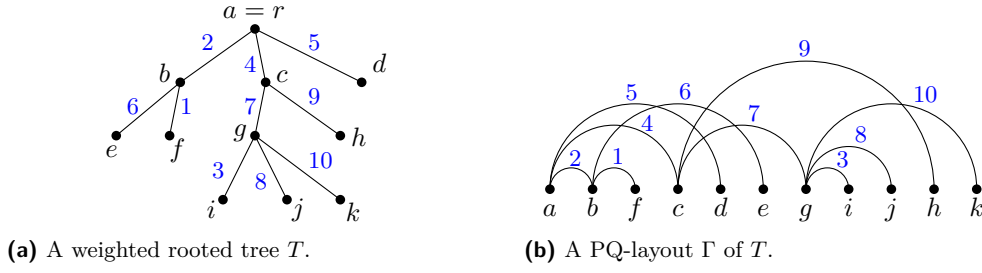
**Proof.** We describe an algorithm that constructs a PQ-layout  $\Gamma$  of  $T$ ; see Figure 3 for an example. For each vertex  $v \in V \setminus \{r\}$ , we define its weight  $w(v)$  as  $w(v) = w(vp(v))$ . As an initial layout, we first place  $r$  as the leftmost vertex and the children of  $r$  in increasing order of their weights to the right of  $r$ . During our algorithm, we keep track of the *last expanded vertex*  $v^*$ , which is the rightmost vertex whose children (if any) have already been placed. Note that initially  $v^* = r$ . We now maintain the following invariants:

- T.1** For each vertex  $v$  with  $v \preceq v^*$ , all children have already been placed.
- T.2** For each vertex  $v$  with  $v^* \prec v$ , no child has been placed yet. In addition,  $p(v) \preceq v^*$ .
- T.3** Let  $S$  denote the set of already placed vertices succeeding  $v^*$  in  $\prec$ . Then, the vertices in  $S$  occur in  $\prec$  in increasing order of their weights.
- T.4** The already constructed PQ-layout has priority queue number 1.

It is easy to see that the invariants hold for the initial layout. We now iteratively consider the vertex  $v'$  immediately succeeding  $v^*$  in  $\prec$  and perform the following operations. For each child  $c$  of  $v'$ , we place  $c$  to the right of  $v'$  such that  $S \cup \{c\} \setminus \{v'\}$  is ordered by weight. This can be easily done since by **T.3**,  $S$  (and thus also  $S \setminus \{v'\}$ ) is already ordered by weight, i.e., there is a unique position for each  $c$  and **T.3** holds again after  $v'$  has become the new  $v^*$ . Next, consider edge  $v'c$ . By **T.4**, the only way that the resulting linear layout has not priority queue number 1 is if  $v'c$  is included in a forbidden configuration. Consider an edge  $uv$  with  $u \prec v$  that is co-active with  $v'c$ . Clearly,  $v \in S$  and by **T.2**, we have that  $u \preceq v'$  and  $u = p(v)$ . By **T.3**, it follows that  $c$  and  $v$  are sorted by their weights, which are equal to  $w(v'c)$  and  $w(uv)$ , thus,  $uv$  and  $v'c$  form no forbidden configuration, i.e., **T.4** is maintained.

After placing all children of  $v'$ , we set  $v^* = v'$ . Clearly, **T.1** is maintained as we placed all children of  $v'$  whereas by induction the children of the vertices preceding  $v'$  have already been placed. Moreover, **T.2** is maintained as we only placed all children of  $v'$  to the right of  $v'$  whereas by induction the children of other vertices of  $S$  are not placed yet.

For every vertex, we need to find its position depending on the weight. This can be done in  $\mathcal{O}(\log n)$  time if we maintain a suitable search data structure. The rest of the insertion can be done in constant time. Overall, this results in a running time in  $\mathcal{O}(n \log n)$ . ◀



■ **Figure 3** A PQ-layout of a weighted rooted tree computed with the algorithm in Lemma 7.

Next, we construct a specific layout of caterpillars, which are a special subclass of trees. Namely, a *caterpillar*  $C$  consists of an *underlying path*  $P(C)$  and additional leaves, each having exactly one neighbor on  $P(C)$ . Be aware that, for the same caterpillar  $C$ , there are several choices for  $P(C)$ , and in particular  $P(C)$  might end in a degree-1 vertex of  $C$ .

► **Lemma 8.** *Let  $\langle C, w \rangle$  be an  $n$ -vertex edge-weighted caterpillar with underlying path  $P(C)$  and let  $r$  be one of the two degree-1 vertices in  $P(C)$ . Then, a PQ-layout  $\Gamma$  of  $\langle C, w \rangle$  with  $\text{pqn}(\Gamma) = 1$  where  $r$  is the rightmost vertex can be constructed in  $\mathcal{O}(n)$  time.*

**Proof.** We label  $P(C)$  as  $p_1 p_2 \dots p_k = r$  and arrange the vertices of  $P(C)$  in the order  $p_1 \prec p_2 \prec \dots \prec r$ . Then for  $i \in \{1, \dots, k\}$ , we place all leaves at  $p_i$  between  $p_{i-1}$  and  $p_i$  in an arbitrary order (note that for  $p_1$ , we place its leaves before  $p_1$ ). It is easy to see that the resulting layout contains no forbidden configuration and the statement follows. ◀

We now shift our attention to graphs containing cycles. First, we consider cycles alone.

