On the queue-number of graphs with bounded tree-width

Veit Wiechert*

Institut für Mathematik
Technische Universität Berlin
Berlin, Germany

wiechert@math.tu-berlin.de

Submitted: Aug 31, 2016; Accepted: Mar 24, 2017; Published: Mar 31, 2017
Mathematics Subject Classifications: 05C62

Abstract

A *queue layout* of a graph consists of a linear order of the vertices and an assignment of the edges to *queues*, such that no two edges in a single queue are nested. The minimum number of queues needed in a queue layout of a graph is called its *queue-number*.

We show that for each $k \geqslant 0$, graphs with tree-width at most k have queue-number at most $2^k - 1$. This improves upon double exponential upper bounds due to Dujmović et al. and Giacomo et al. As a consequence we obtain that these graphs have track-number at most $2^{O(k^2)}$.

We complement these results by a construction of k-trees that have queue-number at least $k+1$. Already in the case $k = 2$ this is an improvement to existing results and solves a problem of Rengarajan and Veni Madhavan, namely, that the maximal queue-number of 2-trees is equal to 3.

1 Introduction

A *queue layout* of a graph consists of a linear order of the vertices and an assignment of the edges to *queues*, such that no two edges in a single queue are nested. This is a dual concept to stack layouts, which are defined similarly, except that no two edges in a single stack may cross. The minimum number of queues (stacks) needed in a queue layout (stack layout) of a graph is called its *queue-number* (*stack-number*).

*V. Wiechert is supported by the Deutsche Forschungsgemeinschaft within the research training group 'Methods for Discrete Structures' (GRK 1408).

The notion of queue-number was introduced by Heath and Rosenberg [19] in 1992. Queue layouts were implicitly used before and have applications in fault-tolerant processing, sorting with parallel queues, matrix computations, and scheduling parallel processors (see [18, 19, 22] for more details).

In their seminal paper, Heath and Rosenberg characterize graphs admitting a 1-queue layout as so-called *arched leveled-planar* graphs and show that it is NP-hard to recognize them. This contrasts the situation for graphs with a 1-stack layout, since these graphs are exactly the outerplanar graphs [2] and hence can be recognized in linear time. Several other results relating these two types of layouts are studied in [18]. While planar graphs have stack-number at most 4 [28], it remains open whether the queue-number of planar graphs is bounded by a constant. This is one of the most tantalizing problems regarding queue layouts and was conjectured to be true by Heath and colleagues [18, 19]. In fact, they even conjecture that the queue-number can be bounded by a function of the stack-number; see [11] for a comprehensive study of this question.

There are some partial results towards a positive resolution of this conjecture. Improving on an earlier result by Di Battista et al. [5], Dujmović showed that planar graphs have queue-number $O(\log n)$ [7]. This result was extended to graphs with bounded Euler genus by Dujmović, Morin, and Wood [13]. In the more general case of graphs that exclude a fixed graph as a minor they obtained a $\log^{O(1)} n$ bound on the queue-number.

In this paper we focus on queue layouts of bounded tree-width graphs. A comprehensive list of references to papers about further aspects of queue layouts can be found in [9].

1.1 Queue layouts and tree-width

Several graph classes are known to have bounded queue-number. For example, trees have 1-queue layouts [19], outerplanar graphs have 2-queue layouts [18], partial 2-trees (that is, series-parallel graphs) have 3-queue layouts [24], and graphs of path-width at most p have p-queue layouts [8].

All these graphs have bounded tree-width and it was first asked by Ganley and Heath [15] whether there is a constant upper bound on the queue-number of bounded tree-width graphs (for the stack-number this is true as shown in [15]). This question was answered in the affirmative for graphs that additionally have bounded maximum degree by Wood [26], and later in full by Dujmović, Morin, and Wood [8]. In the latter result, Dujmović and Wood establish the upper bound $3^k 6^{(4^k-3k-1)/9} - 1$ on the queue-number of graphs with tree-width at most k. In fact, they provide upper bounds as solutions of a system of equations. Giacomo et al. [6] present an improved system of equations with smaller solutions for each $k \geqslant 1$ (without trying to find a nice expression for the corresponding upper bound), but still being double exponential in k. Answering a question of Dujmović et al. [8] we prove a single exponential upper bound.

