Algorithmica (2020) 82:2564–2585
https://doi.org/10.1007/s00453-020-00697-4

Queue Layouts of Planar 3-Trees

Jawaherul Md. Alam[1] · Michael A. Bekos[2] · Martin Gronemann[3] · Michael Kaufmann[2] · Sergey Pupyrev[1]

Received: 6 October 2018 / Accepted: 6 March 2020 / Published online: 23 March 2020

Abstract

A *queue layout* of a graph G consists of a *linear order* of the vertices of G and a partition of the edges of G into *queues*, so that no two independent edges of the same queue are nested. The *queue number* of graph G is defined as the minimum number of queues required by any queue layout of G. In this paper, we continue the study of the queue number of planar 3-trees, which form a well-studied subclass of planar graphs. Prior to this work, it was known that the queue number of planar 3-trees is at most seven. In this work, we improve this upper bound to five. We also show that there exist planar 3-trees whose queue number is at least four. Notably, this is the first example of a planar graph with queue number greater than three.

Keywords Queue layouts · Constant queue number · Planar 3-trees

A preliminary version of this article has appeared in the proceedings of the 26th International Symposium on Graph Drawing and Network Visualization (GD 2018). This work is supported by the DFG Grant Ka812/17-1 and DAAD Project 57419183.

✉ Michael A. Bekos
bekos@informatik.uni-tuebingen.de

Jawaherul Md. Alam
jawaherul@gmail.com

Martin Gronemann
gronemann@informatik.uni-koeln.de

Michael Kaufmann
mk@informatik.uni-tuebingen.de

Sergey Pupyrev
spupyrev@gmail.com

[1] Department of Computer Science, University of Arizona, Tucson, USA

[2] Institut für Informatik, Universität Tübingen, Tübingen, Germany

[3] Institut für Informatik, Universität zu Köln, Köln, Germany

1 Introduction

In a *queue* layout [19], the vertices of a graph are restricted to a line and its edges are drawn at different half-planes delimited by this line, called *queues*. The task is to find a linear order of the vertices along the underlying line and a corresponding assignment of the edges of the graph to the queues, so that no two independent edges of the same queue are nested; see Fig. 1 for an illustration. Recall that two edges are called *independent*, if they do not share an endvertex. The *queue number* of a graph is the smallest number of queues that are required by any queue layout of the graph. Note that queue layouts form the "dual" concept of *stack* layouts [22] (widely known also as *book embeddings*), which do not allow two edges of the same stack to cross.

Apart from the intriguing theoretical interest, queue layouts find applications in several domains [3, 18, 23, 29]. As a result, they have been studied extensively over the years [5, 8, 13, 17, 19, 24, 25, 27–30]. The most remarkable result in this area is due to Dujmović et al. [9], who recently proved that planar graphs have constant queue number (by the time of writing at most 49) improving upon a series of results [2, 5, 7] and thus settling in the positive an old conjecture by Heath, Leighton and Rosenberg [18]. Notably, this breakthrough result has several important implications, e.g., that (i), (ii) every n-vertex planar graph admits a $\mathcal{O}(1) \times \mathcal{O}(1) \times \mathcal{O}(n)$ straight-line grid drawing [31], (iii) every Hamiltonian bipartite planar graph admits a 2-layer drawing and an edge-coloring of bounded size, in which edges of the same color do not cross [12], and (iv) the queue number of k-planar graphs is also bounded by a constant for fixed values of k [13].

Improved upper bounds on the queue number are known for several subclasses of planar graphs. Every tree has queue number one [19], outerplanar graphs have queue number at most two [18], and series-parallel graphs have queue number at most three [27]. Planar 3-trees have queue number at most seven [30], although they were conjectured to have super-constant queue number by Pemmaraju [24]; recall that a *planar 3-tree* is a triangulated plane graph G with $n \geq 3$ vertices, such that G is either a 3-cycle, if $n = 3$, or has a vertex whose deletion gives a planar 3-tree with $n - 1$ vertices, if $n > 3$. As a matter of fact, every graph that admits a 1-queue layout is planar with at most $2n - 3$ edges; however, testing this property is an $\mathcal{NP}$ -complete problem [18]; for a survey that also covers results for non-planar graphs the interested reader is referred to [13].

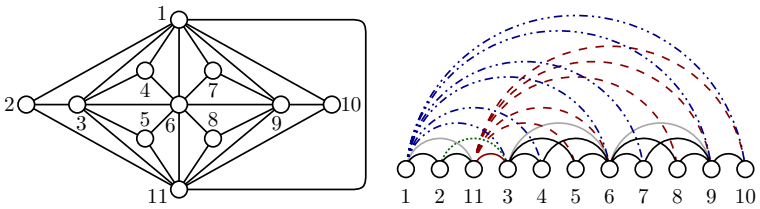


Fig. 1 **a** The Goldner-Harary planar 3-tree, and **b** a 5-queue layout of it produced by our algorithm, in which edges of different queues are colored differently

Table 1 Queue numbers of various subclasses of planar graphs

Graph class	Upper bound		Lower bound	
	Old	New	Old	New
Tree	1 [19]		1 [19]	
Outerplanar	2 [18]		2 [19]	
Series-parallel	3 [27]		3 [30]	
Planar 3-tree	7 [30]	5 [Thm. 1]	3 [30]	4 [Thm. 2]
Planar	49 [9]		3 [30]	4 [Thm. 2]

Our contribution. In this paper, we present improved upper and lower bounds on the queue number of planar 3-trees. More precisely, we first discuss in Sect. 2 known results that are useful for our work. In Sect. 3, we improve the upper bound on the queue number of planar 3-trees from seven [30] to five. In Sect. 4, we show that there exist planar 3-trees whose queue number is at least four, thus strengthening a corresponding result of Wiechert [30] for general (that is, not necessarily planar) 3-trees; we stress that our lower bound is also the best known for planar graphs. Table 1 puts our results in the context of existing bounds. In Sect. 5, we discuss implications of our results to the closely-related *track layouts* [11]. Our work leads to a number of interesting research questions, which we list in Sect. 6.

Remark The core of the approach introduced by Dujmović et al. [9] consists of partitioning the input planar graph into sets of vertices, called bags, such that the graph induced by the bags is planar and has treewidth at most 3. The queue number of the input graph is then upper bounded by an expression that depends on the queue number of planar 3-trees. It is worth noting that the authors of [9] use our main result to obtain the bound on the queue number of planar graphs, as paper [9] appeared after the conference version [1] of this work.

2 Preliminaries

In this section, we present preliminary notions and notation that is used throughout this paper. We also present known results from the literature that are useful for our work. We assume familiarity with basic graph theoretic notions (see, e.g., [16]). Also, for standard definitions on planar graphs and drawings, we point the reader to [4, 20].

For a pair of distinct vertices u and v of a graph G, we write $u \prec v$ if u precedes v in a linear order of G. We also write $[v_1, v_2, \ldots, v_k]$ to denote that v_i precedes v_{i+1} for all $1 \leq i < k$. Let F be a set of $k \geq 2$ independent edges (s_i, t_i), that is, $F = \{(s_i, t_i); i = 1, 2, \ldots, k\}$. Assume without loss of generality that $s_i \prec t_i$, for all $1 \leq i \leq k$. If the linear order is $[s_1, \ldots, s_k, t_k, \ldots, t_1]$, then we say that F is a *k-rainbow*, while if the linear order is $[s_1, \ldots, s_k, t_1, \ldots, s_k]$, we say that F is a *k-twist*.

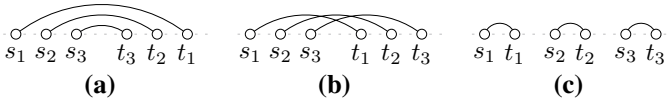


Fig. 2 Illustration of: **a** a 3-rainbow, **b** a 3-twist, and **c** a 3-necklace

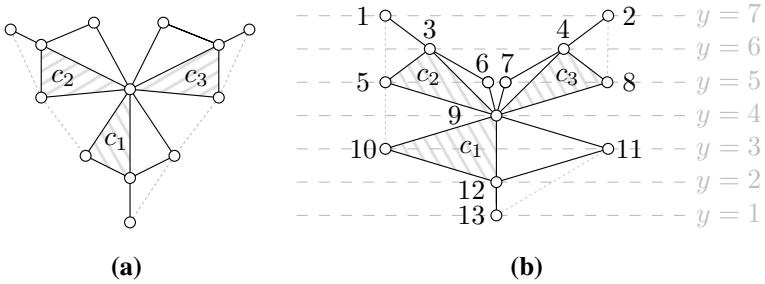


Fig. 3 Illustrations of: **a** an internally-triangulated outerplane graph G_0; the dotted-gray edges have been added to make it biconnected; its gray-shaded faces contain components c_1, c_2 and c_3 of the graph G_1 formally introduced Section 3; **b** the drawing $\Gamma(G_0)$ by Lemma 2; the vertex-labels indicate the linear order of its 2-queue layout; the anchor vertices of faces $\langle 9, 10, 12\rangle$, $\langle 3, 5, 9\rangle$ and $\langle 4, 8, 9\rangle$ are 10, 5, 8, respectively

The edges of F form a *k-necklace*, if $[s_1, t_1, \ldots, s_k, t_k]$; see Fig. 2. A preliminary result for queue layouts is the following.