► **Lemma 9.** *Let  $\langle C = (V, E), w \rangle$  be an  $n$ -vertex edge-weighted cycle, and let  $v$  be any given vertex of  $V$ . Then, a PQ-layout  $\Gamma$  of  $\langle C, w \rangle$  with  $\text{pqn}(\Gamma) = 1$  where  $v$  is the leftmost vertex can be constructed in  $\mathcal{O}(n)$  time.*

**Proof.** We maintain two *candidate* vertices  $a$  and  $b$  to be placed next. We call the neighbor of  $a$  ( $b$ , resp.) that has already been placed *anchor* vertex  $a'$  ( $b'$ , resp.). Initially,  $a$  and  $b$  are the two neighbors of the leftmost vertex  $v$  and  $v = a' = b'$ . We iteratively append the candidate vertex whose edge to its anchor vertex has the smallest weight to the right of the linear order (if both weights are equal, we take any of both candidates). If  $w(a'a) \leq w(b'b)$ , we append  $a$  while we do not yet place  $b$ . After appending  $a$ , we set  $a' = a$  and the other neighbor of  $a$  becomes the new candidate vertex  $a$ . We proceed symmetrically with  $b$  if  $w(a'a) > w(b'b)$ . When  $a = b$ , we place the last remaining vertex  $a$  as the rightmost vertex.

Clearly, every vertex is placed. We show that no forbidden configuration occurs. Suppose that there are two co-active edges  $uv$  (where  $u \prec v$ ) and  $xy$  (where  $x \prec y$ ) with  $w(uv) > w(xy)$  and  $v \prec y$ . In the course of our algorithm, after  $x$  and  $u$  have been placed and before  $v$  and  $y$  have been placed, the candidate vertices were  $v$  and  $y$  whose anchor vertices were  $u$  and  $x$ . There could not have been a different anchor or candidate vertex because  $C$  is a cycle. Then, however, we would not have placed  $v$  first because  $w(uv) > w(xy)$  – a contradiction.

Note that the running time of the initialization, termination and each iterative step is constant. Therefore, the overall running time is in  $\mathcal{O}(n)$ . ◀

We then consider the family of legged cycles. Namely, a *legged cycle*  $L$  consists of a single *underlying cycle*  $C(L)$  and additional leaves, each having exactly one neighbor on  $C(L)$ .



► **Lemma 10** ( $\star$ ). *Let  $\langle L, w \rangle$  be an  $n$ -vertex edge-weighted legged cycle with underlying cycle  $C(L)$ . Then, a PQ-layout  $\Gamma$  of  $\langle L, w \rangle$  with  $\text{pqn}(\Gamma) = 1$  can be constructed in  $\mathcal{O}(n \log n)$  time.*

**Proof Sketch.** We remove the heaviest edge  $e^*$  of  $C(L)$  and all leaves where the weight of the incident edge is greater than  $w(e^*)$ . We lay out the remaining graph using Lemma 8 such that the endpoints of  $e^*$  are the first and the last vertex. We reinsert  $e^*$  and, in increasing order of the weights of their incident edges, we add the initially removed leaves to the right. ◀

Next, we show that we may attach one caterpillar (Lemma 11) or two caterpillars to a single quadrangle (Lemma 12) or a single triangle (Corollary 13).

► **Lemma 11.** *Let  $\langle G, w \rangle$  be an  $n$ -vertex edge-weighted graph consisting of a cycle  $O$  and a caterpillar  $C$  with underlying path  $P(C)$  such that  $V(O) \cap V(C) = \{r\}$  is a degree-1 vertex of  $P(C)$ . Then, a PQ-layout  $\Gamma$  of  $\langle G, w \rangle$  with  $\text{pqn}(\Gamma) = 1$  can be constructed in  $\mathcal{O}(n)$  time.*

**Proof.** We first draw  $C$  and  $O$  separately using Lemmas 8 and 9, respectively. Then, we identify the two occurrences of  $r$  in  $C$  and  $O$  with each other, which yields our PQ-layout  $\Gamma$  of  $G$ . This does not result in a forbidden configuration since  $r$  is the rightmost vertex of  $C$  and the leftmost vertex of  $O$  and, hence, there is no edge spanning over  $r$ . ◀

► **Lemma 12** ( $\star$ ). *Let  $\langle G, w \rangle$  be an  $n$ -vertex edge-weighted graph consisting of:*

- *A 4-cycle  $\square = abcd$ .*
- *A caterpillar  $C_a$  with underlying path  $P(C_a)$  starting at vertex  $a \in \square$ .*
- *A caterpillar  $C_c$  with underlying path  $P(C_c)$  starting at vertex  $c \in \square$ .*

*Then, a PQ-layout  $\Gamma$  of  $\langle G, w \rangle$  with  $\text{pqn}(\Gamma) = 1$  can be constructed in  $\mathcal{O}(n \log n)$ .*

**Proof Sketch.** We arrange the cycle such that only the heaviest edge is co-active with the other three edges. We use Lemma 7 for laying out the one caterpillar with designated leftmost vertex and we use Lemma 8 for laying out the other caterpillar with designated rightmost vertex. In the full version [20], we argue that we can combine these three subgraphs, potentially adding some leaves later on, while still using only one priority queue. ◀

By Lemma 5, if  $G$  is a minor of a graph as described in Lemma 12, then also  $\text{pqn}(G) = 1$ .

► **Corollary 13.** *If  $G$  consists of a triangle  $\triangle = abc$  and two caterpillars  $C_a$  and  $C_b$  with underlying paths starting at vertex  $a \in \triangle$  and  $b \in \triangle$ , respectively, then  $\text{pqn}(G) = 1$ .*

Finally, we consider graphs that contain more than one cycle. If we exclude disconnected graphs, there exist exactly two such graphs with priority queue number 1, the complete bipartite graph  $K_{2,3}$  and the graph obtained from  $K_4$  by removing an edge. We will later see that these are the only ones.