Theorem 1. *Let $k \geqslant 0$. For all graphs G with tree-width at most k,*

$$\mathrm{qn}(G) \leqslant 2^k - 1,$$

where $\mathrm{qn}(G)$ *denotes the queue-number of* G.

Observe that this bound is not only asymptotically much smaller than previous best bounds, it is also strong for small values of k. As special cases we obtain the above mentioned results that trees and partial 2-trees have queue-number at most 1 and 3, respectively. (And as we will show, 3 is best possible in the latter case). Interestingly, in his PhD thesis, Pemmaraju [22] conjectured that a certain family of planar 3-trees (the *stellated triangles*) has queue-number $\Omega(\log n)$. Of course, this conjecture has already been disproved by Dujmović et al. with their upper bound for k-trees. However, now with Theorem 1 we have that planar 3-trees (and more generally partial 3-trees) have queue-number at most 7.

1.2 Track layouts

For the proofs of their upper bounds, Dujmović et al. and Giacomo et al. use track layouts of graphs, which we define now. Let G be graph and let $\{V_i : i = 1, \dots, \ell\}$ be a partition of $V(G)$ into independent sets. A set V_i combined with a linear order $<_i$ of its elements is a *track* of G. Then a set of tracks $\{(V_i, <_i) : i = 1, \dots, \ell\}$ is called a *track assignment* of G. Two edges ab and cd form an *X-crossing* in a track assignment $\{(V_i, <_i) : i = 1, \dots, \ell\}$ if there are $i, j \in \{1, \dots, \ell\}$ such that $a <_i c$ and $d <_j b$. A track assignment without an X-crossing is called a *track layout*. The minimum number of tracks in a track layout of G is the *track-number* of G, which we denote by $\mathrm{tn}(G)$.

In [8] the upper bound $3^k 6^{(4^k-3k-1)/9}$ is actually shown for the track-number of graphs with tree-width at most k. Since the authors also show that $\mathrm{qn}(G) \leqslant \mathrm{tn}(G) - 1$ for every graph G, they obtain their bound for the queue-number from the track-number bound. Using Theorem 1, we prove the following result.

Theorem 2. *Let* $k \geqslant 0$. *For all graphs* G *with tree-width at most* k,

$$\mathrm{tn}(G) \leqslant (k+1)(2^{k+1}-2)^k.$$

Clearly, this $2^{O(k^2)}$ bound is asymptotically a big improvement upon the double exponential bound discussed before. However, for small values of k (that is, $k \in \{1, 2, 3\}$) better bounds are known. For example, Giacomo et al. [6] show that graphs of tree-width at most 2 admit a 15-track layout.

We prove Theorem 2 now. A proper coloring of the vertices of a graph G is *acyclic* if any two color classes induce a forest (so each cycle receives at least three colors). The minimum number of colors used in an acyclic coloring of G is the *acyclic chromatic number* of G. Dujmović et al. [8] obtained the following relationship between track-number and queue-number.

Lemma 3 ([8]). *Every graph* G *with acyclic chromatic number at most* c *and queue-number at most* q *has track-number*

$$\mathrm{tn}(G) \leqslant c(2q)^{c-1}.$$

It is well-known that graphs of tree-width at most k have acyclic chromatic number at most $k+1$. Using this, we immediately obtain a proof of our claimed upper bound on the track-number of bounded tree-width graphs.

Proof of Theorem 2. Combine Theorem 1 and Lemma 3. □

1.3 Three-dimensional drawings

Another reason to study queue-number and track-number is their connection to three-dimensional drawings of graphs. A *three-dimensional straight-line grid drawing* is an embedding of the vertices onto distinct points of the grid $\mathbb{Z}^3$ with edges represented as straight line segments that connect their end-vertices, such that any two edges intersect only if they share a common end-vertex, and a vertex only intersects an edge if it is the end-vertex of that edge.