Lemma 1 (Heath and Rosenberg [19]) *A linear order of the vertices of a graph admits a k-queue layout if and only if there exists no* $(k + 1)$ *-rainbow.*

Central in our approach is also the following construction by Dujmović et al. [11] for internally-triangulated outerplane graphs; for an illustration refer to Figs. 3a, b.

Lemma 2 (Dujmović et al. [11]) *Every internally-triangulated outerplane graph G admits a straight-line outerplanar drawing* $\Gamma(G)$*, such that the y -coordinates of the vertices of G are integers, and the absolute value of the difference of the y -coordinates of the endvertices of each edge of G is either one or two. Furthermore, the drawing can be used to construct a* 2*-queue layout of G.*

Let $\langle u, v, w\rangle$ be a face of a drawing $\Gamma(G)$ produced by the constructive algorithm supporting Lemma 2, where G is an internally triangulated outerplane graph. Up to renaming of the vertices of this face, we may assume that $y(v) > y(w)$, $y(v) - y(u) = y(u) - y(w) = 1$ and $y(v) - y(w) = 2$. We refer to vertex u as to the *anchor* vertex of the face $\langle u, v, w\rangle$ of $\Gamma(G)$. Vertices v and w are referred to as *top* and *bottom*, respectively. It is easy to verify that drawing $\Gamma(G)$ can be converted to a 2-queue layout of G as follows:

– for any two distinct vertices u and v of G, $u \prec v$ if and only if the y-coordinate of u is strictly greater than the one of v, or the y-coordinate of u is equal to the one of v and u is to the left of v in $\Gamma(G)$,
– edge (u, v) is assigned to the first (to the second) queue if and only if the absolute value of the difference of the y-coordinates of u and v is one (two, respectively) in $\Gamma(G)$.

In the following, we present interesting properties of the 2-queue layout produced from drawing $\Gamma(G)$. Let $\langle u, v, w\rangle$ and $\langle u', v', w'\rangle$ be two distinct faces of $\Gamma(G)$, such that u and u' are their anchors, v and v' are their top vertices, and w and w' are their bottom vertices.

Property 1 *If $u \neq u'$ and $u \prec u'$ in the 2-queue layout, then $v \prec v'$ (if $v \neq v'$) and $w \prec w'$ (if $w \neq w'$).*

Proof The property clearly holds if vertices u and u' do not have the same y-coordinate. Otherwise, the property holds since $\Gamma(G)$ is planar. □

Property 2 *If $u = u'$, $v \neq v'$ and $v \prec v'$ in the 2-queue layout, then $w \prec w'$* (if $w \neq w'$).

Proof The property follows from the fact that $\Gamma(G)$ is planar. □

Property 3 *If $u = u'$, $w \neq w'$ and $w \prec w'$ in the 2-queue layout, then $v \prec v'$ (if $v \neq v'$*).

Proof The property follows from the fact that $\Gamma(G)$ is planar. □

3 The Upper Bound

In this section, we prove that the queue number of every planar 3-tree is at most five. Our approach is inspired by the algorithm of Wiechert [30] which results in 7-queue layouts for general (that is, not necessarily planar) 3-trees. To reduce the number of required queues in the produced layouts, we make use of structural properties of the input graph. In particular, we put the main ideas of the algorithm of Wiechert [30] into a *peeling-into-levels* approach (see, e.g., [32]), according to which the vertices and the edges of the input graph are partitioned as follows:

– Vertices incident to the outerface are at *level zero*,
– Vertices incident to the outerface of the graph induced by deleting all vertices of levels $0, \ldots, i-1$ are at *level i*,
– Edges between same-level vertices are called *level edges*, and
– Edges between vertices of different levels are called *binding edges*.

To keep the description simple, we first show how to compute a 5-queue layout of a planar 3-tree G, assuming that G has only two levels (refer to Sect. 3.1). Then, we extend our approach to the general case of more than two levels (refer to Sect. 3.2). We conclude by discussing the differences between the approach of Wiechert [30] and ours (refer to Sect. 3.3); we also describe which properties of planar 3-trees we exploited to reduce the required number of queues.

3.1 The Two-Level Case

We start with the (intuitively easier) case in which the given planar 3-tree G consists of two levels, which we denote by L_0 and L_1. Since we use this case as a tool to cope with the general case of more than two levels, we consider a slightly more general scenario. In particular, we make the following assumptions; see Fig. 3a for an illustration:

A.1 the graph G_0 induced by the vertices of level L_0 is outerplane and internally triangulated, and
A.2 each connected component of the graph G_1 induced by the vertices of level L_1 is outerplane and resides within a (triangular) face of graph G_0.

Without loss of generality we may also assume that graph G_0 is biconnected, as otherwise we can augment it to being biconnected by adding (level-L_0) edges without affecting its outerplanarity (see, e.g., the dotted-gray edges of Fig. 3a). Note that in a planar 3-tree, the graph induced by the vertices at level zero is simply a 3-cycle (and not any outerplane graph, as we have assumed), and as a result the graph induced by the vertices at level one is a single outerplane component. Our algorithm maintains the following invariants:

I.1 the linear order is such that all vertices of level L_0 precede all vertices of level L_1,
I.2 the level edges of G are assigned to two queues, which we denote by $\mathcal{Q}_0$ and $\mathcal{Q}_1$, and
I.3 the binding edges between L_0 and L_1 of G are assigned to three queues, which we denote by $\mathcal{Q}_2$, $\mathcal{Q}_3$, and $\mathcal{Q}_4$.

To compute an order that satisfies Invariant I.1, we construct two orders, one for the vertices of level L_0 (that satisfies Invariant I.2) and one for the vertices of level L_1 (that also satisfies Invariant I.2). Then, we concatenate them, so that all the vertices of level L_0 precede all the vertices of level L_1 (as required by Invariant I.1). Note that critical in this step is to also maintain Invariant I.3.

To compute an order of the vertices of level L_0 satisfying Invariant I.2, we directly apply Lemma 2, as by our initial assumption A.1, graph G_0 is internally-triangulated and outerplane. It follows that Invariant I.2 is satisfied for the vertices of level L_0. To compute an order of the vertices of level L_1 satisfying Invariant I.2, we apply Lemma 2 individually to each connected component of graph G_1, which by

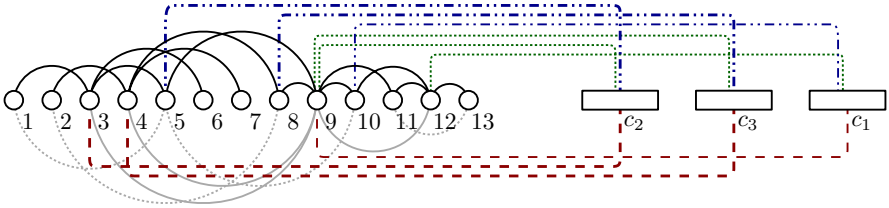


Fig. 4 The 5-queue layout for the graph in Fig. 3; since $5 \prec 8$ and $8 \prec 10$ in the order of the vertices of level L_0 as seen in Fig. 3, c_2 precedes c_3, and c_3 precedes c_1

our initial assumption A.2 is an internally-triangulated and outerplane graph. Then, the resulting orders are concatenated, such that for every two connected components of graph G_1, all vertices of the first one either precede or follow all vertices of the second one (critical at this point is to maintain Invariant I.3, as already mentioned above). The latter property allows us to use queues $\mathcal{Q}_0$ and $\mathcal{Q}_1$ for all the level edges of level L_1. Therefore, Invariant I.2 is satisfied.

To guarantee that Invariant I.3 is satisfied, we proceed as follows. Consider a connected component c of G_1. By our initial assumption A.2, component c resides within a triangular face $\langle u, v, w\rangle$ of graph G_0. Without loss of generality, let u, v and w be the anchor, top and bottom vertices of the face, respectively. We assign the binding edges incident to u to queue $\mathcal{Q}_2$, the binding edges incident to v to queue $\mathcal{Q}_3$ and the binding edges incident to w to queue $\mathcal{Q}_4$; see the blue, red, and green edges in Fig. 4.

