Queue Layouts of Planar 3-Trees

Jawaherul Md. Alam[1], Michael A. Bekos[2], Martin Gronemann[3(✉)], Michael Kaufmann[2], and Sergey Pupyrev[1]

[1] Department of Computer Science, University of Arizona, Tucson, USA
jawaherul@gmail.com, spupyrev@gmail.com
[2] Institut für Informatik, Universität Tübingen, Tübingen, Germany
{bekos,mk}@informatik.uni-tuebingen.de
[3] Institut für Informatik, Universität zu Köln, Köln, Germany
gronemann@informatik.uni-koeln.de

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 G is 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. As opposed to general planar graphs, whose queue number is not known to be bounded by a constant, the queue number of planar 3-trees has been shown to be at most seven. In this work, we improve the upper bound to five. We also show that there exist planar 3-trees, whose queue number is at least four; this is the first example of a planar graph with queue number greater than three.

1 Introduction

In a *queue* layout [12], the vertices of a graph are restricted to a line and the 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 queues are nested; see Fig. 1. 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 [14], 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 [2,11,15,20]. As a result, they have been studied extensively over the years [3,5,9,10,12,16–21]. An important open problem in this area is whether the queue number of *planar* graphs is bounded by a constant. A positive answer to this problem would have several important implications, e.g., (i) that every n-vertex planar graph admits a $\mathcal{O}(1) \times \mathcal{O}(1) \times \mathcal{O}(n)$ straight-line grid drawing [22], (ii) that every Hamiltonian bipartite planar graph admits a 2-layer drawing and an edge-coloring of bounded size, such that edges of the same

This work is supported by the DFG grant Ka812/17-1 and DAAD project 57419183.

T. Biedl and A. Kerren (Eds.): GD 2018, LNCS 11282, pp. 213–226, 2018.
https://doi.org/10.1007/978-3-030-04414-5_15

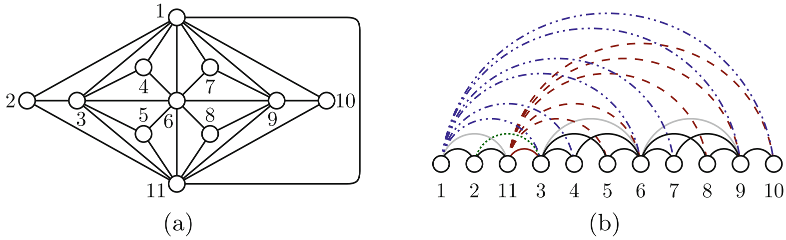


Fig. 1. (a) The Goldner-Harary planar 3-tree, and (b) a 5-queue layout of it produced by our algorithm, in which edges of different queues are colored differently. (Color figure online)

color do not cross [8], and (iii) that the queue number of k-planar graphs is also bounded by a constant [9]. The best-known upper bound is due to Dujmović [4], who showed that the queue number of an n-vertex planar graph is at most $\mathcal{O}(\log n)$ (improving upon an earlier bound by Di Battista et al. [3]).

It is worth noting that many subclasses of planar graphs have bounded queue number. Every tree has queue number one [12], outerplanar graphs have queue number at most two [11], and series-parallel graphs have queue number at most three [18]. Surprisingly, planar 3-trees have queue number at most seven [21], although they were conjectured to have super-constant queue number by Pemmaraju [16]. 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 $\mathcal{NP}$-complete [11]; for a survey refer to [9].

Our Contribution. In Sect. 2, we improve the upper bound on the queue number of planar 3-trees from seven [21] to five; 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 $n>3$. In Sect. 3, we show that there exist planar 3-trees, whose queue number is at least four, thus strengthening a corresponding result of Wiechert [21] 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. We conclude in Sect. 4 with open problems.

Preliminaries. For a pair of distinct vertices u and v, we write $u \prec v$, if u precedes v in a linear order. We also write $[v_1, v_2, \ldots, v_k]$ to denote that v_i precedes v_{i+1} for all $1 \leq i < k$. Assume that F is a set of $k \geq 2$ independent edges (s_i, t_i) with $s_i \prec t_i$, 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*. The edges of F form a *k-necklace*, if $[s_1, t_1, \ldots, s_k, t_k]$; see Fig. 2a. A preliminary result for queue layouts is the following.