Using the *moment curve* one can show that each graph has such a drawing. Therefore, we would like to minimize the volume of the bounding box defined by the grid points used for the embedding. Cohen et al. [4] showed that the complete graph K_n requires $\Theta(n^3)$ volume. Graphs with bounded chromatic number can be drawn on the three-dimensional grid with $O(n^2)$ volume, as shown by Pach et al. [21], and this is best possible for complete bipartite graphs. The latter result was improved by Bose et al. [3], who showed that graphs with n vertices and m edges need at least $\frac{1}{8}(n+m)$ volume. In particular, this implies that graphs with three-dimensional drawings of linear volume have only linearly many edges. Dujmović and Wood [12] showed that graphs with bounded degeneracy admit drawings with $O(n^{3/2})$ volume. A major open problem in this area, due to Felsner et al. [14], asks whether planar graphs can be drawn with linear volume. The best known volume bound for this problem is $O(n \log n)$ and was given by Dujmović [7] (see also [13] for an extension of this bound to apex-minor free graphs, and an $n \log^{O(1)} n$ bound for proper minor-closed families). Dujmović et al. [8] argue that if planar graphs have bounded queue-number, then this would imply a linear bound on the required volume for three-dimensional drawings of planar graphs.

Let us focus on graphs of bounded tree-width now. For outerplanar graphs, which have tree-width at most 2, Felsner et al. [14] proved a linear volume bound. Their argument is based on track layouts and a technique called "wrapping". Dujmović and Wood [10] showed that c-colorable graphs with track-number at most t have $O(c^2 t) \times O(c) \times O(c^4 n)$ drawings, implying that bounded tree-width graphs can be drawn with linear volume. To be more precise, using the bounds on the track-number obtained by Dujmović and Wood one can deduce that graphs of tree-width at most k admit $O(k^2 t_k) \times O(k) \times O(k^4 n)$ drawings, where $t_k = 3^k \cdot 6^{(4^k - 3k - 1)/9}$. The resulting volume was slightly improved by Giacomo et al. [6] with their new bounds on track-number. Now, the aforementioned volume bound in terms of the track-number combined with our Theorem 2 significantly reduces the required volume for bounded tree-width graphs to $2^{O(k^2)} \times O(k) \times O(k^4 n)$.

1.4 Lower bounds on the queue-number

Heath and Rosenberg [19] gave a simple argument for the fact that the queue-number is always larger than a quarter of the average degree. In particular, graph families with more than linearly many edges have unbounded queue-number. A more subtle result in this direction is that graphs obtained from (in a certain sense) dense graphs by subdividing each edge a bounded number of times have unbounded queue-number, see Nešetřil et al. [20]. Thus, graphs with linearly many edges can have arbitrarily large queue-number, and this even holds in the special case of bounded degree graphs as shown by Wood [27]. Gregor et al. [16] proved that the n-dimensional hypercube has queue-number at least $(\frac{1}{2} - \epsilon)n - O(1/\epsilon)$, for every $\epsilon > 0$.

To the author's knowledge, the best published lower bound on the queue-number of planar graphs is 2 [24]. In fact, the example given is an outerplanar graph and hence has tree-width at most 2. In the same paper it is conjectured that there are planar graphs with queue-number 5 (without providing deep evidence for this).

For graphs of tree-width at most k the situation is similar. It is easy to see that complete graphs and complete bipartite graphs yield examples with queue-number at least $\lfloor \frac{k+1}{2} \rfloor$, but besides that no lower bounds depending on k have been discussed. (For the maximal track-number of graphs with tree-width at most k, a non-trivial $\Omega(k^2)$ lower bound is proven in [8].) For the special case of 2-trees we already mentioned that their queue-number is at most 3. The aforementioned example also shows that there are 2-trees with queue-number 2. We close this gap and thereby answer a question of Rengarajan and Veni Madhavan [24] with the following general lower bound.

Theorem 4. *For each $k \geqslant 2$, there is a k-tree with queue-number at least $k + 1$.*

In particular, since 2-trees are planar, there are planar graphs with queue-number at least 3.

1.5 Proof ideas and organization

For the proof of Theorem 1 we make use of tree-partitions, which were introduced by Seese [25] and independently by Halin [17]. A *tree-partition* of a graph is a partition of its vertex set into "bags", combined with an underlying tree (or forest) on the bags so that each edge of the graph is either contained within a bag, or it goes along an edge of the tree. The fact that k-trees admit tree-partitions such that each bag induces a $(k-1)$-tree (see [8]) allows us to apply induction. In contrast to the proofs of [6, 8], we do not construct a track layout as an intermediate step, but directly build a queue layout of the given graph.

The rest of the paper is organized as follows. In Section 2 we provide necessary definitions and basic propositions for our proofs. In Section 3 we prove Theorem 1. Then we show the lower bound of Theorem 4 in Section 4. We conclude the paper with some open problems in Section 5.

2 Preliminaries

In this section we introduce the necessary definitions and basic concepts for our main result.