Next, we describe how to compute the relative order of the connected components of graph G_1. Let c and c' be two such components. By our initial assumption A.2, components c and c' reside within two triangular faces, say $\langle u, v, w\rangle$ and $\langle u', v', w'\rangle$, of graph G_0. Without loss of generality, assume that u and u' are the anchors, v and v' are top, and w and w' are bottom vertices of the two faces. If $u \neq u'$, then component c precedes component c' if and only if $u \prec u'$ in the order of the vertices of level L_0. If $u = u'$, we have $v \neq v'$ or $w \neq w'$. If $v \neq v'$, then component c precedes component c' if and only if $v \prec v'$ in the order of the vertices of level L_0. Otherwise (that is, $u = u'$ and $v = v'$), component c precedes component c' if and only if $w \prec w'$ in the order of the vertices of level L_0. We claim that for the resulting order of the vertices of level L_1, Invariant I.3 is satisfied, that is, no two independent edges of each of the queues $\mathcal{Q}_2$, $\mathcal{Q}_3$ and $\mathcal{Q}_4$ are nested.

We start our proof with queue $\mathcal{Q}_2$. Consider two independent edges $(x, y) \in \mathcal{Q}_2$ and $(x', y') \in \mathcal{Q}_2$, where $x, x' \in L_0$ and $y, y' \in L_1$ (see the blue edges incident to vertices 5 and 8 in Fig. 4). By construction of queue $\mathcal{Q}_2$, vertices x and x' are anchors of two distinct faces f_x and $f_{x'}$ of graph G_0 (see the faces of Fig. 3b that contain components c_2 and c_3). Without loss of generality, we assume that $x \prec x'$ in the order of the vertices of level L_0. It follows that the two components, say c_y and $c_{y'}$, of graph G_1, that reside within faces f_x and $f_{x'}$ and contain y and y', respectively, are such that all vertices of component c_y precede all vertices of component $c_{y'}$ (in Fig. 4, $x = 5$ precedes $y = 8$; thus, $c_y = c_2$ precedes $c_{y'} = c_3$). Since $y \in c_y$ and $y' \in c_{y'}$, edges (x, y) and (x', y') are not nested, as desired.

We continue our proof with queue $\mathcal{Q}_3$. Let (x, y) and (x', y') be two independent edges of $\mathcal{Q}_3$, where $x, x' \in L_0$ and $y, y' \in L_1$ (see the red edges in Fig. 4 incident to vertices 3 and 4). By construction of $\mathcal{Q}_3$, vertices x and x' are the top vertices of two distinct faces f_x and $f_{x'}$ of graph G_0 (see the faces of Fig. 3b that contain components c_2 and c_3). Let c_y and $c_{y'}$ be the components of graph G_1 that reside within faces f_x and $f_{x'}$ and contain vertices y and y'. Finally, let u and u' be the anchors of faces f_x and $f_{x'}$, respectively. Suppose first that $u \neq u'$ and assume that $u \prec u'$ in the order the vertices of level L_0. Since $u \prec u'$, by Property 1 it follows that $x \prec x'$ and that all vertices of component c_y precede all vertices of component $c_{y'}$ (in Fig. 4, $u = 5$ precedes $u' = 8$, which implies that $x = 3$ precedes $x' = 4$; thus, $c_y = c_2$ precedes $c'_y = c_3$). Since $y \in c_y$ and $y' \in c_{y'}$, it follows that edges (x, y) and (x', y') are not nested. Suppose now that $u = u'$ and assume without loss of generality that $x \prec x'$ in the order of the vertices of level L_0. Since $u = u'$ and $x \prec x'$, all vertices of component c_y precede all vertices of component $c_{y'}$. Since $y \in c_y$ and $y' \in c_{y'}$, it follows that edges (x, y) and (x', y') are not nested, as desired.

We conclude our proof with queue $\mathcal{Q}_4$. Let as above (x, y) and (x', y') be two independent edges of $\mathcal{Q}_4$, where $x, x' \in L_0$ and $y, y' \in L_1$. It follows that vertices x and x' are the bottom vertices of two distinct faces f_x and $f_{x'}$ of graph G_0. Let c_y and $c_{y'}$ be the components of graph G_1 that reside within faces f_x and $f_{x'}$ and contain vertices y and y'. Finally, let u and u' be the anchors, and v and v' be the top vertices of faces f_x and $f_{x'}$, respectively. If $u \neq u'$, or $u = u'$ and $v \neq v'$, then the proof that edges (x, y) and (x', y') are not nested follows using similar arguments as in corresponding cases of queue $\mathcal{Q}_3$. Thus, it remains to consider the case in which $u = u'$ and $v = v'$. Assume without loss of generality that $x \prec x'$ in the order of the vertices of level L_0. It follows that all vertices of component c_y precede all vertices of component $c_{y'}$. Since $y \in c_y$ and $y' \in c_{y'}$, it follows that (x, y) and (x', y') are not nested. Hence, Invariant I.3 is satisfied, as desired.

The discussion above concludes the two-level case. However, before we proceed with the multi-level case, we make a useful observation. To satisfy Invariant I.3, we did not impose any restriction on the order of the vertices of each connected component of graph G_1 (any order that satisfies Invariant I.2 for level L_1 would be suitable for us, that is, not necessarily the one constructed by Lemma 2). What we fixed, was the relative order of these components. With this observation in mind, we are now ready to proceed to the multi-level case.

3.2 The Multi-Level Case

We now consider the general case, in which our planar 3-tree G consists of more than two levels, say $L_0, L_1, \ldots, L_\lambda$ with $\lambda \geq 2$. For $i = 0, 1, \ldots, \lambda$, let G_i be the subgraph of G induced by the vertices of level L_i. We claim that the connected components of each graph G_i are internally-triangulated outerplane graphs that are not necessarily biconnected. Clearly, this holds for graph G_0, which is a 3-cycle. Assuming that, for some $i = 1, \ldots, \lambda$, graph G_{i-1} has the claimed property, we observe that each connected component of graph G_i resides within a facial 3-cycle of graph G_{i-1}. Since each non-empty facial 3-cycle of graph G_{i-1} induces a planar 3-tree in graph

G [21], the claim follows by observing that the removal of the outer face of a planar 3-tree yields a plane graph, whose outer vertices induce an internally-triangulated outerplane graph.

For the recursive step of our algorithm, we will assume that for some $i \in \{0, 1, \ldots, \lambda - 1\}$ we have a 5-queue layout for each of the connected components of the graph H_{i+1} induced by the vertices of levels $L_{i+1}, \ldots, L_\lambda$, that satisfies the following invariants[1]:

M.1 the linear order is such that all vertices of level L_j precede all vertices of level L_{j+1} for every $j = i + 1, \ldots, \lambda - 1$;
M.2 the level edges of each of the levels $L_{i+1}, \ldots, L_\lambda$ have been assigned to two queues, which we denote by $\mathcal{Q}_0$ and $\mathcal{Q}_1$;
M.3 for every $j = i + 1, \ldots, \lambda - 1$, the binding edges between L_j and L_{j+1} have been assigned to three queues, which we denote by $\mathcal{Q}_2$, $\mathcal{Q}_3$, and $\mathcal{Q}_4$.

In the following, we show how to construct a 5-queue layout (that satisfies Invariants M.1–M.3) for each of the connected components of the graph H_i induced by the vertices of levels $L_i, \ldots, L_\lambda$. Let C_i be such a component. By definition, C_i is delimited by a connected component c_i of graph G_i, which is internally-triangulated and outerplane. If none of the faces of component c_i contains a connected component of H_{i+1}, then we compute a 2-queue layout of it using Lemma 2. Consider now the more general case, in which some of the faces of component c_i contain connected components of H_{i+1}. By Invariants M.1–M.3, we may assume that we have computed 5-queue layouts for all the connected components, say $d_1, \ldots, d_k$, of H_{i+1} that reside within the faces of component c_i.

We proceed by applying the two-level algorithm to the subgraph of C_i induced by the vertices of c_i and the vertices incident to the outer faces of $d_1, \ldots, d_k$. By the last observation that we made in the two-level case, this will result in:

i. A linear order $\mathcal{O}(c_i)$ of the vertices of component c_i,
ii. A relative order of the components $d_1, \ldots, d_k$,
iii. all vertices of c_i precede those of $d_1, \ldots, d_k$,
iv. An assignment of the (level-L_i) edges of c_i into $\mathcal{Q}_0$ and $\mathcal{Q}_1$, and
v. An assignment of the binding edges between c_i and each of $d_1, \ldots, d_k$ into $\mathcal{Q}_2$, $\mathcal{Q}_3$ and $\mathcal{Q}_4$.