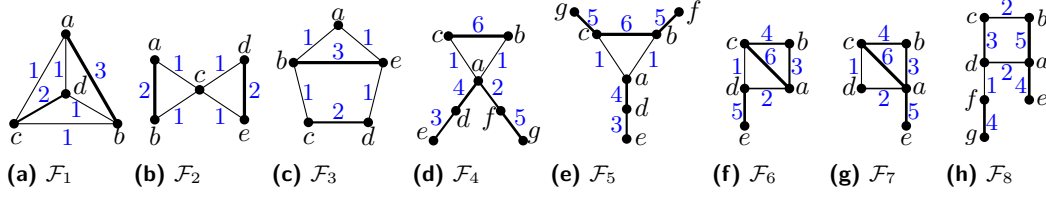
► **Lemma 14** ( $\star$ ).  $\text{pqn}(K_{2,3}) = 1$ .

In addition, by Lemma 5, we obtain the following from Lemma 14.

► **Corollary 15.**  $\text{pqn}(K_4 - e) = 1$  where  $K_4 - e$  is obtained from  $K_4$  by removing one edge.

### 4.3 Forbidden Minors for Graphs with Priority Queue Number 1

Each of the graphs  $\mathcal{F}_1, \dots, \mathcal{F}_8$  shown in Figure 4 has priority queue number at least two. The numbers at the edges specify edge-weight functions for which one priority queue does not suffice. In the full version [20], we prove the following observation:



■ **Figure 4** Forbidden minors for graphs with priority queue number 1. The graphs with the provided edge weights do not admit a linear layout with priority queue number 1. The thickness of an edge indicates the weight of that edge.

► **Lemma 16** (★). *In any PQ-layout  $\Gamma$  of an edge-weighted cycle  $\langle C, w \rangle$  with  $\text{pqn}(\Gamma) = 1$ , the last vertex on the spine is incident to an edge of maximum weight in  $\langle C, w \rangle$ .*

We now use Lemma 16 to prove the main result of this subsection:

► **Lemma 17** (★). *For each graph  $\mathcal{F} \in \{\mathcal{F}_1, \dots, \mathcal{F}_8\}$  (see Figure 4),  $\text{pqn}(\mathcal{F}) > 1$  holds.*

**Proof of Cases  $\mathcal{F}_1$ – $\mathcal{F}_3$ .** To show  $\text{pqn}(\mathcal{F}) > 1$ , it suffices to show  $\text{pqn}(\mathcal{F}, w) > 1$  for some edge weighting  $w$ . We claim that the edge weightings in Figure 4 require more than one priority queue. We consider each of the eight graphs individually and suppose for a contradiction that there is a PQ-layout  $\Gamma$  with  $\text{pqn}(\Gamma) = 1$ . We consider  $\mathcal{F}_1$ – $\mathcal{F}_3$  here and  $\mathcal{F}_4$ – $\mathcal{F}_8$  in the full version [20].

$\mathcal{F}_1$ : Consider the triangle  $\triangle abd$ . By Lemma 16,  $a$  or  $b$  is the last vertex of  $\triangle abd$  (on the spine) since  $ab$  is its heaviest edge. In the triangle  $\triangle bcd$ ,  $cd$  is the heaviest edge, which implies that  $c$  or  $d$  is the last vertex of  $\triangle bcd$ . Hence, neither  $b$  nor  $d$  can be the (overall) rightmost vertex. Since the heaviest edge of the 4-cycle  $abcd$  is again  $ab$ ,  $a$  is the rightmost vertex. In the triangle  $\triangle acd$ ,  $cd$  is the heaviest edge, which implies that  $a$  cannot be the rightmost vertex; a contradiction.

$\mathcal{F}_2$ : By Lemma 16 and symmetry, we may assume that  $a$  and  $d$  are the last vertices of  $\triangle abc$  and  $\triangle cde$ , respectively. Moreover, assume by symmetry that  $a \prec d$ , i.e.,  $c \prec a \prec d$ . However, then edge  $ab$  ends below the lighter edge  $cd$ ; a contradiction.

$\mathcal{F}_3$ : By Lemma 16,  $c$  or  $d$  is the last vertex of the 5-cycle  $abcde$ . In the 4-cycle  $bcde$ , however,  $b$  or  $e$  is the last vertex; a contradiction. ◀

## 4.4 Characterization and Recognition

We will now combine the results shown in the previous subsections to show the following:

► **Theorem 18** (★). *Let  $G$  be a graph. Then,  $\text{pqn}(G) = 1$  if and only if  $G$  does not contain any  $\mathcal{F}_i$  for  $i \in \{1, \dots, 8\}$  as a minor.*

**Proof Sketch.** We analyze the structure of  $G$  and we see that either  $G$  is a graph that has  $\text{pqn}(G) = 1$  due to a result from Section 4.2, or  $G$  has some  $\mathcal{F}_i$  for  $i \in \{1, \dots, 8\}$  as a minor.

If  $G$  is a tree, then  $\text{pqn}(G) = 1$  by Lemma 7. If  $G$  contains exactly one cycle  $C$ , then  $G - E(C)$  is a forest  $F$ , and we consider each component of  $F$  as rooted at  $C$ . If every component is an isolated vertex or a rooted star, then  $\text{pqn}(G) = 1$  by Lemma 10. If some component is more complex than a rooted star or a rooted caterpillar, then  $G$  contains  $\mathcal{F}_4$  as a minor. Now suppose some component  $K$  is a rooted caterpillar. If all other components are isolated vertices, then  $\text{pqn}(G) = 1$  by Lemma 11. If there are two further non-trivial components, then  $G$  contains  $\mathcal{F}_5$  as a minor. So assume that there is exactly one further

non-trivial component  $K' \neq K$ . If the roots of  $K$  and  $K'$  have distance at least 3 in one direction along  $C$ , then  $G$  contains  $\mathcal{F}_8$  as a minor. If none of the above applies, either  $C$  is a triangle and  $\text{pqn}(G) = 1$  by Corollary 13, or  $C$  is a quadrangle and the roots of  $K$  and  $K'$  are opposite on  $C$ , and  $\text{pqn}(G) = 1$  by Lemma 12. The case where  $G$  contains at least two cycles remains. If  $G$  has two edge-disjoint cycles, then  $G$  contains  $\mathcal{F}_2$  as a minor. If two cycles have at least two vertices but no edge in common, then  $G$  contains  $\mathcal{F}_7$  as a minor. If any two cycles share at least one edge, then either  $G = K_{2,3}$  or  $G = K_4 - e$  and we have  $\text{pqn}(G) = 1$  by Lemma 14 and Corollary 15, or  $G$  contains  $\mathcal{F}_1$ ,  $\mathcal{F}_6$ , or  $\mathcal{F}_7$  as a minor.  $\blacktriangleleft$