2.1 Queue layouts

Let $G = (V, E)$ be a graph and let L be a linear order of the vertices of G. We say that edges $uv, u'v' \in E$ are *nested* with respect to L if $u < u' < v' < v$ or $u' < u < v < v'$ in L. A set Q of edges in G forms a *queue* with respect to L if no two edges of Q are nested in L. A *queue layout* of G is a linear order L of the vertices of G together with a partition of the edge set of G into queues with respect to L. The minimum number of queues in a queue layout of G is called the *queue-number* of G, and denoted by $\mathrm{qn}(G)$.

There is a different approach to the queue-number via k-rainbows. Given a linear order L of the vertices of a graph G, the edges $a_1b_1, \dots, a_kb_k$ form a *rainbow* of size k (or *k-rainbow*) if

$$a_1 < \cdots < a_k < b_k < \cdots < b_1$$

in L. Clearly, if k is the maximum size of a rainbow in L, then each queue layout using L as the linear order will consist of at least k queues. It is not hard to see that k queues suffice in this case.

Proposition 5 ([19]). *If G has no rainbow of size $k+1$ with respect to a given linear order L, then G has a queue layout using at most k queues with respect to L.*

As a consequence, the queue-number can be described as the minimum number taken over the maximal size of a rainbow in a linear order of $V(G)$.

2.2 Tree-width

Let $G = (V, E)$ be a graph. A *tree-decomposition* of G is a pair $(T, \{T_x\}_{x \in V})$ consisting of a tree T and a family of non-empty subtrees of T, such that $V(T_x) \cap V(T_y) \neq \emptyset$ for each edge $xy \in E$. The vertices of T are called *nodes*, and each node $u \in V(T)$ induces a *bag* $\{x \in V : u \in T_x\}$. The maximum size of a bag minus one is the *width* of the tree-decomposition. Then the *tree-width* of G can be defined as the minimum width of a tree-decomposition of G.

For our purposes it is convenient to follow the work of Dujmovic et al. [8] and define k-trees as introduced by Reed [23]. Given some fixed integer $k \geqslant 0$, a *k-tree* is defined recursively. The empty graph is a k-tree, and each graph obtained by adding a vertex v to a k-tree so that the adjacent vertices of v form a clique of size at most k is also a k-tree. (Arnborg and Proskurowski [1] introduced k-trees in a slightly more restrictive way. They start with defining a k-clique to be a k-tree, and each graph obtained from a k-tree by adding a vertex being adjacent to a k-clique is also a k-tree. Sometimes the notion of *strict* k-trees is used for this more restrictive version.) A subgraph of a k-tree is called a *partial k-tree*. It is well-known that a graph has tree-width at most k if and only

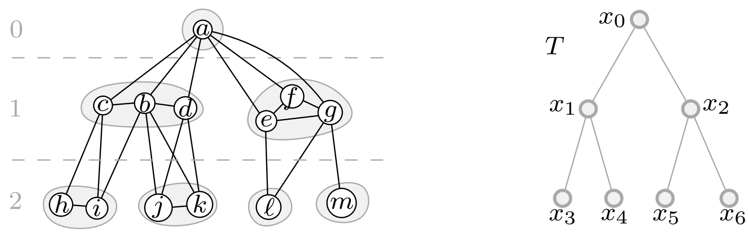


Figure 1: A tree-partition of a 3-tree whose vertices are labeled with $a, b, \ldots, m$. The figure on the right shows the underlying tree of the tree-partition.

if it is a partial k-tree. Moreover, k-trees are *chordal* graphs, that is, they do not contain a cycle on more than three vertices as an induced subgraph.

2.3 Tree-partitions

For the construction of a queue layout in our main proof, we do not use a specific tree-decomposition, but instead we use a tree-partition. Given a graph G, a *tree-partition* of G is a pair consisting of a tree T (or forest) and a partition of $V(G)$ into sets $\{T_x : x \in V(T)\}$ being indexed by the vertices of T, such that for each edge uv in G we either have that $u, v \in T_x$ for some $x \in V(T)$, or there is an edge xy of T with $u \in T_x$ and $v \in T_y$. We refer to the vertices of T as *nodes*, and say that T_x $(x \in V(T))$ is a *bag* of the tree-partition. By $G[T_x]$ we denote the subgraph of G induced by the vertices of T_x. For an example of a tree-partition see Figure 1. A fixed tree-partition of G naturally divides the edges of G into two classes. If both endpoints of an edge are contained in the same bag, then we call it an *intrabag edge*. In the other case, so if the two endpoints lie in different bags, then we call it an *interbag edge*.