Up to renaming, we may assume without loss of generality that $d_1, \ldots, d_k$ is the computed order of these components; see Fig. 5a. By (iv) and (v), all edges of C_i are assigned to $\mathcal{Q}_0, \ldots, \mathcal{Q}_4$, since the edges of components $d_1, \ldots, d_k$ have been recursively assigned to these queues. Next, we describe the order of the vertices of C_i. First, we partition the order of the vertices of C_i into $\lambda - i + 1$ disjoint intervals, say $I_i, \ldots, I_\lambda$, such that I_μ precedes I_ν if and only if $\mu < \nu$. All the (level-L_i) vertices of c_i

[1] Observe that Invariants M.1–M.3 are modifications of Invariants I.1–I.3.

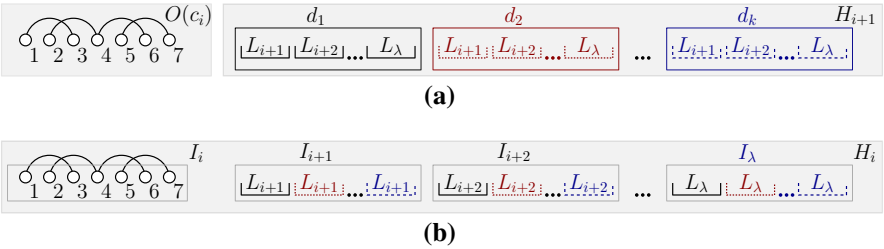


Fig. 5 Illustrations for the proof of Theorem 1. **a** For each of the components $d_1, \ldots, d_k$, all vertices of level L_j precede all vertices of level L_{j+1}; $j = i+1, \ldots, \lambda-1$, and **b** the computed linear order based on $p_i, \ldots, p_\lambda$

are contained in I_i in the order $\mathcal{O}(c_i)$ by (i). For $j = i+1, \ldots, \lambda$, interval I_j contains the vertices of level L_j of each of the components $d_1, \ldots, d_k$, such that the vertices of level L_j of component d_μ precede the vertices of level L_j of component d_ν if and only if $\mu < \nu$; see Fig. 5b. We are not ready to state the main theorem of this section.

Theorem 1 *Every planar* 3*-tree has queue number at most* 5.

Proof Since Invariant M.1 is clearly satisfied for C_i, it remains to prove that the assignment of the edges of C_i to queues $\mathcal{Q}_0, \ldots, \mathcal{Q}_4$ is such that Invariants M.2 and M.3 are satisfied.

Since the edges of H_i are partitioned into level and binding, the endvertices of each edge are either in the same or in two consecutive intervals. In the former case, the edge is level and thus assigned either to $\mathcal{Q}_0$ or to $\mathcal{Q}_1$, while in the latter case the edge is binding and thus assigned to one of $\mathcal{Q}_2$, $\mathcal{Q}_3$ and $\mathcal{Q}_4$. Edges assigned to $\mathcal{Q}_0$ and $\mathcal{Q}_1$ cannot nest, as otherwise our two-level algorithm has computed an invalid assignment for the level edges of c_i or an invalid assignment in $\mathcal{Q}_0$ and $\mathcal{Q}_1$ has been recursively computed for some of the components $d_1 \ldots, d_k$. For the same reason, any two (binding) edges of $\mathcal{Q}_2$, $\mathcal{Q}_3$ or $\mathcal{Q}_4$ cannot be nested, if both bridge c_i with the same or with two different components of level L_{i+1}, or both belong to the same component d_j, for some $j = 1, \ldots, k$. It remains to prove that there exist no two nested edges of $\mathcal{Q}_2$, $\mathcal{Q}_3$ or $\mathcal{Q}_4$ that belong to two different components d_μ and d_ν and their endvertices are in two consecutive intervals I_j and I_{j+1}, where $1 \leq \mu, \nu \leq k$ and $j = i, \ldots, \lambda-1$. This holds because all vertices of d_μ either precede or follow all vertices of d_ν in both I_j and I_{j+1} (by the choice of the relative order). Therefore, Invariants M.1–M.3 are satisfied and the proof follows. □

3.3 Differences with Wiechert's Algorithm

We conclude this section by discussing the main differences between our algorithm and the previously best-known algorithm by Wiechert [30]. It is worth noting that the latter algorithm builds upon another algorithm by Dujmović et al. [10]. Both yield queue layouts for general (that is, not necessarily planar) k-trees, using the

breadth-first search starting from an arbitrary vertex r of graph G. For each $d > 0$ and each connected component C induced by the vertices at distance d from vertex r, create a node (called *bag*) "containing" all vertices of component C; two bags are adjacent if and only if there is an edge of graph G between them. For a k-tree, the result is a tree of bags, which we denote by T, called *tree-partition*, which has the following properties:

P.1. every node of T induces a connected $(k-1)$-tree, and
P.2. for each non-root node $x \in T$, if $y \in T$ is the parent of x, then the vertices in y having a neighbor in x form a clique of size k.

Both algorithms order the bags of T such that the vertices of the bags at distance d from r precede those at distance $d+1$. The vertices within each bag are ordered by induction using P.1. The algorithms differ in the way the edges are assigned to queues; the more efficient one by Wiechert [30] uses $2^k - 1$ queues (2^{k-1} for the inter- and $2^{k-1}+1$ for the intra-bag edges), which is worst-case optimal for 1- and 2-trees.

If graph G is a planar 3-tree and the breadth-first search is started from a dummy vertex incident to the three outervertices of graph G, then the intra- and inter-bag edges correspond to the *level* and *binding* edges of our approach, while the bags at distance d from r in T correspond to different connected components of level $d-1$.

To reduce the number of queues, we observed that in graph G every node of tree-partition T induces a connected outerplanar graph, while each clique of size three by P.2 is a triangular face of graph G. By the first observation, we reduced the number of queues for intra-bag edges; by the second, we combined orders from different bags more efficiently.

4 The Lower Bound

In this section, we will prove that the queue number of planar 3-trees is at least four. To this end, we will define recursively a planar 3-tree and we will show that it contains at least one 4-rainbow in any linear order of its vertices. By Lemma 1, this directly implies that the queue number of this graph is at least four. Recall that according to the formal definition that we gave in Sect. 1, a planar 3-tree is a maximal planar graph. To keep the presentation simple, however, in this section we will relax this constraint by defining a subgraph of a planar 3-tree.

Starting with a set of T independent edges (s_i, t_i) with $1 \leq i \leq T$ and T to be determined later, we connect their endpoints to two unique vertices, which we denote by A and B; see Fig. 6a. We refer to these edges as *(s, t)-edges*. We also assume A and B to be adjacent. As a next step, we *stellate* each 3-cycle $\langle A, s_i, t_i\rangle$ with a vertex x_i, that is, we introduce vertex x_i and connect it to A, s_i and t_i. Symmetrically, we also stellate each 3-cycle $\langle B, s_i, t_i\rangle$ with a vertex y_i. Afterwards, we perform a second round of stellations, where we stellate each of the 3-cycles $\langle x_i, s_i, t_i\rangle$, $\langle y_i, s_i, t_i\rangle$, $\langle A, x_i, s_i\rangle$, $\langle A, x_i, t_i\rangle$, $\langle B, y_i, s_i\rangle$ and $\langle B, y_i, t_i\rangle$ with vertices α_i, β_i, p_i, q_i, u_i and v_i, respectively. We

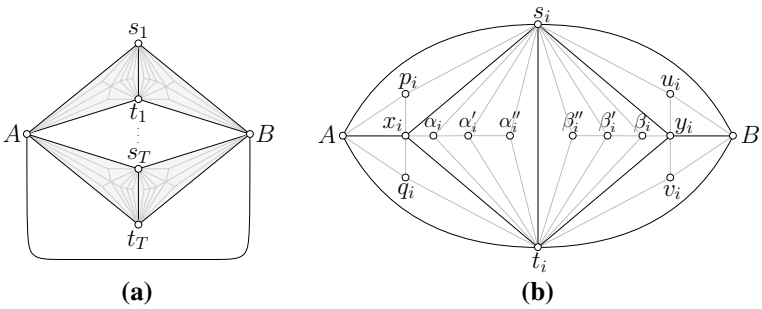


Fig. 6 Construction of graph G_T: Each gray subgraph in **a** corresponds to a copy of the graph of **b**

further stellate $\langle s_i, t_i, \alpha_i \rangle$ with α'_i and then $\langle s_i, t_i, \alpha'_i \rangle$ with α''_i. Symmetrically, we stellate $\langle s_i, t_i, \beta_i \rangle$ with β'_i, and $\langle s_i, t_i, \beta'_i \rangle$ with β''_i. For an illustration refer to Fig. 6b.