Lemma 1 (Heath and Rosenberg [12]). *A linear order of the vertices of a graph admits a k-queue layout if and only if there exists no $(k+1)$-rainbow.*

Table 1. Queue numbers of various subclasses of planar graphs

Graph class	Upper bound		Lower bound	
	Old	New	Old	New
Tree	1 [12]		1 [12]	
Outerplanar	2 [11]		2 [12]	
Series-parallel	3 [18]		3 [21]	
Planar 3-tree	7 [21]	**5** [Theorem 1]	3 [21]	**4** [Theorem 2]
Planar	$\mathcal{O}(\log n)$ [4]		3 [21]	**4** [Theorem 2]

Central in our approach is also the following construction by Dujmović et al. [7] for internally-triangulated outerplane graphs; for an illustration see Figs. 2b–c.

Lemma 2 (Dujmović, Pór, Wood [7]). *Every internally-triangulated outerplane graph, G, admits a straight-line outerplanar drawing, $\Gamma(G)$, such that the y-coordinates of 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 construction of Lemma 2, where G is an internally triangulated outerplane graph. Up to renaming of the vertices of this face, we may assume that $|y(u)-y(v)| = |y(u)-y(w)| = 1$, $|y(v)-y(w)| = 2$ and $y(v) > y(w)$. We refer to vertex u as to the *anchor* of the face $\langle u, v, w\rangle$ of $\Gamma(G)$; 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: (i) 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 $\Gamma(G)$, (ii) edge (u, v) is assigned to the first (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 $\Gamma(G)$.

Finally, let $\langle u, v, w\rangle$ and $\langle u', v', w'\rangle$ be two 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. If u and u' are distinct and $u \prec u'$ in the 2-queue layout, then $v \prec v'$ (if $v \neq v'$) and $w \prec w'$ (if $w \neq w'$). The property clearly holds, if u and u' do not have the same y-coordinate. Otherwise, the property holds, since $\Gamma(G)$ is planar.

2 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 [21] to compute 7-queue layouts for general (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

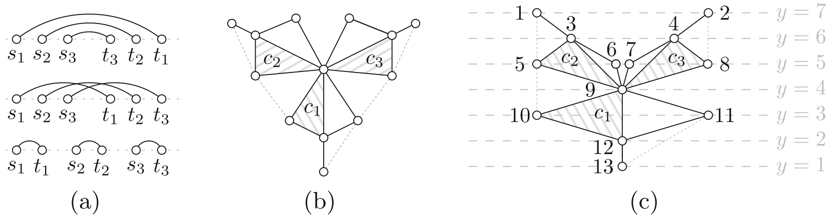


Fig. 2. (a) 3-rainbow, 3-twist and 3-necklace (from top to bottom); (b) an internally-triangulated outerplane graph G_0; the dotted-gray edges are added to make it biconnected; its gray-shaded faces contain components c_1, c_2 and c_3 of G_1; (c) the drawing $\Gamma(G_0)$ by Lemma 2; the vertex-labels 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.

Wiechert [21] into a *peeling-into-levels* approach (see, e.g., [23]), according to which the vertices and the edges of the input graph are partitioned as follows: (i) vertices incident to the outerface are at level zero, (ii) vertices incident to the outerface of the graph induced by deleting all vertices of levels $0, \dots, i-1$ are at level i, (iii) edges between same-level vertices are called *level edges*, and (iv) 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. Then, we extend our approach to more than two levels. We conclude by discussing the differences between the approach of Wiechert [21] and ours; we also describe which properties of planar 3-trees we exploited to reduce the required number of queues.

The Two-Level Case. We start with the (intuitively easier) case in which the given planar 3-tree G consists of two levels, 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. 2b): (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 G_0. Without loss of generality we may also assume that G_0 is biconnected, as otherwise we can augment it to being biconnected by adding (level-L_0) edges without affecting its outerplanarity. Note that in a planar 3-tree, graph G_0 is simply a triangle (and not an outerplane graph, as we have assumed), and as a result G_1 is a single outerplane component. Our algorithm maintains the following invariants:

I.1 the linear order is such that all vertices of L_0 precede all vertices of L_1;
I.2 the level edges use two queues, $\mathcal{Q}_0$ and $\mathcal{Q}_1$;
I.3 the binding edges use three queues, $\mathcal{Q}_2$, $\mathcal{Q}_3$, and $\mathcal{Q}_4$.