3 Upper bound – Proof of Theorem 1

First of all, note that it is enough to prove Theorem 1 for k-trees. Indeed, this follows from the two facts that each graph of tree-width k can be extended to a k-tree by adding edges to the graph (for example, by taking the *chordal completion* that minimizes the size of the maximum clique), and that the queue-number of a graph does not decrease under the addition of edges.

As noted before, our queue layout construction relies on tree-partitions that capture the structure of k-trees. The following theorem by Dujmović et al. will give us such a tree-partition.

Theorem 6 ([8]). *Let G be a k-tree. Then there is a rooted tree-partition $(T, \{T_x \colon x \in V(T)\})$ of G such that*

1. *for each node x of T, the induced subgraph $G[T_x]$ is a connected $(k-1)$-tree,*

2. *for each nonroot node $x \in T$, if $y \in T$ is the parent node of x in T then the vertices in T_y with a neighbor in T_x form a clique.*

Let us give a brief sketch of how one can obtain a tree-partition of a connected k-tree G as in the theorem. Fix an arbitrary vertex r of G and perform a *Breadth-first Search* (BFS) in G starting from r. For each $d \geqslant 0$ and each component induced by the vertices at distance d from r, we introduce a node and associate with this node a bag containing the vertices of the component. Two nodes become adjacent if their corresponding sets of vertices are joined by at least one edge of G. Using the chordality of G one can show that the constructed graph T on the nodes is indeed a tree, and that the vertices of each bag induce a $(k-1)$-tree. Note that the bag of the root node of T contains only one vertex (r in our case). The tree-partition in Figure 1 can be obtained with the described procedure by starting the BFS from vertex a.

Let $(T, \{T_x \colon x \in V(T)\})$ be a rooted tree-partition as in the previous theorem. For each nonroot node x of T, we denote by $p(x)$ the parent node of x in T. Moreover, we let C_x denote the clique in $T_{p(x)}$ according to item 2 of Theorem 6. For instance, in our example of Figure 1 we have that $p(x_3) = x_1$ and $C_{x_3} = \{b, c\}$.

We are now ready to prove Theorem 1. In fact, we show the following slightly stronger result.

Theorem 7. *Let $k \geqslant 0$. For each k-tree G, there is a queue layout using at most $t_k = 2^k - 1$ queues, such that for each $v \in V(G)$, edges with v as their right endpoint in the layout are assigned to pairwise different queues.*

Proof. We prove the theorem by induction on k. In the base case $k = 0$, the graph G has no edges and thus no queues are needed in a queue layout of G.

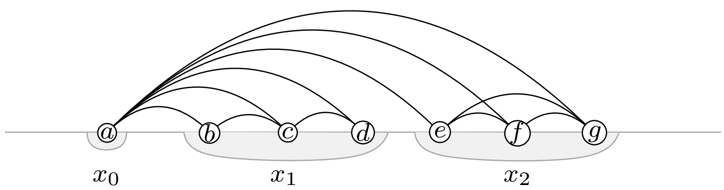
Suppose now that G is a k-tree for some $k \geqslant 1$, and that the theorem holds for $k-1$. We may assume that G is connected, since we can combine layouts of different components of G by putting them next to each other, and since we can resuse queues for different components. Let $(T, \{T_x \colon x \in V(T)\})$ be a tree-partition of G as given by Theorem 6, and denote the root of T by r. Then we can assign to each node of T a *depth* according to its distance to r in T (with r being at depth 0). We say that a vertex $v \in V(G)$ is at depth d if v is contained in a bag of some node at depth d.

In the following, we first construct a linear order L^G for the queue layout of G, and then we assign the edges to queues. Let us give some intuition of how we obtain L^G now. We build L^G by going through the depths one by one (starting with depth 0). That is to say, given the already produced linear order of vertices at depth $d-1$, we construct a linear order of vertices at depth d and append it to the right of the one already produced. To do so, we first specify a linear order L^T_d of the nodes at depth d in T, and then we replace each node x in L^T_d by the linear order of the layout obtained by applying induction to the $(k-1)$-tree $G[T_x]$.