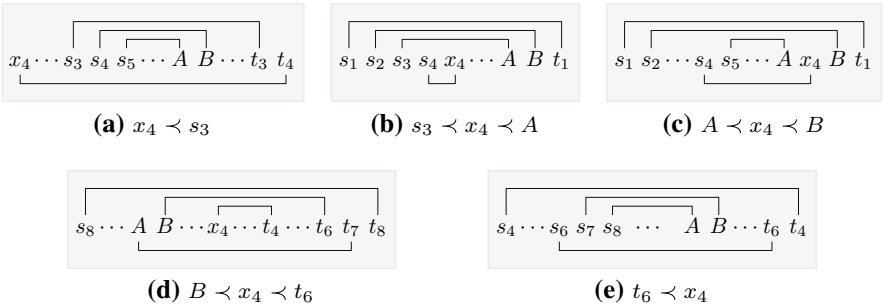
Let G_T be the graph constructed so far. We refer to vertices A and B as the *poles* of G_T and we assume that G_T admits a 3-queue layout $\mathcal{Q}$. By symmetry, we may assume without loss of generality that $A \prec B$ and that $s_i \prec t_i$ for each edge (s_i, t_i). Consider a single edge (s_i, t_i) and the relative order of its endvertices with respect to the poles A and B of graph G_T. Clearly, there exist six possible permutations:

P.1. $s_i \prec A \prec B \prec t_i$,
P.2. $A \prec s_i \prec B \prec t_i$,
P.3. $s_i \prec A \prec t_i \prec B$,
P.4. $A \prec B \prec s_i \prec t_i$,
P.5. $s_i \prec t_i \prec A \prec B$, and
P.6. $A \prec s_i \prec t_i \prec B$.

The main steps in our proof are the following. We first observe that by the pigeonhole principle and by setting $T = 6\tau$, at least one of the permutations P.1, … , P.6 applies to at least τ edges. Then, we show that if "too many" (s, t)-edges share one of the permutations P.1, … , P.5, there exists a 4-rainbow regardless of the linear order of the vertices of G_T, which by Lemma 1 contradicts the fact that $\mathcal{Q}$ is a 3-queue layout for graph G_T. Therefore, if T is large enough, then for at least one (s, t)-edge of G_T permutation P.6 applies. Based on this implication, we describe how to augment graph G_T, such that we can also rule out permutation P.6. Thereby, proving the claimed lower bound of four. We start with an auxiliary lemma.

Lemma 3 *In every queue that contains r^2 independent edges, there exists either an r-twist or an r-necklace.*

Proof Assume that no r-twist exists, as otherwise there is nothing to prove. We will prove the existence of an r-necklace. Let $(s_1, t_1), \ldots, (s_{r^2}, t_{r^2})$ be the r^2 independent edges such that $s_i \prec t_i$, for $i = 1, \ldots, r^2$. Assume without loss of generality that $s_i \prec s_{i+1}$ for each $i = 1, \ldots, r^2 - 1$. Consider the edge (s_1, t_1). Since s_1 is the first vertex in the order and no two edges nest, each vertex t_i, with $i > 1$, is to the



(a) $x_4 \prec s_3$ **(b)** $s_3 \prec x_4 \prec A$ **(c)** $A \prec x_4 \prec B$

(d) $B \prec x_4 \prec t_6$ **(e)** $t_6 \prec x_4$

Fig. 7 Illustration for the Case P.1

right of t_1. Since no r-twist exists, vertex s_r is to the right of vertex t_1. Thus, edges (s_1, t_1) and (s_r, t_r) do not cross. The removal of the edges $(s_1, t_1), \ldots, (s_{r-1}, t_{r-1})$ makes vertex s_r first. By applying this argument $r-1$ times, we conclude that $(s_1, t_1), (s_r, t_r), \ldots \left(s_{(r-1)^2+1}, t_{(r-1)^2+1}\right)$ form an r-necklace and the proof of the lemma follows. □

Applying the pigeonhole principle to a k-queue layout, we obtain the following.

Corollary 1 *Every k-queue layout with at least kr^2 independent edges contains at least one r-twist or at least one r-xnecklace.*

We exploit Corollary 1 as follows. Recall that $\mathcal{Q}$ is a 3-queue layout for graph G_T. So, if we set $T = 18r^2$ for an $r > 0$ of our choice, then at least $3r^2$ (s, t)-edges of graph G_T share the same permutation. Since these edges are by construction independent, by Corollary 1 at least r of them, say without loss of generality $(s_1, t_1), \ldots, (s_r, t_r)$, form a necklace or a twist (while also sharing the same permutation). In the following, we show that, for an appropriate choice of r, if $(s_1, t_1), \ldots, (s_r, t_r)$ form a necklace or a twist and simultaneously share one of the permutations P.1, … , P.5, then a 4-rainbow is inevitably induced, which by Lemma 1 contradicts the fact that $\mathcal{Q}$ is a 3-queue layout for G_T. We consider each case separately.

Case P.1 Let $r = 8$. It is sufficient to consider the case in which the edges $(s_1, t_1), \ldots, (s_8, t_8)$ form a twist, since for $r > 1$ the necklace case is impossible. Since $(s_1, t_1), \ldots, (s_8, t_8)$ share permutation P.1, the order is $[s_1 \ldots s_8 A B t_1 \ldots t_8]$. In the following, we prove the existence of a 4-rainbow, contradicting the fact that $\mathcal{Q}$ is a 3-queue layout for graph G_T. More precisely:

- if $x_4 \prec s_3$, then the edges (x_4, t_4), (s_3, t_3), (s_4, B) and (s_5, A) form a 4-rainbow; see Fig. 7a,
- if $s_3 \prec x_4 \prec A$, then the edges (s_1, t_1), (s_2, B), (s_3, A) and (s_4, x_4) form a 4-rainbow; see Fig. 7b,
- if $A \prec x_4 \prec B$, then the edges (s_1, t_1), (s_2, B), (s_4, x_4) and (s_5, A) form a 4-rainbow; see Fig. 7c,

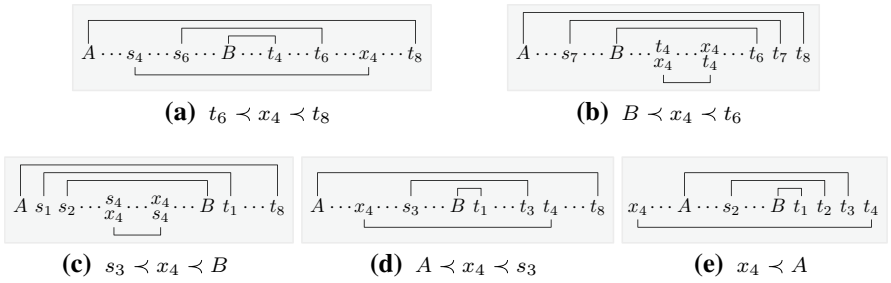


Fig. 8 Illustration for the Case P.2 when $x_4 \prec t_8$ holds

– if $B \prec x_4 \prec t_6$, then the edges (s_8, t_8), (A, t_7), (B, t_6) and (x_4, t_4) form a 4-rainbow; see Fig. 7d, and
– if $t_6 \prec x_4$, then the edges (s_4, t_4), (s_6, t_6), (s_7, B) and (s_8, A) form a 4-rainbow; see Fig. 7e.

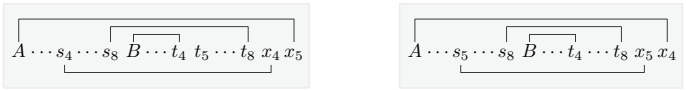
Since each subcase yields a 4-rainbow, the proof of Case P.1 follows. □

Case P.2 As in the previous case, we set $r = 8$ and we only consider the case, in which $(s_1, t_1), \ldots, (s_8, t_8)$ form a twist, since the necklace case is again impossible. Hence, the order is $[As_1 \ldots s_8 B t_1 \ldots t_8]$. First, we claim that $t_8 \prec x_4$. To prove the claim, assume to the contrary that $x_4 \prec t_8$. In the following, we prove that this assumption inevitably implies a 4-rainbow, contradicting the fact that $\mathcal{Q}$ is a 3-queue layout for graph G_T. More precisely:

– if $t_6 \prec x_4 \prec t_8$, then the edges (A, t_8), (s_4, x_4), (s_6, t_6) and (B, t_4) form a 4-rainbow; see Fig. 8a,
– if $B \prec x_4 \prec t_6$, then the edges (A, t_8), (s_7, t_7), (B, t_6) and (x_4, t_4) form a 4-rainbow; see Fig. 8b,
– if $s_3 \prec x_4 \prec B$, then the edges (A, t_8), (s_1, t_1), (s_2, B) and (s_4, t_4) form a 4-rainbow; see Fig. 8c,
– if $A \prec x_4 \prec s_3$, then the edges (A, t_8), (x_4, t_4), (s_3, t_3) and (B, t_1) form a 4-rainbow; see Fig. 8d, and
– if $x_4 \prec A$, then the edges (x_4, t_4), (A, t_3), (s_2, t_2) and (B, t_1) form a 4-rainbow; see Fig. 8e.