In the following lemma, we show how to determine a (partial) linear order of the vertices of levels L_0 and L_1 that satisfies the first two invariants of our algorithm.

Lemma 3. *There is an order of vertices of level L_0 and a partial order of vertices of level L_1 such that I.1 and I.2 are satisfied.*

Proof. To compute an order that satisfies I.1, we construct two orders, one for the vertices of level L_0 (that satisfies I.2) and one for the vertices of level L_1 (that also satisfies I.2), and then we concatenate them so that the vertices of L_0 precede the vertices of L_1.

To compute an order of the vertices of L_0 satisfying I.2, we apply Lemma 2, as by our initial assumption A.1, graph G_0 is internally-triangulated and outerplane. Thus, I.2 is satisfied for the vertices of level L_0. To compute an order of the vertices of L_1 satisfying I.2, we apply Lemma 2 individually for every connected component of G_1, which can be done by our initial assumption A.2. Then the resulting orders are concatenated (as defined by next Lemma 4). Since for every two connected components of G_1, all vertices of the first one either precede or follow all vertices of the second one, we can use the same two queues (denoted by $\mathcal{Q}_0$ and $\mathcal{Q}_1$ in I.2) for all the vertices of L_1. Therefore, I.2 is satisfied. □

Next, we complete the order of the vertices of G, in a way that the binding edges between L_0 and L_1 require at most three additional queues so as to satisfy I.3.

Lemma 4. *Given the linear order of the vertices of level L_0 and the partial order of the vertices of level L_1 produced by Lemma 3, there is a total order of the vertices of L_0 and L_1 that extends their partial orders and an assignment of the binding edges between L_0 and L_1 into three queues such that I.3 is satisfied.*

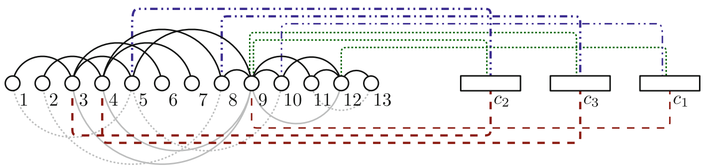


Fig. 3. The 5-queue layout for the graph in Fig. 2; since $5 \prec 8$ and $8 \prec 10$ in the order of the vertices of level L_0 as seen in Fig. 2, c_2 precedes c_3, and c_3 precedes c_1. (Color figure online)

Proof. 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 G_0. 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 ones incident to v to queue $\mathcal{Q}_3$ and the ones incident to w to queue $\mathcal{Q}_4$; see the blue, red, and green edges in Fig. 3.

Next we describe how to compute the relative order of the connected components of G_1. Let c and c' be two such components. By our initial assumption A.2, c and c' reside within two triangular faces $\langle u, v, w \rangle$ and $\langle u', v', w' \rangle$ of G_0. Assume that u and u' are the anchors of the two faces, v, v' are top and w, w' are bottom vertices. If $u \neq u'$, then c precedes c' if and only if $u \prec u'$ in the order of L_0.

If $u = u'$, we have $v \neq v'$ or $w \neq w'$. If $v \neq v'$, then c precedes c' if and only if $v \prec v'$ in the order of L_0. Otherwise (that is, $u = u'$ and $v = v'$), c precedes c' if and only if $w \prec w'$ in the order of L_0. We claim that for the resulting order of L_1, I.3 is satisfied, that is, no two edges of each of $\mathcal{Q}_2$, $\mathcal{Q}_3$ and $\mathcal{Q}_4$ are nested.

We start our proof with $\mathcal{Q}_2$. 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 in Fig. 3 incident to 5 and 8). By construction of $\mathcal{Q}_2$, x and x' are anchors of two different faces f_x and $f_{x'}$ of G_0 (see the faces of Fig. 2c that contain c_2 and c_3). Without loss of generality we assume that $x \prec x'$ in the order of L_0. Then, the two components c_y and $c_{y'}$ of G_1, that reside within f_x and $f_{x'}$ and contain y and y', are such that all vertices of c_y precede all vertices of $c_{y'}$ (in Fig. 3, $x = 5$ 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') do not nest.