Theorem 18 implies that the structure of a graph  $G$  with  $\text{pqn}(G) = 1$  is quite limited. In fact, forbidden minors  $\mathcal{F}_1, \mathcal{F}_2$  and  $\mathcal{F}_3$  imply that  $G$  can contain at most three cycles. More precisely, if  $G$  contains two cycles, it is necessarily a  $K_{2,3}$  or a  $K_4 - e$  since forbidden minors  $\mathcal{F}_6$  and  $\mathcal{F}_7$  forbid any other edge. Moreover, if  $G$  contains a single cycle, forbidden minor  $\mathcal{F}_4$  implies that no non-trivial tree (i.e., a tree that is more than a caterpillar) can be attached to the cycle. Finally, forbidden minors  $\mathcal{F}_5$  and  $\mathcal{F}_8$  dictate how legs can be combined with an attached caterpillar.  $\mathcal{F}_5$  implies that at most one vertex of the cycle can have legs whereas  $\mathcal{F}_8$  implies that, for a cycle of length 4, legs can only be attached to the vertex opposite of the vertex attached to the caterpillar whereas for cycles of lengths larger than 4, no legs are allowed. These observations in conjunction with Lemmas 7, 8, 10, 12, and 14 and Corollaries 13 and 15 imply the following:

► **Corollary 19.** *Given a graph  $G$ , it can be decided in  $\mathcal{O}(n)$  time if  $\text{pqn}(G) = 1$ . Moreover, if  $\text{pqn}(G) = 1$ , a PQ-layout  $\Gamma$  of  $\langle G, w \rangle$  with  $\text{pqn}(\Gamma) = 1$  can be computed in  $\mathcal{O}(n \log n)$  time for a given edge-weight function  $w$ .*

## 5 PQ-Layouts of Graphs with Bounded Pathwidth and Treewidth

We present results for graphs of bounded pathwidth and bounded treewidth. We assume familiarity with these concepts; otherwise see the full version [20] for formal definitions.

► **Theorem 20.** *Let  $G$  be a graph with pathwidth at most  $p$ . Then,  $\text{pqn}(G) \leq p + 1$ .*

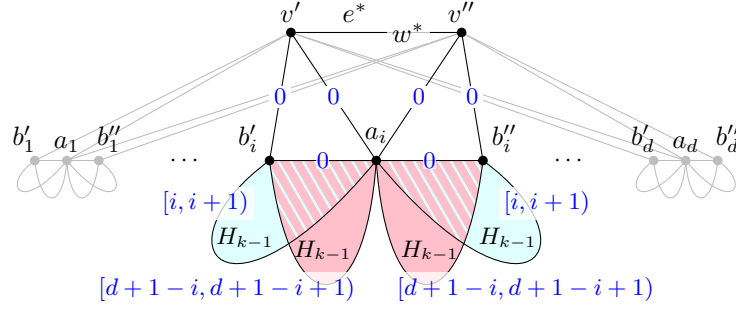
**Proof.** We may assume w.l.o.g. that  $G = (V, E)$  is edge-maximal of pathwidth  $p$ , i.e.,  $G$  is an interval graph with clique number  $\omega(G) = p + 1$  [15]. Let  $\{I_v = [a_v, b_v]\}_{v \in V}$  be an interval representation of  $G$  with distinct interval endpoints. In particular  $b_u \neq b_v$  for any  $u \neq v \in V$ .

We take the ordering  $v_1, \dots, v_n$  of vertices given by their increasing right interval endpoints. That is,  $b_{v_1} < \dots < b_{v_n}$ . Crucially, for any  $i < j < k$  with  $v_i v_k \in E$ , also  $v_j v_k \in E$ .

To define the partition of  $E$  into  $p + 1$  priority queues  $\mathcal{P}_1, \dots, \mathcal{P}_{p+1}$ , we consider a proper vertex coloring of  $G$  with  $\chi(G) = \omega(G) = p + 1$  colors. This exists, as  $G$  is an interval graph. Now for  $c = 1, \dots, p + 1$  let  $\mathcal{P}_c$  be the set of all edges in  $G$  whose right endpoint in the vertex ordering has color  $c$ . In fact, each  $\mathcal{P}_c$  contains none of the forbidden configurations in Figure 2, and hence is a priority queue independent of the edge-weights, since in each forbidden configuration in Figure 2 there would be an edge in  $G$  between the right endpoints  $v_r$  and  $v_{r'}$  of the two edges  $e$  and  $e'$  forming the forbidden configuration. But this would imply different colors for  $v_r$  and  $v_{r'}$  and therefore different priority queues for  $e$  and  $e'$ .  $\blacktriangleleft$

We remark that Corollary 4 implies that there are graphs with pathwidth  $p$  and priority queue number at least  $\left\lfloor \frac{3-\sqrt{5}}{8}(p+1) \right\rfloor$  as  $K_{p+1}$  has pathwidth  $p$ .

Next, we shall construct for every integer  $p \geq 1$  a graph  $G = G_p$  of treewidth 2 together with edge weights  $w$  so that every vertex ordering  $\sigma$  of  $G$  contains a  $p$ -inversion. If every vertex ordering of  $\langle G, w \rangle$  contains a  $p$ -inversion, then  $\text{pqn}(G, w) \geq p$  and hence  $\text{pqn}(G) \geq p$ .