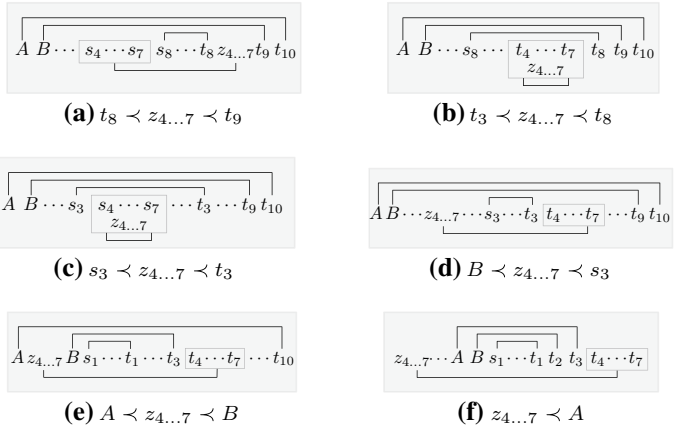
An analogous case distinction yields that $t_8 \prec x_5$ must also hold. However, we have no knowledge about the relative order of x_4 and x_5. In the following, we show that regardless of the relative order of x_4 and x_5, a 4-rainbow is inevitably formed. More precisely:

– if $x_4 \prec x_5$, then the edges (A, x_5), (s_4, x_4), (s_8, t_8) and (B, t_4) form a 4-rainbow; see Fig. 9a, and



(a) $t_8 \prec x_4 \prec x_5$ **(b)** $t_8 \prec x_5 \prec x_4$

Fig. 9 Illustration for the Case P.2 when $t_8 \prec x_4$ and $t_8 \prec x_5$ hold



(a) $t_8 \prec z_{4\ldots7} \prec t_9$ **(b)** $t_3 \prec z_{4\ldots7} \prec t_8$

(c) $s_3 \prec z_{4\ldots7} \prec t_3$ **(d)** $B \prec z_{4\ldots7} \prec s_3$

(e) $A \prec z_{4\ldots7} \prec B$ **(f)** $z_{4\ldots7} \prec A$

Fig. 10 Illustration for the Case P.4 when $z_{4\ldots7} \prec t_9$ holds

– if $x_5 \prec x_4$, then the edges (A, x_4), (s_5, t_5), (s_8, t_8) and (B, t_4) form a 4-rainbow; see Fig. 9b.

Since each subcase yields a 4-rainbow, the proof of this case follows. □

Case P.3 This case can be ruled out as Case P.2 due to symmetry. □

Case P.4 We set $r = 10$ and we proceed by distinguishing two subcases based on whether the edges $(s_1, t_1), \ldots, (s_{10}, t_{10})$ form a twist or a necklace. Note that in contrast to the previous cases, here both subcases are possible.

We start with the twist case. Hence, the order is $[ABs_1 \ldots s_{10}t_1 \ldots t_{10}]$. Let $Z_{4\ldots7} = \{x_4, \ldots, x_7\} \cup \{y_4, \ldots y_7\}$ and let $z_{4\ldots7}$ be any element of $Z_{4\ldots7}$. By construction of graph G_T, vertex $z_{4\ldots7}$ has a neighbor in $\{s_4, \ldots, s_7\}$, and a neighbor in $\{t_4, \ldots, t_7\}$. Denote by $s_{4\ldots7}$ the endvertex of the former neighbor. Symmetrically, denote by $t_{4\ldots7}$ the latter neighbor. First, we claim that $t_9 \prec z_{4\ldots7}$, that is, all $x_4, \ldots, x_7$ and $y_4, \ldots, y_7$ are preceded by t_9. To prove the claim, assume to the contrary that $z_{4\ldots7} \prec t_9$. In the following, we prove that this assumption inevitably implies a 4-rainbow, contradicting the fact that $\mathcal{Q}$ is a 3-queue layout for graph G_T. More precisely:

– if $t_8 \prec z_{4\ldots7} \prec t_9$, then the edges (A, t_{10}), (B, t_9), $(s_{4\ldots7}, z_{4\ldots7})$ and (s_8, t_8) form a 4-rainbow; see Fig. 10a,

- if $t_3 \prec z_{4\ldots7} \prec t_8$, then the edges (A, t_{10}), (B, t_9), (s_8, t_8) and $(z_{4\ldots7}, t_{4\ldots7})$ form a 4-rainbow; see Fig. 10b,
- if $s_3 \prec z_{4\ldots7} \prec t_3$, then the edges (A, t_{10}), (B, t_9), (s_3, t_3) and $(s_{4\ldots7}, z_{4\ldots7})$ form a 4-rainbow; see Fig. 10c,
- if $B \prec z_{4\ldots7} \prec s_3$, then the edges (A, t_{10}), (B, t_9), $(z_{4\ldots7}, t_{4\ldots7})$ and (s_3, t_3) form a 4-rainbow; see Fig. 10d,
- if $A \prec z_{4\ldots7} \prec B$, then the edges (A, t_{10}), $(z_{4\ldots7}, t_{4\ldots7})$, (B, t_3) and (s_1, t_1) form a 4-rainbow; see Fig. 10e, and
- if $z_{4\ldots7} \prec A$, then the edges $(z_{4\ldots7}, t_{4\ldots7})$, (A, t_3), (B, t_2) and (s_1, t_1) form a 4-rainbow; see Fig. 10f.

From the above case analysis, it follows that $t_9 \prec z_{4\ldots7}$, that is, all $x_4, \ldots, x_7$ and $y_4, \ldots, y_7$ are preceded by t_9. Let us now take a closer look at the ordering of the eight vertices in $Z_{4\ldots7}$. We claim that their ordering has to comply with the following two requirements:

R.1. the indices of the first 7 elements of $Z_{4\ldots7}$ are non-decreasing, and
R.2. for the last 7 elements of $Z_{4\ldots7}$, all x_i's precede all y_j's.

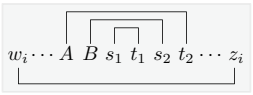
Assume to the contrary, that R.1 does not hold. Then, there exists a pair of vertices, say without loss of generality x_i and x_j with $i < j$, such that $x_j \prec x_i$, and x_i is not the last element of $Z_{4\ldots7}$. Since the last element of $Z_{4\ldots7}$ has a connection to either A or B, this connection and the edges (s_i, x_i), (s_j, x_j) and (s_9, t_9) form a 4-rainbow, contradicting the fact that $\mathcal{Q}$ is a 3-queue layout for graph G_T. Hence, R.1 holds, as desired.

To complete the proof of our claim, now assume to the contrary that R.2 does not hold. Then, there exists a pair of vertices, say without loss of generality x_i and y_j, such that $y_j \prec x_i$, and y_j is not the first element of $Z_{4\ldots7}$. Without loss of generality, let x_k be the first element of $Z_{4\ldots7}$. Then, the edges (A, x_i), (B, y_j), (s_k, x_k) and (s_9, t_9) form a 4-rainbow, contradicting the fact that $\mathcal{Q}$ is a 3-queue layout for graph G_T. Hence, R.2 holds, as desired.

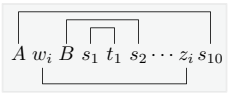
Now, we show that R.1 and R.2 cannot simultaneously hold, which by our claim implies the existence of a 4-rainbow. Consider the last element of $Z_{4\ldots7}$. Assume that R.1 and R.2 both hold. By R.2, we may deduce that the last three elements of $Z_{4\ldots7}$ belong to $\{y_4, \ldots y_7\}$. Let them be y_i, y_j, y_ℓ as they appear from left to right. Then, by R.1 we have that $i < j$. Consider now x_j. By R.1, $y_i \prec x_j$ must hold. This contradicts the fact that y_i, y_j, y_ℓ are the last three elements of $Z_{4\ldots7}$.

We continue with the necklace case. Here, the order is $[ABs_1t_1 \ldots s_{10}t_{10}]$. In the following, we make several observations in the form of propositions.