We continue our proof with $\mathcal{Q}_3$ (the proof for $\mathcal{Q}_4$ is similar). 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. 3 incident to 3 and 4). By construction of $\mathcal{Q}_3$, x and x' are the top vertices of two different faces f_x and $f_{x'}$ of G_0 (see the faces of Fig. 2c that contain c_2 and c_3). Let c_y and $c_{y'}$ be the components of G_1 that reside within f_x and $f_{x'}$ and contain y and y'. Finally, let u and u' be the anchors of f_x and $f_{x'}$, respectively. Suppose first that $u \neq u'$ and assume that $u \prec u'$ in the order of L_0. Since $u \prec u'$, it follows that $x \prec x'$ and that all vertices of c_y precede all vertices of $c_{y'}$ (in Fig. 3, $u = 5$ 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 (x, y) and (x', y') are not nested. Suppose now that $u = u'$ and assume that $x \prec x'$ in the order of L_0. Since $u = u'$ and $x \prec x'$, all vertices of c_y precede all vertices of $c_{y'}$. Since $y \in c_y$ and $y' \in c_{y'}$, it follows that (x, y) and (x', y') are not nested. Hence, I.3 is satisfied, which concludes the proof. □

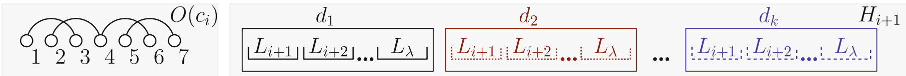
Lemmas 3 and 4 conclude the two-level case. Before we proceed with the multi-level case, we make a useful observation. To satisfy I.3, we did not impose any restriction on the order of the vertices of each connected component of G_1 (any order that satisfies I.2 for level L_1 would be suitable for us, that is, not necessarily the one constructed by Lemma 2). What we fixed, was the relative order of these components. We are now ready to proceed to the multi-level case.

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 $\lambda \geq 2$. Let G_i be the subgraph of G induced by the vertices of level L_i; $i = 0, 1, \ldots, \lambda$. The connected components of each graph G_i are internally-triangulated outerplane graphs that are not necessarily biconnected: Clearly, this holds for G_0, which is a triangle. Assuming that for some $i = 1, \ldots, \lambda$, graph G_{i-1} has the claimed property, we observe that each connected component of G_i resides within a facial triangle of G_{i-1}. Since each non-empty facial triangle of G_{i-1} in G induces a planar 3-tree [13], 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.

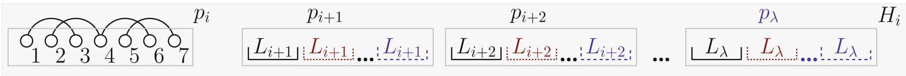
For the recursive step of our algorithm, assume that for some $i = 0, \ldots, \lambda - 1$ we have a 5-queue layout for each of the connected components of the graph H_{i+1} induced by the vertices of $L_{i+1}, \ldots, L_\lambda$, that satisfies the following invariants.

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

Based on these layouts, we show how to construct a 5-queue layout (satisfying M.1–M.3) for each of the connected components of the graph H_i induced by the vertices of $L_i, \ldots, L_\lambda$. Let C_i be such a component. By definition, C_i is delimited by a connected component c_i of G_i which is internally-triangulated and outerplane. If none of the faces of 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 c_i contain connected components of H_{i+1}. By M.1–M.3, 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 c_i.



(a) For each of $d_1, \ldots, d_k$ all vertices of L_j precede all vertices of L_{j+1}; $j = i+1, \ldots, \lambda-1$



(b) The computed linear order based on $p_i, \ldots, p_\lambda$

Fig. 4. Illustrations for the proof of Theorem 1.

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 we made in the two-level case, this will result in: (a) a linear order $\mathcal{O}(c_i)$ of the vertices of c_i, (b) a relative order of the components $d_1, \ldots, d_k$, (c) an assignment of the (level-L_i) edges of c_i into $\mathcal{Q}_0$ and $\mathcal{Q}_1$, and (d) 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 assume that $d_1, \ldots, d_k$ is the computed order of these components; see Fig. 4a.