■ **Figure 5** Illustration of the 2-tree  $H_k$  with edge-weighting  $w_k$ , for  $k \geq 2$ .

► **Theorem 21** (★). *For every  $p$  there is a graph  $G_p$  of treewidth 2 and  $\text{pqn}(G_p) \geq p$ .*

**Proof Sketch.** Let  $p \geq 1$  be a fixed integer. Our desired graph  $G_p$  will be a *rooted 2-tree*, i.e., an edge-maximal graph of treewidth 2 with designated root edge, which are defined inductively through the following construction sequence:

- A single edge  $e^* = v'v''$  is a 2-tree rooted at  $e^*$ .
- If  $G$  is a 2-tree rooted at  $e^*$  and  $e = uv$  is any edge of  $G$ , then adding a new vertex  $t$  with neighbors  $u$  and  $v$  is again a 2-tree rooted at  $e^*$ . We say that  $t$  is *stacked onto*  $uv$ .

We define a vertex ordering  $\sigma$  of a 2-tree rooted at  $e^* = v'v''$  to be *left-growing* if no vertex (different from  $v', v''$ ) appears to the right of both its parents. First we show that it is enough to force a  $p$ -inversion (and hence  $\text{pqn}(G) \geq p$ ) in every left-growing vertex ordering.

▷ **Claim 22** (★). *If  $\langle G, w \rangle$  is an edge-weighted rooted 2-tree so that every left-growing vertex ordering contains a  $p$ -inversion, then there exists an edge-weighted rooted 2-tree  $\langle \tilde{G}, \tilde{w} \rangle$  so that every vertex ordering contains a  $p$ -inversion.*

**Proof Sketch.** Intuitively, whenever in the construction sequence of  $G$  a vertex  $t$  is stacked onto an edge  $uv$ , in  $\tilde{G}$  we instead stack  $p^2$  “copies” of  $t$  called  $t_1, \dots, t_{p^2}$  onto  $uv$  (treating each  $t_i$  like  $t$  henceforth). The weights  $\tilde{w}(ut_i)$  and  $\tilde{w}(vt_i)$ ,  $i = 1, \dots, p^2$ , are chosen very close to  $w(ut)$  and  $w(vt)$  but increasing with  $i$  for  $ut_i$  and decreasing with  $i$  for  $vt_i$ . This way, if all  $p^2$  copies of  $t$  lie right of  $u$  and  $v$ , we get a  $p$ -inversion among the  $ut_i$ ’s or  $vt_i$ ’s. ◀

We then construct a sequence  $\langle H_1, w_1 \rangle, \langle H_2, w_2 \rangle, \dots$  of edge-weighted rooted 2-trees, so that for every  $k$ , every left-growing vertex ordering of  $\langle H_k, w_k \rangle$  contains a  $k$ -inversion. The construction of  $\langle H_k, w_k \rangle$  is recursive, starting with  $\langle H_1, w_1 \rangle$  being just a single edge of any weight. For  $k \geq 2$ , we start with  $d = k^2$  vertices  $a_1, \dots, a_d$  stacked onto the root edge  $e^* = v'v''$ , and stacking a vertex  $b'_i$  onto each  $v'a_i$ , as well as a vertex  $b''_i$  onto  $v''a_i$  for  $i = 1, \dots, d$ . All these edges have weight 0. Then, each  $b'_ia_i$  and  $b''_ia_i$  is used as the base edge of two separate copies of  $\langle H_{k-1}, w_{k-1} \rangle$ , where however the edge-weights  $w_{k-1}$  of each copy are scaled and shifted to be in a specific half-open interval  $[x, y) \subset \mathbb{R}$  depending on the current base edge ( $b'_ia_i$  or  $b''_ia_i$ ). See Figure 5 for an illustration.

By induction, there is a  $(k-1)$ -inversion in each such copy of  $H_{k-1}$ . Using the pigeon-hole principle, and the Erdős–Szekeres theorem, we then identify an edge  $a_iv'$  (or  $a_iv''$ ) which forms a  $k$ -inversion either together with  $(k-1)$ -inversion of one copy of  $H_{k-1}$ , or with  $k-1$  edges, each from the  $(k-1)$ -inversion of a different copy of  $H_{k-1}$ .

To finish the proof, it is then enough to take  $\langle H_p, w_p \rangle$  as constructed above, and then apply Claim 22 to obtain the desired edge-weighted 2-tree  $\langle G_p, w_p \rangle$  with  $\text{pqn}(G_p, w_p) \geq p$ . ◀

As graphs of treewidth 2 are always planar graphs, we conclude with the following result that contrasts similar results for the queue and stack number [23, 43]:

► **Corollary 23.** *The priority queue number of planar graphs is unbounded. In particular, there is a planar graph  $G$  with  $n$  vertices and  $\text{pqn}(G) \geq \log_4 n$ .*

**Proof.** We observe that  $H_k$  in the proof of Theorem 21 contains 4 copies of  $H_{k-1}$  while  $H_1$  contains 2 vertices. Thus, there are at least  $4^p$  vertices in  $G := H_p$ , which has treewidth  $p$ . ◀

## 6 Complexity of Fixed-Ordering PQ-Layouts

In the light of Corollary 19, one might wonder whether deciding if a given graph  $G$  has priority queue number  $k$  is polynomial-time solvable for all values of  $k$ . However, we show that it is NP-complete if the ordering of the vertices is already fixed.