Now let us be more precise. At depth 0 we only have the root node r of T, and hence we set L^T_0 to be the linear order consisting only of r. We apply induction on the $(k-1)$-tree $G[T_r]$ and obtain a linear order L^G_0 of vertices at depth 0 (as noted before, T_r actually contains only one vertex).

So suppose that we have built the linear order L^G_{d-1} containing all vertices at depth at most $d-1$ in G. Let L^T_{d-1} be the linear order of the nodes at depth $d-1$ that was produced in the last step of our procedure. We proceed by constructing L_d^T now.

As in a *lexicographical breadth-first ordering* (Lex-BFS ordering), we order the nodes according to their parent nodes. That is, for nodes x, y at depth d we set $x < y$ in L_d^T if $p(x) < p(y)$ in L_{d-1}^T. It remains to specify the order of nodes sharing a parent node. So suppose that $x_1, \ldots, x_\ell$ have the same parent node y at depth $d-1$. Consider the cliques $C_{x_1}, \ldots, C_{x_\ell}$ in T_y. For each $i \in \{1, \ldots, \ell\}$, let c_{x_i} be the rightmost vertex of C_{x_i} in L_{d-1}^G. Then we order $x_1, \ldots, x_\ell$ according to the positions of $c_{x_1}, \ldots, c_{x_\ell}$, which means that we set $x_i < x_j$ in L_d^T if c_{x_i} appears before c_{x_j} in L_{d-1}^G. Nodes with the same parent node and with the same rightmost vertex in their parent clique are still not ordered with this rule. We order those nodes arbitrarily so that L_d^T becomes a linear order of nodes at depth d. To illustrate this procedure, consider the following linear order, where vertices at depth at most 1 of our example from Figure 1 have been ordered so far.



Here we have $x_1 < x_2$ in L_1^T, and since $x_1 = p(x_3) = p(x_4)$ and $x_2 = p(x_5) = p(x_6)$, this implies that x_3, x_4 are placed before x_5, x_6 in L_2^T. As $c_{x_3} = c < d = c_{x_4}$ in the order, we set $x_3 < x_4$ in L_2^T. The order between x_5 and x_6 in L_2^T can be chosen arbitrarily as $c_{x_5} = c_{x_6} = g$.

By Theorem 6, the bag of each node x in the tree-partition induces a $(k-1)$-tree, which allows us to apply induction. Let L_x be the linear order of the queue layout obtained in this way. Now we replace each node x in L_d^T by the linear order L_x. We put the resulting order of vertices at depth d to the right of L_{d-1}^G, which yields a linear order L_d^G of all vertices at depth at most d. This concludes the step for vertices at depth d.

Iterating this construction until we reach the maximum depth d' of a vertex in G, we obtain the linear order $L_{d'}^G$ that contains every vertex of G. Let L^G be this linear order. Similarly, let L^T be the linear order of all the nodes of T obtained during the procedure. Recall that by our applied rules, L^T has the following properties. For nodes $x, y \in V(T)$ with depths $d(x)$ and $d(y)$, respectively,

$$\text{if } d(x) < d(y) \text{ in } T, \text{ then } x < y \text{ in } L^T, \tag{1}$$

$$\text{if } p(x) < p(y) \text{ in } L^T, \text{ then } x < y \text{ in } L^T. \tag{2}$$

Property (1) asserts that L^T is a BFS ordering, and combined with property (2) we have that L^T is a Lex-BFS ordering. Therefore, no two edges of T are nested in L^T. This has an immediate consequence for interbag edges as they go along edges of T. Let uv and $u'v'$ be interbag edges such that $u < v$ and $u' < v'$ in L^G. Then we have the property that if uv and $u'v'$ are nested in L^G, then u and u' are contained in the same bag of the tree-partition.

We now assign the edges of G to queues. For convenience, let us instead first color the edges with colors from $\{1, \ldots, 2t_{k-1} + 1\}$ and then show that each color class induces a queue with respect to L^G.

We start with the intrabag edges. For each bag T_x, we color the contained edges according to the queue assignment that is given by the induction hypothesis for the $(k-1)$-tree $G[T_x]$. We use the colors $1, \ldots, t_{k-1}$ for this coloring (so we reuse the same colors for different bags).