By (c) and (d), all edges of C_i are assigned to $\mathcal{Q}_0, \ldots, \mathcal{Q}_4$, since the edges of $d_1, \ldots, d_k$ have been recursively assigned to these queues. Next, we partition the order of vertices of C_i into $\lambda - i + 1$ disjoint intervals, say $p_i, \ldots, p_\lambda$, such that p_μ precedes p_ν if and only if $\mu \prec \nu$. All the (level-L_i) vertices of c_i are contained in p_i in the order $\mathcal{O}(c_i)$ by (a). For $j = i+1, \ldots, \lambda$, p_j contains the vertices of L_j of each of the components $d_1, \ldots, d_k$, such that the vertices of L_j of d_μ precede the vertices of L_j of d_ν if and only if $\mu \prec \nu$; see Fig. 4b. The proof that M.1–M.3 are satisfied can be found in the full version [1]. We summarize in the following.

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

We note here that queue layouts are closely related to track layouts; for definitions refer to [7]. The following result follows immediately from a known result by Dujmović, Morin, Wood [6]; see the full version [1] for details.

Corollary 1. *The track number of a planar 3-tree is at most* 4000.

Differences with Wiechert's Algorithm. Wiechert's algorithm [21] builds upon a previous algorithm by Dujmović et al. [6]. Both yield queue layouts for general k-trees, using the breadth-first search (BFS) starting from an arbitrary vertex r of G. For each $d > 0$ and each connected component C induced by the vertices at distance d from r, create a node (called *bag*) "containing" all vertices of C; two bags are adjacent if there is an edge of G between them. For a k-tree, the result is a tree of bags T, called *tree-partition*, so that (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 [21] 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 G is a planar 3-tree and the BFS is started from a dummy vertex incident to the three outervertices of 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.

To reduce the number of queues, we observed that in G (i) every node of T induces a connected outerplanar graph, while (ii) each clique of size three by P.2 is a triangular face of 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.

3 The Lower Bound

In the following, we prove that the queue number of planar 3-trees is at least four. To this end, we will define recursively a subgraph of a planar 3-tree G and we will show that it contains at least one 4-rainbow in any ordering. 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, say A and B, which we assume to be neighboring. We refer to these edges as (s,t)-*edges*.

As a next step, we *stellate* each triangle $\langle A, s_i, t_i \rangle$ 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

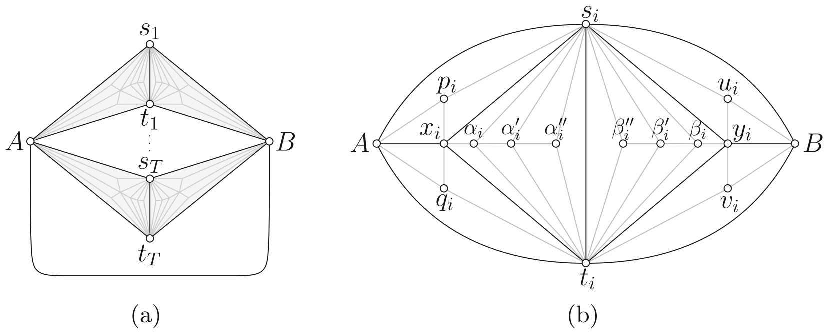


Fig. 5. Construction of graph G_T: Each gray subgraph in (a) corresponds to a copy of the graph of (b).

stellate each triangle $\langle B, s_i, t_i \rangle$ with a vertex y_i. Afterwards, we add one more level, that is, we stellate each of the triangles $\langle A, s_i, t_i \rangle$, $\langle B, s_i, t_i \rangle$, $\langle A, x_i, s_i \rangle$, $\langle A, x_i, t_i \rangle$, $\langle B, y_i, s_i \rangle$ and $\langle B, y_i, t_i \rangle$ with vertices α_i, β_i, p_i, q_i, u_i and v_i, respectively; see Fig. 5b. We 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.

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 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 to A and B. Then, 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) $A \prec s_i \prec t_i \prec B$.