► **Theorem 24.** *Given an edge-weighted graph  $\langle G, w \rangle$  with  $G = (V, E)$ , a linear ordering  $\sigma$  of  $V$ , and a positive integer  $k$ , it is NP-complete to decide whether  $\langle G, w \rangle$  admits a PQ-layout with linear ordering  $\sigma$  and a partitioning of  $E$  into  $k$  priority queues.*

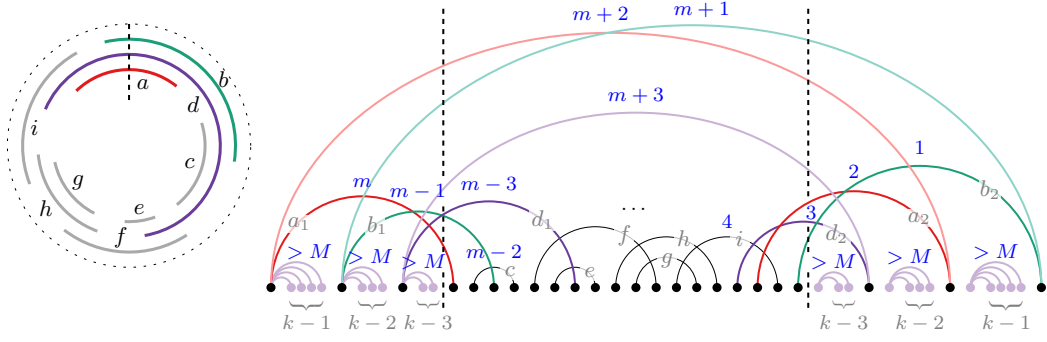
**Proof.** Containment in NP is clear as we can check in polynomial time whether a given assignment of edges to  $k$  priority queues corresponds to a valid PQ-layout.

We show NP-hardness by reduction from the problem of coloring circular-arc graphs, which is known to be NP-complete if  $k$  is not fixed [27]. A circular-arc graph is the intersection graph of arcs of a circle (see Figure 6 on the left), that is, the arcs are the vertices and two vertices share an edge if and only if the corresponding arcs share a point along the circle.

For a given circular-arc graph  $H$ , we next describe how to obtain an edge-weighted graph  $\langle G, w \rangle$  and a vertex ordering  $\sigma$  such that a  $k$ -coloring of  $H$  directly corresponds to a PQ-layout of  $\langle G, w \rangle$  under  $\sigma$  with  $k$  priority queues. Without loss of generality, we assume that we have a circular-arc representation of  $H$  where all endpoints of the arcs are distinct.

We “cut” the circle at some point, which gives an interval representation where some vertices  $S$  of  $H$  refer to two intervals; in Figure 6, the circular arcs  $a$ ,  $b$ , and  $d$  are cut into intervals  $a_1$  and  $a_2$ ,  $b_1$  and  $b_2$ , and  $d_1$  and  $d_2$ , respectively. We let the endpoints of all intervals be the vertices of  $G$ , whose ordering  $\sigma$  along the spine is taken from the ordering of the corresponding endpoints of the intervals (for the intervals belonging to  $S$ , the order is arbitrary). Further, for each interval, we have an edge in  $G$  connecting its two endpoints. To obtain  $w$ , we assign to the edges of  $G$  the numbers 1 to  $m$ , where  $m = |V(H)| + |S|$ , in decreasing order of the right endpoints of the intervals. This way, every pair of edges whose corresponding intervals share a common point need to be assigned to distinct priority queues.

So far, there is no mechanism that assures that  $a_1$  and  $a_2$ ,  $b_1$  and  $b_2$ , etc., are assigned to the same priority queue. To this end, we add, for each such pair  $\langle a_1, a_2 \rangle$ , an edge from the left endpoint of  $a_1$  to the right endpoint of  $a_2$ . We call these  $|S|$  new edges *synchronization edges*, and we assign them the weights  $m + 1, \dots, m + |S|$  in decreasing order of their right endpoints. Furthermore, we add  $k - 1, k - 2, \dots$  *heavy* edges below the leftmost, second leftmost,  $\dots$  vertex on the spine, respectively, as well as,  $k - 1, k - 2, \dots$  heavy edges below the rightmost, second rightmost,  $\dots$  vertex on the spine, respectively, see Figure 6. Clearly,  $k \geq |S|$ , otherwise we have a no-instance. The heavy edges have distinct right endpoints and their weights are chosen such that (i) they are greater than  $M$ , where  $M = m + |S|$  is the largest weight used so far, and (ii) they are chosen in decreasing order from left to right such that each pair of heavy edges needs to be assigned to distinct priority queues if they overlap.



■ **Figure 6** Illustration of our reduction.

▷ **Claim 25** (\*). Let  $\Gamma$  be a PQ-layout of  $\langle G, w \rangle$  under vertex ordering  $\sigma$  having  $k$  priority queues. For each vertex  $a \in S$  ( $\subseteq V(H)$ ), the two corresponding edges  $a_1$  and  $a_2$  in  $G$  are assigned to the same priority queue in  $\Gamma$ .

**Proof Sketch.** Consider the vertices of  $G$  from the outside to the inside. At the outer endpoint of  $a_1$  ( $a_2$ , resp.), the edge  $a_1$  ( $a_2$ , resp.) and the synchronization edge get the same color by the pigeonhole principle because all but one priority queue is occupied by the heavy edges and the previously considered edges, which all induce pairwise forbidden configurations. ◁

If we have a PQ-layout of  $\langle G, w \rangle$  under vertex ordering  $\sigma$  with  $k$  priority queues, we obtain, due to the choice of  $w$  and  $\sigma$ , a coloring of  $H$  by using the priority queues as colors. In particular, due to Claim 25, both parts of the “cut” circular arcs get the same color. Conversely, we can easily assign the edges of  $\langle G, w \rangle$  to  $k$  priority queues if we are given a  $k$ -coloring of  $H$ . Clearly, our reduction can be implemented to run in polynomial time. ◀

## 7 Open Problems

We conclude by summarizing a few intriguing problems that remain open. (i) Improve the upper and lower bounds for  $\text{pqn}(K_n)$ . (ii) Determine the complexity of deciding  $\text{pqn}(G) \leq k$  for  $k > 1$ , or deciding  $\text{pqn}(G, w) \leq k$  for a given edge weighting  $w$  when  $k$  is a constant or no vertex ordering is given. (iii) Find graph families with bounded priority queue number; what about outerplanar graphs? (iv) Can the edge density of a graph  $G$  be upper bounded by a term in  $\text{pqn}(G)$ ? (v) One can also investigate the structure of edge-weighted graphs  $\langle G, w \rangle$  with a fixed vertex-ordering  $\prec$ . Edge-partitions of  $G$  into  $k$  priority queues are equivalent to proper  $k$ -colorings of an associated graph  $H$ , while  $t$ -inversions correspond to  $t$ -cliques in  $H$ . Is  $\chi(H)$  bounded by a function in terms of  $\omega(H)$ ? Is determining  $\omega(H)$  NP-complete?