Let us continue with the interbag edges now, and let $uv \in E(G)$ be one of those. Say, u is at a smaller depth than v. Then there is a node x in T such that $v \in T_x$ and $u \in T_{p(x)}$. If $u = c_x$, then we color uv with $2t_{k-1} + 1$. Otherwise, if $u \neq c_x$, then we color uv with $i + t_{k-1}$, where $i \in \{1, \ldots, t_{k-1}\}$ is the color of the intrabag edge uc_x.

Claim. *For each color $c \in \{1, \ldots, 2t_{k-1} + 1\}$, the edges of G colored c form a queue with respect to L^G.*

Proof. Suppose for a contradiction that there are edges uv and $u'v'$ with color c that are nested in L^G. Say $u < u' < v' < v$ in L^G.

If $c \in \{1, \ldots, t_{k-1}\}$ then uv and $u'v'$ are both intrabag edges. However, if they lie within the same bag, then they cannot be nested as we used a valid queue layout from the induction hypothesis. And if they lie in different bags, then both endpoints of one edge lie before both endpoints of the other edge in L^G. Thus, the two edges are not nested in L^G, a contradiction.

Hence $c \geqslant t_{k-1} + 1$ and consequently uv and $u'v'$ are interbag edges. By the consequences of properties (1) and (2) for interbag edges, it follows that u and u' both are contained in the same bag. Suppose that this is bag T_y, and let $x, x' \in V(T)$ be such that $v \in T_x$ and $v' \in T_{x'}$. Note that $u \in C_x$ and $u' \in C_{x'}$. We distinguish two cases now.

First, suppose $c = 2t_{k-1} + 1$. Then u and u' are rightmost in L^G among vertices of C_x and $C_{x'}$, respectively. So we have $u = c_x$ and $u' = c_{x'}$, and hence $x \neq x'$. Recall that since x and x' share the parent y, they are ordered in L^T according to the positions of c_x and $c_{x'}$ in L^G. Thus, as $c_x = u < u' = c_{x'}$ in L^G, this implies $x < x'$ in L^T. It follows that vertices of T_x lie before vertices of $T_{x'}$ in L^G, a contradiction to our assumption $v' < v$ in L^G.

So we are left with the case $c \in \{t_{k-1} + 1, \ldots, 2t_{k-1}\}$. Let $i \in \{1, \ldots, t_{k-1}\}$ be such that $c = i + t_{k-1}$. This time we have $u \neq c_x$ and $u' \neq c_{x'}$. Since $u \in C_x$ and $u' \in C_{x'}$, it follows that $u < c_x$ and $u' < c_{x'}$ in L^G. By our coloring, edges uc_x and $u'c_{x'}$ are colored with i. This implies $c_x \neq c_{x'}$ as otherwise c_x is the right endpoint of two intrabag edges of the same color, which is contradicting the induction hypothesis. In particular, this yields $x \neq x'$. By our assumption that $v' < v$ in L^G, we conclude $x' < x$ in L^T. And since x and x' are ordered in L^T according to the positions of c_x and $c_{x'}$ in L^G, this in turn implies $c_{x'} < c_x$ in L^G. Together with $c_{x'}$ being the rightmost vertex of $C_{x'}$ in L^G, we deduce $u < u' < c_{x'} < c_x$ in L^G. It follows that the edges uc_x and $u'c_{x'}$ are nested. However, note that both edges are contained in T_y and have the same color i. This is a contradiction to the fact that we colored these edges according to the queue layout obtained by the induction hypothesis. This concludes the proof of the claim. □

To complete the induction step, we have to show that for each $v \in V(G)$, no two edges with v as their right endpoint in L^G are colored with the same color. Suppose for a contradiction that there are distinct edges uv and $u'v$ colored with c such that $u < v$ and $u' < v$ in L^G. By the induction hypothesis we cannot have $c \in \{1, \ldots, t_{k-1}\}$. Therefore, both edges are interbag edges and $c \in \{t_{k-1} + 1, \ldots, 2t_{k-1} + 1\}$. Let $x \in V(T)$ be such that $v \in T_x$. Then u and u' are vertices of the clique C_x. Since c_x is the unique vertex of C_x that is connected by an edge in color $2t_{k-1}+1$ to v, we deduce $c \neq 2t_{k-1}+1$. However, then our coloring rule for the edges uv and $u'v$ implies that the edges uc_x and $u'c_x$ are colored with $c - t_{k-1} \in \{1, \ldots, t_{k-1}\}$. As c_x is the rightmost vertex of C_x with respect to L^G, we obtain that the intrabag edges uc_x, $u'c_x$ have the same color and the same right endpoint in L^G, which is a contradiction to the induction hypothesis. We conclude that any two edges with the same right endpoint in L^G are colored with different colors.