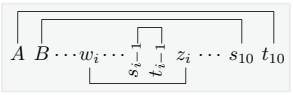
Proposition 1 *Let w_i be a neighbor of s_i or t_i not in $\{A, B, s_i, t_i\}$, for $3 \leq i \leq 8$. Then, either $s_{i-1} \prec w_i \prec t_{i+1}$, or $s_{10} \prec w_i$ holds in $\mathcal{Q}$.*



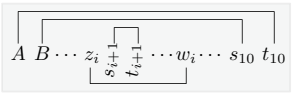
(a) $w \prec A$



(b) $A \prec w \prec B$



(c) $B \prec w \prec s_{i-1}$



(d) $t_{i+1} \prec w \prec s_{10}$

Fig. 11 Illustrations for the proof of Proposition 1

Proof Let $z_i \in \{s_i, t_i\}$ be the neighbor of vertex w_i. We prove in the following that for any placement of w_i such that neither $s_{i-1} \prec w_i \prec t_{i+1}$ nor $s_{10} \prec w_i$ hold, there is a 4-rainbow:

- if $w_i \prec A$, then the edges (w_i, z_i), (A, t_2), (B, s_2) and (s_1, t_1) form a 4-rainbow; see Fig. 11a;
- if $A \prec w_i \prec B$, then the edges (A, s_{10}), (w_i, z_i), (B, s_2) and (s_1, t_1) form a 4-rainbow; see Fig. 11b;
- if $B \prec w_i \prec s_{i-1}$, then (A, t_{10}), (B, s_{10}), (w_i, z_i) and (s_{i-1}, t_{i-1}) form a 4-rainbow; see Fig. 11c, and
- if $t_{i+1} \prec w_i \prec s_{10}$, then (A, t_{10}), (B, s_{10}), (z_i, w_i) and (s_{i+1}, t_{i+1}) form a 4-rainbow; see Fig. 11d.

Since each case yields a 4-rainbow, the proof follows. □

Proposition 2 *Let w_i and z_i be two vertices not in $\{A, B\}$ that form a K_4 with s_i and t_i for $3 \le i \le 8$. Then, at least one of the following holds in $\mathcal{Q}$: $s_{10} \prec w_i$ or $s_{10} \prec z_i$.*

Proof The edges of the K_4 formed by the vertices s_i, t_i, w_i and z_i induce a 2-rainbow regardless of the relative order of their endvertices. By Proposition 1, each of w_i and z_i is either between s_{i-1} and t_{i+1}, or after s_{10}. But if both w_i and z_i were between s_{i-1} and t_{i+1}, then the 2-rainbow composed of the edges of the K_4, along with the two edges (A, t_{10}), (B, s_{10}), would form a 4-rainbow; a contradiction. Hence, $s_{10} \prec w_i$ or $s_{10} \prec z_i$ must hold, as desired. □

Proposition 3 *For $4 \le i \le 8$, each vertex from the set $\{x_i, y_i, p_i, q_i, u_i, v_i\}$ is between s_{i-1} and t_{i+1} in $\mathcal{Q}$.*

Proof Let w_i be an arbitrary vertex from the set $\{x_i, y_i, p_i, q_i, u_i, v_i\}$. Observe that, by construction of graph G_T, vertex w_i has an edge (which we call *long*) to exactly one of A and B, and an edge (which we call *short*) to at least one of s_i and t_i. By Proposition 1, it is sufficient to prove that w_i is not after s_{10}. Assume for a contradiction that $s_{10} \prec w_i$.

(a) $x_i \prec s_i \prec t_i \prec y_i$ **(b)** $s_i \prec x_i \prec y_i \prec t_i$

Fig. 12 Contradiction for placing $x_i, y_i, p_i, q_i, u_i, v_i$ in range (s_{i-1}, t_{i+1}), $4 \le i \le 8$

Consider the edges (x_{i-1}, α_{i-1}), $(\alpha'_{i-1}, \alpha''_{i-1})$, (y_{i-1}, β_{i-1}), $(\beta'_{i-1}, \beta''_{i-1})$, and observe that the endpoints of each of these edges create a K_4 with s_{i-1} and t_{i-1}. By Proposition 2, one endvertex from each of these four edges is after s_{10}. By the pigeonhole principle, at least two of these endvertices either precede or follow w_i. Call them a_{i-1} and b_{i-1}, where without loss of generality $a_{i-1} \prec b_{i-1}$. Then, the edges (s_{i-1}, b_{i-1}), (t_{i-1}, a_{i-1}) and (s_9, t_9) form a 3-rainbow, since $s_{i-1} \prec t_{i-1} \prec s_9 \prec t_9 \prec s_{10} \prec a_{i-1} \prec b_{i-1}$. This 3-rainbow together with the short edge of w_i (when both a_{i-1} and b_{i-1} follow w_i, that is, $w_i \prec a_{i-1} \prec b_{i-1}$), or with the long edge of w_i (when both a_{i-1} and b_{i-1} precede w_i, that is, $a_{i-1} \prec b_{i-1} \prec w_i$) yield a 4-rainbow; a contradiction. □

Since s_i and t_i are between s_{i-1} and t_{i+1}, it follows from Proposition 3, that each vertex from the set $\{s_i, t_i, x_i, y_i, p_i, q_i, u_i, v_i\}$ is between s_{i-1} and t_{i+1}, for $4 \le i \le 8$. Then, the edges between these vertices cannot form a 2-rainbow, as otherwise this 2-rainbow along with the two edges (A, t_{10}) and (B, s_{10}) would form a 4-rainbow, contradicting the fact that $\mathcal{Q}$ is a 3-queue layout for G_T. Assume without loss of generality that $x_i \prec y_i$. Since (x_i, y_i) is the only missing edge between vertices $\{x_i, y_i, s_i, t_i\}$, it follows that in order to avoid a 2-rainbow, we may assume that one of the following two cases applies (for an illustration, refer to Fig. 12):

- $x_i \prec s_i \prec t_i \prec y_i$,
- $s_i \prec x_i \prec y_i \prec t_i$.

In both cases, vertex p_i must precede both x_i and s_i, as otherwise either (p_i, s_i), (x_i, t_i), or (p_i, x_i), (s_i, t_i) would form a 2-rainbow; see Fig. 12. But then there is no valid position for q_i without creating a 2-rainbow in either case, resulting together with (A, t_{10}) and (B, s_{10}) in a 4-rainbow. This completes the description of Case P.4. □

Case P.5 This case can be ruled out as Case P.4 due to symmetry. □

From the above case analysis it follows that if r is at least 10 (which implies that T is at least 1,800), then for at least one (s, t)-edge of G_T permutation P.6 applies, that is, there exists $1 \le i_0 \le T$ such that $A \prec s_{i_0} \prec t_{i_0} \prec B$. Notice that the edges (A, B) and (s_{i_0}, t_{i_0}) form a 2-rainbow.

We proceed by augmenting graph G_T as follows. For each edge (s_i, t_i) of graph G_T, we introduce a new copy of graph G_T, which has s_i and t_i as poles. Let G'_T be the augmented graph and let $(s'_1, t'_1), \ldots, (s'_T, t'_T)$ be the (s, t)-edges of the copy of graph G_T in G'_T corresponding to the edge (s_{i_0}, t_{i_0}) of the original graph G_T. Then, by our arguments above, in any 3-queue layout of G'_T, there exists $1 \le i_1 \le T$, such that

$s_{i_0} \prec s'_{i_1} \prec t'_{i_1} \prec t_{i_0}$. Hence, the edges (A, B), (s_{i_0}, t_{i_0}) and (s'_{i_1}, t'_{i_1}) form a 3-rainbow, since $A \prec s_{i_0} \prec t_{i_0} \prec B$ holds. If we apply the same augmentation procedure to graph G'_T, then we guarantee that the resulting graph G''_T, which is clearly a subgraph of a planar 3-tree, has inevitably a 4-rainbow. Hence, Theorem 2 follows. We stress here that graph G''_T is rather large. Since $T \geq 1,800$, the total number of (s, t)-edges in G''_T is at least $1,800^3$.

Theorem 2 *There exist planar* 3*-trees that have queue number at least* 4.

5 Track Layouts

In this section, we discuss implications of our results to the closely-related track layouts [11], which are formally defined as follows. Let $\{V_i : 1 \leq i \leq t\}$ be a partition of the vertex set of a graph G such that for every edge (u, v) of G, if $u \in V_i$ and $v \in V_j$, then $i \neq j$. Suppose that $<_i$ is a total order of V_i. Then, the ordered set $(V_i, <_i)$ is called a *track* and the partition is called a *t-track assignment* of G. An *X-crossing* in a track assignment consists of two edges (u, v) and (u', v') such that u and u' are on the same track V_i, v and v' are on a different track V_j with $u <_i u'$ and $v' <_j v$. A *track layout* is a track assignment with no X-crossings. In other words, a track layout of a graph is a partition of its vertices into a number of tracks, such that the vertices in each track form an independent set and the edges between each pair of tracks form a non-crossing set. The *track number* of graph G is the minimum t, such that G has a t-track layout.

Track and queue layouts are closely related to each other, as observed by Dujmović et al. [11]. In particular, every graph that admits a t-track layout has queue number at most $(t - 1)$, while every graph that admits a q-queue layout has track number at most $4q \cdot 4^{q(2q-1)(4q-1)}$. For the case of graphs with bounded tree-width, several upper bounds on the track number are known; trees have track number 3 [14], outerplanar graphs have track number 5 [11], series-parallel graphs have track number at most 15 [6], and planar 3-trees have track number at most 5415 [6].