By the pigeonhole principle and by setting $T = 6l$, we may claim that at least one of the permutations P.1–P.6 applies to at least l edges. We will show that if too many (s, t)-edges share one of the permutations P.1–P.5, then there exists a 4-rainbow, contradicting the fact that $\mathcal{Q}$ is a 3-queue layout for G_T. This implies that if T is large enough, then for at least one (s, t)-edge of G_T permutation P.6 applies. Based on this fact, we describe later how to augment the graph that we have constructed so far using a recursive construction 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 5. *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 the lemma holds. 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. Assume w.l.o.g. 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 right of t_1. Since no r-twist exists, vertex s_r is to the right of t_1. Thus, (s_1, t_1) and (s_r, t_r) do not cross. The removal of

$(s_1, t_1), \ldots, (s_{r-1}, t_{r-1})$ makes s_r first. By applying this argument $r-1$ times, we obtain 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. □

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

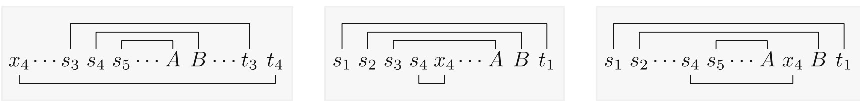
Corollary 2. *Every k-queue layout with at least kr^2 independent edges contains at least one r-twist or at least one r-necklace.*

We exploit this result for permutations P.1–P.6 as follows. Recall that $\mathcal{Q}$ is a 3-queue layout for 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 G_T share the same permutation. Moreover, these edges are by construction independent. Therefore, by Corollary 2 at least r of them form a necklace or a twist (while also sharing the same permutation). In the following, we show that if r (s,t)-edges, say w.l.o.g. $(s_1, t_1), \ldots, (s_r, t_r)$, form a necklace or a twist (for an appropriate choice of r) and simultaneously share one of the permutations P.1–P.5, then a 4-rainbow is inevitably induced, which contradicts the fact that $\mathcal{Q}$ is a 3-queue layout. We consider each case separately.

Case P.1: Let $r = 8$. It suffices to consider the case, in which $(s_1, t_1), \ldots, (s_8, t_8)$ form a twist, since in general for $r > 1$ the necklace case is impossible. Hence, the order is $[s_1 \ldots s_8 A B t_1 \ldots t_8]$. We show that x_4 always yields a 4-rainbow; Fig. 6 shows the three subcases arising when x_4 is such that $x_4 \prec B$ holds. Clearly, each yields a 4-rainbow. Since we did not use the edge (x_4, A), by symmetry, a 4-rainbow is also obtained when $B \prec x_4$.

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 $[A s_1 \ldots s_8 B t_1 \ldots t_8]$. One may verify that placing x_4 and x_5 to the left of t_8 always results in a 4-rainbow (see the full version [1] for details). For the case in which x_4 and x_5 are preceded by t_8, we distinguish between if $x_4 \prec x_5$ holds or not. Both result in a 4-rainbow.

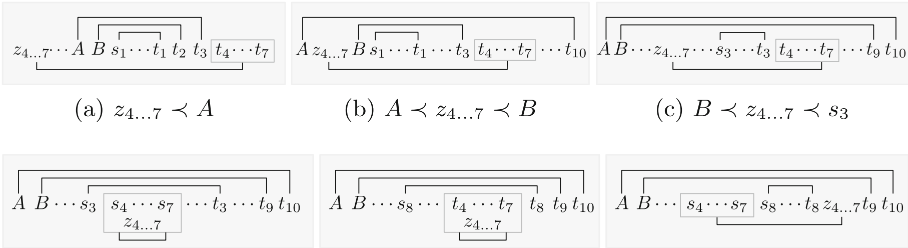
Case P.3: This case can be ruled out like Case P.2 due to symmetry.



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

Fig. 6. Illustration for the Case P.1 when $x_4 \prec B$ holds.

Case P.4: Let $r = 10$. We distinguish two subcases based on whether the edges $(s_1, t_1), \ldots, (s_{10}, t_{10})$ form a twist or a necklace (in contrast to the previous case, here both cases are possible).