## References

- 1 Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. Queue layouts of planar 3-trees. *Algorithmica*, 82(9):2564–2585, 2020. doi:10.1007/s00453-020-00697-4.
- 2 Patrizio Angelini, Michael A. Bekos, Philipp Kindermann, and Tamara Mchedlidze. On mixed linear layouts of series-parallel graphs. *Theor. Comput. Sci.*, 936:129–138, 2022. doi:10.1016/j.tcs.2022.09.019.
- 3 Christopher Auer, Christian Bachmaier, Franz-Josef Brandenburg, Wolfgang Brunner, and Andreas Gleißner. Plane drawings of queue and deque graphs. In Ulrik Brandes and Sabine Cornelsen, editors, *Proc. 18th International Symposium on Graph Drawing and Network Visualization (GD 2010)*, volume 6502 of *Lecture Notes in Computer Science*, pages 68–79. Springer, 2010. doi:10.1007/978-3-642-18469-7\_7.



- 4 Christopher Auer and Andreas Gleißner. Characterizations of deque and queue graphs. In Petr Kolman and Jan Kratochvíl, editors, *Proc. 37th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2011)*, volume 6986 of *Lecture Notes in Computer Science*, pages 35–46. Springer, 2011. doi:10.1007/978-3-642-25870-1\_5.
- 5 Michael J. Bannister, William E. Devanny, Vida Dujmović, David Eppstein, and David R. Wood. Track layouts, layered path decompositions, and leveled planarity. *Algorithmica*, 81(4):1561–1583, 2019. doi:10.1007/s00453-018-0487-5.
- 6 Giuseppe Di Battista, Fabrizio Frati, and János Pach. On the queue number of planar graphs. *SIAM J. Comput.*, 42(6):2243–2285, 2013. doi:10.1137/130908051.
- 7 Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Marco Tais. Schematic representation of large biconnected graphs. *J. Graph Algorithms Appl.*, 25(1):311–352, 2021. doi:10.7155/jgaa.00560.
- 8 Michael A. Bekos, Stefan Felsner, Philipp Kindermann, Stephen G. Kobourov, Jan Kratochvíl, and Ignaz Rutter. The rique-number of graphs. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Proc. 30th International Symposium on Graph Drawing and Network Visualization (GD 2022)*, volume 13764 of *Lecture Notes in Computer Science*, pages 371–386. Springer, 2022. doi:10.1007/978-3-031-22203-0\_27.
- 9 Michael A. Bekos, Henry Förster, Martin Gronemann, Tamara Mchedlidze, Fabrizio Montecchiani, Chrysanthi N. Raftopoulou, and Torsten Ueckerdt. Planar graphs of bounded degree have bounded queue number. *SIAM J. Comput.*, 48(5):1487–1502, 2019. doi:10.1137/19M125340X.
- 10 Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. An improved upper bound on the queue number of planar graphs. *Algorithmica*, 85(2):544–562, 2023. doi:10.1007/s00453-022-01037-4.
- 11 Michael A. Bekos, Michael Kaufmann, Fabian Klute, Sergey Pupyrev, Chrysanthi N. Raftopoulou, and Torsten Ueckerdt. Four pages are indeed necessary for planar graphs. *J. Comput. Geom.*, 11(1):332–353, 2020. doi:10.20382/jocg.v11i1a12.
- 12 Michael A. Bekos, Michael Kaufmann, Maria Eleni Pavlidi, and Xenia Rieger. On the deque and rique numbers of complete and complete bipartite graphs. In Denis Pankratov, editor, *Proc. 35th Canadian Conference on Computational Geometry (CCCG 2023)*, pages 89–95, 2023. URL: [https://wadscg2023.ensc.concordia.ca/assets/pdf/CCCG\\_2023\\_proc.pdf#section.0.11](https://wadscg2023.ensc.concordia.ca/assets/pdf/CCCG_2023_proc.pdf#section.0.11).
- 13 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *J. Comb. Theory, Ser. B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.
- 14 Sandeep N. Bhatt, Fan R. K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. Scheduling tree-dags using FIFO queues: A control-memory trade-off. *J. Parallel Distributed Comput.*, 33(1):55–68, 1996. doi:10.1006/jpdc.1996.0024.
- 15 Hans L. Bodlaender. A partial  $k$ -arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.
- 16 Jonathan F. Buss and Peter W. Shor. On the pagewidth of planar graphs. In Richard A. DeMillo, editor, *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC 1984)*, pages 98–100. ACM, 1984. doi:10.1145/800057.808670.
- 17 Fan R. K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. DIOGENES: A methodology for designing fault-tolerant VLSI processor arrays. In *Proc. 13th Int. Conf. Fault-Tolerant Computing*, pages 26–32, 1983.
- 18 Fan R. K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. Embedding graphs in books: A layout problem with applications to VLSI design. *SIAM Journal on Algebraic Discrete Methods*, 8(1):33–58, 1987. doi:10.1137/0608002.
- 19 Philipp de Col, Fabian Klute, and Martin Nöllenburg. Mixed linear layouts: Complexity, heuristics, and experiments. In Daniel Archambault and Csaba D. Tóth, editors, *Proc. 27th International Symposium on Graph Drawing and Network Visualization (GD 2019)*, volume 11904 of *Lecture Notes in Computer Science*, pages 460–467. Springer, 2019. doi:10.1007/978-3-030-35802-0\_35.