In the following, we improve the upper bound on the track number of planar 3-trees, utilizing the following relation between the acyclic chromatic number of a graph admitting a q-queue layout and its track number. Recall that a vertex coloring is *acyclic*, if there is no bichromatic cycle, that is, all cycles consists of at least three colors.

Lemma 4 (Dujmović et al. [10]) *Every q-queue graph with acyclic chromatic number c has track-number at most* $c(2q)^{c-1}$.

Since the unique 4-coloring of a planar 3-tree is acyclic [15], combining Lemma 4 with Theorem 1 we obtain the following result.

Theorem 3 *The track number of a planar* 3*-tree is at most* 4000.

Notice that the upper bound of Theorem 3 has been recently improved [26].

6 Conclusions

In this work, we presented improved bounds on the queue number of planar 3-trees. We conclude by mentioning two interesting open problems that arise from our work:

- The first one concerns the exact upper bound on the queue number of planar 3-trees. Does there exist a planar 3-tree whose queue number is five (as our upper bound) or the queue number of every planar 3-tree is four (as our lower bound example)?
- The second problem is whether the technique that we developed for planar 3-trees can be extended so to improve the upper bound for the queue number of general (that is, non-planar) k-trees, which is currently exponential in k [30].

Acknowledgements Open Access funding provided by Projekt DEAL. The authors would like to thank the reviewers of this work for their very helpful comments and suggestions.

References

1. Alam, J.M., Bekos, M.A., Gronemann, M., Kaufmann, M., Pupyrev, S.: Queue layouts of planar 3-trees. In: T.C. Biedl, A. Kerren (eds.) Graph Drawing and Network Visualization, vol. 11282 , LNCS, pp. 213–226. Springer, Berlin (2018). https://doi.org/10.1007/978-3-030-04414-5_15
2. Bekos, M.A., Förster, H., Gronemann, M., Mchedlidze, T., Montecchiani, F., Raftopoulou, C.N., Ueckerdt, T.: Planar graphs of bounded degree have bounded queue number. SIAM J. Comput. **48**(5), 1487–1502 (2019). https://doi.org/10.1137/19M125340X
3. Bhatt, S.N., Chung, F.R.K., Leighton, F.T., Rosenberg, A.L.: Scheduling tree-dags using FIFO queues: a control-memory trade-off. J. Parallel Distrib. Comput. **33**(1), 55–68 (1996). https://doi.org/10.1006/jpdc.1996.0024
4. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall, New York (1999)
5. Di Battista, G., Frati, F., Pach, J.: On the queue number of planar graphs. SIAM J. Comput. **42**(6), 2243–2285 (2013). https://doi.org/10.1137/130908051
6. Di Giacomo, E., Liotta, G., Meijer, H.: Computing straight-line 3D grid drawings of graphs in linear volume. Comput. Geom. **32**(1), 26–58 (2005). https://doi.org/10.1016/j.comgeo.2004.11.003
7. Dujmović, V.: Graph layouts via layered separators. J. Comb. Theory Ser. B **110**, 79–89 (2015). https://doi.org/10.1016/j.jctb.2014.07.005
8. Dujmović, V., Frati, F.: Stack and queue layouts via layered separators. J. Graph Algorithms Appl. **22**(1), 89–99 (2018). https://doi.org/10.7155/jgaa.00454

9. Dujmović, V., Joret, G., Micek, P., Morin, P., Ueckerdt, T., Wood, D.R.: Planar graphs have bounded queue-number. In: Zuckerman, D. (ed.) FOCS, pp. 862–875. IEEE Computer Society (2019). https://doi.org/10.1109/FOCS.2019.00056
10. Dujmović, V., Morin, P., Wood, D.R.: Layout of graphs with bounded tree-width. SIAM J. Comput. **34**(3), 553–579 (2005). https://doi.org/10.1137/S0097539702416141
11. Dujmović, V., Pór, A., Wood, D.R.: Track layouts of graphs. *Discrete Math. Theoret. Comput. Sci.*, **6**(2), 497–522 (2004). http://dmtcs.episciences.org/315
12. Dujmović, V., Wood, D.R.: Tree-partitions of k-trees with applications in graph layout. In: Bodlaender, H.L. (ed.) WG, vol. 2880 LNCS, pp. 205–217. Springer, Berlin (2003). https://doi.org/10.1007/978-3-540-39890-5_18
13. Dujmović, V., Wood, D.R.: Stacks, queues and tracks: layouts of graph subdivisions. *Discrete Math. Theoret. Comput. Sci.*, **7**(1), 155–202 (2005). http://dmtcs.episciences.org/346
14. Felsner, S., Liotta, G., Wismath, S.K.: Straight-line drawings on restricted integer grids in two and three dimensions. J. Gr. Algorithms Appl. **7**(4), 363–398 (2003). https://doi.org/10.7155/jgaa.00075
15. Fertin, G., Raspaud, A., Reed, B.A.: On star coloring of graphs. In: Brandstädt, A., Le, V.B. (eds.) WG, vol. 2204 of LNCS, pp. 140–153. Springer, Berlin (2001). https://doi.org/10.1007/3-540-45477-2_14
16. Harary, F.: Graph Theory. Addison-Wesley, Reading, MA (1972)
17. Hasunuma, T.: Laying out iterated line digraphs using queues. In: Liotta, G. (ed.) Graph Drawing, volume 2912 of LNCS, pp. 202–213. Springer, Berlin (2003). https://doi.org/10.1007/978-3-540-24595-7_19
18. Heath, L.S., Leighton, F.T., Rosenberg, A.L.: Comparing queues and stacks as mechanisms for laying out graphs. SIAM J. Discrete Math. **5**(3), 398–412 (1992). https://doi.org/10.1137/0405031
19. Heath, L.S., Rosenberg, A.L.: Laying out graphs using queues. SIAM J. Comput. **21**(5), 927–958 (1992). https://doi.org/10.1137/0221055
20. Kaufmann, M., Wagner, D. (eds).: Drawing Graphs, Methods and Models, volume 2025 of LNCS. Springer, Berlin (2001)
21. Mondal, D., Nishat, R.I., Rahman, M.S., Alam, M.J.: Minimum-area drawings of plane 3-trees. J. Gr. Algorithms Appl. **15**(2), 177–204 (2011). https://doi.org/10.7155/jgaa.00222
22. Ollmann, T.: On the book thicknesses of various graphs. In Hoffman, F., Levow, R., Thomas, R. (eds.) Southeastern Conference on Combinatorics, Graph Theory and Computing, volume VIII of Congressus Numerantium, p. 459 (1973)
23. Pach, J., Thiele, T., Tóth, G.: Three-dimensional grid drawings of graphs. In Di Battista, G. (ed.) Graph Drawing, volume 1353 of LNCS, pp. 47–51. Springer, Berlin (1997). https://doi.org/10.1007/3-540-63938-1_49
24. Pemmaraju, S.V.: Exploring the powers of stacks and queues via graph layouts. PhD thesis, Virginia Tech (1992)
25. Pupyrev, S.: Mixed linear layouts of planar graphs. In: Graph Drawing, volume 10692 of LNCS, pp. 197–209. Springer, Berlin (2017). https://doi.org/10.1007/978-3-319-73915-1_17
26. Pupyrev, S.: Improved bounds for track numbers of planar graphs. *CoRR*, 1910.14153, (2019). https://arxiv.org/abs/1910.14153
27. Rengarajan, S., Madhavan, C.E.V.: Stack and queue number of 2-trees. In: Du, D., Li, M. (eds.) COCOON, volume 959 of LNCS, pp. 203–212. Springer, Berlin (1995). https://doi.org/10.1007/BFb0030834
28. Shahrokhi, F., Shi, W.: On crossing sets, disjoint sets, and pagenumber. J. Algorithms **34**(1), 40–53 (2000). https://doi.org/10.1006/jagm.1999.1049
29. Tarjan, R.E.: Sorting using networks of queues and stacks. J. ACM **19**(2), 341–346 (1972). https://doi.org/10.1145/321694.321704
30. Wiechert, V.: On the queue-number of graphs with bounded tree-width. *Electr. J. Comb.*, **24**(1), P1.65, (2017). http://www.combinatorics.org/ojs/index.php/eljc/article/view/v24i1p65
31. Wood, D.R.: Queue layouts, tree-width, and three-dimensional graph drawing. In: Agrawal, M., Seth, A. (eds.) FSTTCS, volume 2556 of LNCS, pp. 348–359. Springer, Berlin (2002). https://doi.org/10.1007/3-540-36206-1_31
32. Yannakakis, M.: Embedding planar graphs in four pages. J. Comput. Syst. Sci. **38**(1), 36–67 (1989). https://doi.org/10.1016/0022-0000(89)90032-9