(a) $z_{4...7} \prec A$ (b) $A \prec z_{4...7} \prec B$ (c) $B \prec z_{4...7} \prec s_3$

(d) $s_3 \prec z_{4...7} \prec t_3$ (e) $t_3 \prec z_{4...7} \prec t_8$ (f) $t_8 \prec z_{4...7} \prec t_9$

Fig. 7. Illustration for the Case P.4 when $z_{4...7} \prec t_9$ holds.

We start with the twist case. Hence, the order is $[ABs_1 \dots s_{10}t_1 \dots t_{10}]$. Let $Z_{4...7} = \{x_4, \dots, x_7\} \cup \{y_4, \dots y_7\}$ and let $z_{4...7}$ be any element of $Z_{4...7}$. Similar to the previous case, we sweep from left to right and rule out easy subcases. However, we have to ensure that we do not use any edge from $z_{4...7}$ to A or B in order to keep the roles of x_i and y_i interchangeable. Figure 7 shows that we may assume that $t_9 \prec z_{4...7}$, that is, all $x_4, \dots, x_7$ and $y_4, \dots, y_7$ are preceded by t_9.

Next, we show that we can always construct a 3-rainbow spanning (s_8, t_8), which then yields the desired 4-rainbow. Let us take a closer look at the ordering of the 8 vertices in $Z_{4...7}$. To prevent the creation of a 3-rainbow that spans (s_8, t_8), we claim that the ordering has to comply with two requirements: (R.1) the indices of the first 7 elements of $Z_{4...7}$ are non-decreasing, and (R.2) for the last 7 elements of $Z_{4...7}$, it must hold that all x precede all y. Assume to the contrary, that R.1 does not hold. Hence, there exists a pair of vertices, say w.l.o.g $x_j \prec x_i$, with $i < j$ and x_i is not the last element of $Z_{4...7}$. Then, $[s_i \dots s_j \dots x_j \dots x_i]$ forms a 2-rainbow and together with the last element of $Z_{4...7}$ that is adjacent to either A or B, we obtain a 3-rainbow spanning (s_8, t_8); a contradiction. Assume now that R.2 does not hold. Then, there exists a pair $y_i \prec x_j$ with y_i not being the first element. Let the first element be x_l. Then, $[A \dots B \dots s_l \dots x_l \dots y_i \dots x_j]$ is a 3-rainbow spanning (s_8, t_8); a contradiction.

Now, we show that R.1 and R.2 cannot simultaneously hold, which implies the existence of a 4-rainbow. Consider the last element of $Z_{4...7}$. Assume that R.1 and R.2 both hold. By R.2, we may deduce that the last three elements of $Z_{4...7}$ belong to $\{y_4, \dots y_7\}$. 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 $Z_{4...7}$.

We continue with the necklace case. Here, the order is $[ABs_1t_1 \dots s_{10}t_{10}]$. We make several observations about the ordering in the form of propositions; their formal proofs can be found in the full version [1].

Proposition 1. *Let w be a neighbor of s_i and t_i for $3 \le i \le 8$. Then, either $s_{i-1} \prec w \prec t_{i+1}$ holds, or $s_{10} \prec w$.*

Proposition 2. *Let w and z be two vertices 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: $s_{10} \prec w$ or $s_{10} \prec z$.*

Proposition 3. *Let w, z be neighbors of both s_i, t_i, for $3 \leq i \leq 8$. Then, at most one of w and z is between s_{i-1} and s_i or between t_i and t_{i-1}. Furthermore, if one of w and z is between s_{i-1} and s_i or between t_i and t_{i-1}, then the other is not between s_i and t_i.*

Proposition 4. *For $4 \leq i \leq 8$, 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}.*

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

By Proposition 4, for $4 \leq i \leq 8$, each vertex from $\{s_i, t_i, x_i, y_i, p_i, q_i, u_i, v_i\}$ is in (s_{i-1}, t_{i+1}). 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. Assume w.l.o.g. that $x_i \prec y_i$. Then, by Proposition 3, one of the following two conditions hold: (i) $x_i \prec s_i \prec t_i \prec y_i$, (ii) $s_i \prec x_i \prec y_i \prec t_i$; see Fig. 8. In both cases, 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. 8. 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.