- 20 Emilio Di Giacomo, Walter Didimo, Henry Förster, Torsten Ueckerdt, and Johannes Zink. Linear layouts of graphs with priority queues. *arXiv preprint*, 2025. doi:10.48550/arXiv.2506.23943.
- 21 Emilio Di Giacomo, Giuseppe Liotta, and Henk Meijer. Computing straight-line 3D grid drawings of graphs in linear volume. *Comput. Geom.*, 32(1):26–58, 2005. doi:10.1016/j.comgeo.2004.11.003.
- 22 Vida Dujmović. Graph layouts via layered separators. *J. Comb. Theory, Ser. B*, 110:79–89, 2015. doi:10.1016/j.jctb.2014.07.005.
- 23 Vida Dujmović, Gwenaël Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David R. Wood. Planar graphs have bounded queue-number. *J. ACM*, 67(4):22:1–22:38, 2020. doi:10.1145/3385731.
- 24 Vida Dujmović, Pat Morin, and David R. Wood. Layout of graphs with bounded tree-width. *SIAM J. Comput.*, 34(3):553–579, 2005. doi:10.1137/S0097539702416141.
- 25 Vida Dujmović, Attila Pór, and David R. Wood. Track layouts of graphs. *Discret. Math. Theor. Comput. Sci.*, 6(2):497–522, 2004. doi:10.46298/DMTCS.315.
- 26 Henry Förster, Michael Kaufmann, Laura Merker, Sergey Pupyrev, and Chrysanthi N. Raftopoulou. Linear layouts of bipartite planar graphs. In Pat Morin and Subhash Suri, editors, *Proc. 18th Algorithms and Data Structures Symposium (WADS 2023)*, volume 14079 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2023. doi:10.1007/978-3-031-38906-1\_29.
- 27 M. R. Garey, David S. Johnson, Gerald L. Miller, and Christos H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM J. Algebraic Discret. Methods*, 1(2):216–227, 1980. doi:10.1137/0601025.
- 28 Christian Haslinger and Peter F. Stadler. RNA structures with pseudo-knots: Graph-theoretical, combinatorial, and statistical properties. *Bull. Math. Biol.*, 61(3):437–467, 1999. doi:10.1006/bulm.1998.0085.
- 29 Lenwood S. Heath. Embedding planar graphs in seven pages. In *Proc. 25th Annual Symposium on Foundations of Computer Science (FOCS 1984)*, pages 74–83. IEEE Computer Society, 1984. doi:10.1109/SFCS.1984.715903.
- 30 Lenwood S. Heath, Frank Thomson Leighton, and Arnold L. Rosenberg. Comparing queues and stacks as mechanisms for laying out graphs. *SIAM J. Discret. Math.*, 5(3):398–412, 1992. doi:10.1137/0405031.
- 31 Lenwood S. Heath and Arnold L. Rosenberg. Laying out graphs using queues. *SIAM J. Comput.*, 21(5):927–958, 1992. doi:10.1137/0221055.
- 32 Sorin Istrail. An algorithm for embedding planar graphs in six pages. *Iasi University Annals, Mathematics-Computer Science*, 34(4):329–341, 1988.
- 33 Paul C. Kainen. The book thickness of a graph, II. *Congressus Numerantium*, 71:127–132, January 1990.
- 34 Donald E. Knuth. *The Art of Computer Programming, Volume I: Fundamental Algorithms*. Addison-Wesley, 1968.
- 35 Frank Thomson Leighton and Arnold L. Rosenberg. Three-dimensional circuit layouts. *SIAM J. Comput.*, 15(3):793–813, 1986. doi:10.1137/0215057.
- 36 Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. The mixed page number of graphs. *Theoretical Computer Science*, 931:131–141, 2022. doi:10.1016/j.tcs.2022.07.036.
- 37 Vaughan R. Pratt. Computing permutations with double-ended queues, parallel stacks and parallel queues. In Alfred V. Aho, Allan Borodin, Robert L. Constable, Robert W. Floyd, Michael A. Harrison, Richard M. Karp, and H. Raymond Strong, editors, *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 268–277. ACM, 1973. doi:10.1145/800125.804058.
- 38 Sergey Pupyrev. Mixed linear layouts of planar graphs. In Fabrizio Frati and Kwan-Liu Ma, editors, *Proc. 25th International Symposium on Graph Drawing and Network Visualization (GD 2017)*, volume 10692 of *Lecture Notes in Computer Science*, pages 197–209. Springer, 2017. doi:10.1007/978-3-319-73915-1\_17.

- 39 Neil Robertson and Paul D. Seymour. Graph minors. XIII. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/JCTB.1995.1006.
- 40 Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner’s conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004. doi:10.1016/J.JCTB.2004.08.001.
- 41 Arnold L. Rosenberg. The diogenes approach to testable fault-tolerant arrays of processors. *IEEE Trans. Computers*, C-32(10):902–910, 1983. doi:10.1109/TC.1983.1676134.
- 42 David R. Wood. Bounded-degree graphs have arbitrarily large queue-number. *Discret. Math. Theor. Comput. Sci.*, 10(1), 2008. doi:10.46298/dmtcs.434.
- 43 Mihalis Yannakakis. Embedding planar graphs in four pages. *J. Comput. Syst. Sci.*, 38(1):36–67, 1989. doi:10.1016/0022-0000(89)90032-9.
- 44 Mihalis Yannakakis. Planar graphs that need four pages. *J. Comb. Theory, Ser. B*, 145:241–263, 2020. doi:10.1016/j.jctb.2020.05.008.