Case P.5: This case can be ruled out like 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 $1 \leq i_0 \leq T$ 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 G_T, we introduce a new copy of 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 there exists $1 \leq i'_0 \leq T$ such that $s_{i_0} \prec s'_{i_0} \prec t'_{i_0} \prec t_{i_0}$. Hence, the edges (A, B), (s_{i_0}, t_{i_0}) and (s'_{i_0}, t'_{i_0}) 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, either G_T does not admit a 3-queue layout, as we initially assumed, or G''_T does not admit a 3-queue layout. In both cases, Theorem 2 follows.

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

4 Conclusions

In this work, we presented improved bounds on the queue number of planar 3-trees. Three main open problems 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 [21]. Finally, the third problem is the central question in the area. Is the queue number of general planar graphs (that is, that are not necessarily planar 3-trees) bounded by a constant?

References

1. Alam, J.M., Bekos, M.A., Gronemann, M., Kaufmann, M., Pupyrev, S.: Queue layouts of planar 3-trees. CoRR abs/1808.10841 (2018)
2. 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)
3. Di Battista, G., Frati, F., Pach, J.: On the queue number of planar graphs. SIAM J. Comput. **42**(6), 2243–2285 (2013)
4. Dujmović, V.: Graph layouts via layered separators. J. Comb. Theory Ser. B **110**, 79–89 (2015)
5. Dujmović, V., Frati, F.: Stack and queue layouts via layered separators. J. Graph Algorithms Appl. **22**(1), 89–99 (2018)
6. Dujmović, V., Morin, P., Wood, D.R.: Layout of graphs with bounded tree-width. SIAM J. Comput. **34**(3), 553–579 (2005)
7. Dujmović, V., Pór, A., Wood, D.R.: Track layouts of graphs. Discret. Math. Theor. Comput. Sci. **6**(2), 497–522 (2004)
8. Dujmović, V., Wood, D.R.: Tree-partitions of k-trees with applications in graph layout. In: Bodlaender, H.L. (ed.) WG 2003. LNCS, vol. 2880, pp. 205–217. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39890-5_18
9. Dujmović, V., Wood, D.R.: Stacks, queues and tracks: layouts of graph subdivisions. Discret. Math. Theor. Comput. Sci. **7**(1), 155–202 (2005)
10. Hasunuma, T.: Laying out iterated line digraphs using queues. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 202–213. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24595-7_19
11. 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)
12. Heath, L.S., Rosenberg, A.L.: Laying out graphs using queues. SIAM J. Comput. **21**(5), 927–958 (1992)
13. Mondal, D., Nishat, R.I., Rahman, M.S., Alam, M.J.: Minimum-area drawings of plane 3-trees. J. Graph Algorithms Appl. **15**(2), 177–204 (2011)
14. 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. Congressus Numerantium, vol. VIII, p. 459 (1973)

15. Pach, J., Thiele, T., Tóth, G.: Three-dimensional grid drawings of graphs. In: DiBattista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 47–51. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63938-1_49
16. Pemmaraju, S.V.: Exploring the powers of stacks and queues via graph layouts. Ph.D. thesis, Virginia Tech (1992)
17. Pupyrev, S.: Mixed linear layouts of planar graphs. In: Frati, F., Ma, K.-L. (eds.) GD 2017. LNCS, vol. 10692, pp. 197–209. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73915-1_17
18. Rengarajan, S., Veni Madhavan, C.E.: Stack and queue number of 2-trees. In: Du, D.-Z., Li, M. (eds.) COCOON 1995. LNCS, vol. 959, pp. 203–212. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0030834
19. Shahrokhi, F., Shi, W.: On crossing sets, disjoint sets, and pagenumber. J. Algorithms **34**(1), 40–53 (2000)
20. Tarjan, R.E.: Sorting using networks of queues and stacks. J. ACM **19**(2), 341–346 (1972)
21. Wiechert, V.: On the queue-number of graphs with bounded tree-width. Electr. J. Comb. **24**(1) (2017). P1.65
22. Wood, D.R.: Queue layouts, tree-width, and three-dimensional graph drawing. In: Agrawal, M., Seth, A. (eds.) FSTTCS 2002. LNCS, vol. 2556, pp. 348–359. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36206-1_31
23. Yannakakis, M.: Embedding planar graphs in four pages. J. Comput. Syst. Sci. **38**(1), 36–67 (1989)