Finally, since we use $2t_{k-1} + 1 = 2(2^{k-1} - 1) + 1 = 2^k - 1$ queues in our layout of G, this completes the proof of the theorem. $\square$

4 Lower bounds – Proof of Theorem 4

This section is devoted to a proof of Theorem 4. We start by introducing a two-player game between Alice and Bob on k-trees (where $k \geqslant 2$), in which Bob has to build a queue-layout of the k-tree to be presented by Alice. We call it the *k-queue game.*

The game starts with a $(k+1)$-clique and an arbitrary linear order of the vertices of this clique. Now, each round of the game consists of two moves. First, Alice introduces a new vertex v and chooses a k-clique of the current graph to which v becomes adjacent. And second, Bob has to specify the position in the current layout where v is inserted. Clearly, since we start with a $(k+1)$-clique, the graphs obtained during the k-queue game remain k-trees. It is the goal of Alice to increase the maximum size of a rainbow in the layout, while Bob tries to keep it small. Alice *wins* the k-queue game if Bob creates a rainbow of size $k + 1$ in the layout. We aim to show the following.

Lemma 8. *For each $k \geqslant 1$, there is an integer d_k such that Alice has a strategy to win the k-queue game within at most d_k rounds.*

Before we prove this lemma, we use it to show Theorem 4. Let us make some new definitions first.

Given a graph H and a clique C in H, we *stack* on C in H by introducing a new vertex v_C and by making v_C adjacent to the vertices of C. (Note that if we stack on a k-clique of a k-tree, then the resulting graph is also a k-tree.) If a graph H' is obtained by simultaneously stacking on each k-clique of H, then we call H' the *k-stack* of H.

We now iteratively construct a family of k-trees $(G_i)_{i \in \mathbb{N}}$. Let G_0 be a $(k+1)$-clique, and given $i \geqslant 1$, we define G_i to be the k-stack of G_{i-1}. Note that with this definition G_i contains G_{i-1} as an induced subgraph. In fact, G_i might contain several distinct induced subgraphs isomorphic to G_{i-1}. For us it is important that G_i contains an *intrinsic* copy G'_{i-1} of G_{i-1} as an induced subgraph, which is such that G_i can be obtained by taking the k-stack of G'_{i-1}.

The following lemma implies Theorem 4.

Lemma 9. *Given $k \geqslant 2$, let d_k be as in the statement of Lemma 8. Then the queue-number of the k-tree G_{d_k} is at least $k+1$.*

Proof. Consider the following variant of the k-queue game. Alice's move in a round of the variant consists of simultaneously stacking on each possible k-clique. It is then Bob's task in this round to insert all the newly introduced vertices in the current layout. Again, Alice wins the game when a rainbow of size $k+1$ appears in the layout.

Clearly, for Bob this variant is harder than the k-queue game, in the sense that if Alice has a strategy to win the k-queue game within d rounds, then she also has a strategy to win the variant within d rounds. In particular, Lemma 8 also holds for the variant.

Now suppose for a contradiction that there is a linear order L of the vertices of G_{d_k} such that there is no rainbow of size $k+1$ in L. We claim that Bob can use L as an instruction to avoid rainbows of size $k+1$ during the first d_k rounds in the variant of the k-queue game.

To see this, observe that after i rounds of the variant, the graph built by Alice is isomorphic to G_i. This gives rise to a strategy for Bob. He only has to fix induced subgraphs $H_0, H_1, \ldots, H_{d_k}$ of G_{d_k} such that $H_{d_k} = G_{d_k}$ and such that H_{i-1} is the intrinsic copy of G_{i-1} in H_i for each $i \in \{1, \ldots, d_k\}$. (Note that H_i is isomorphic to G_i). Then $L|_{V(H_i)}$ is an extension of $L|_{V(H_{i-1})}$ for each $i \in \{1, \ldots, d_k\}$. Therefore, Bob can ensure that the linear order after i rounds is equal to $L|_{V(H_i)}$. Indeed, he only has to read from L how to extend the layout in each round. Applying this strategy, the linear order built after d_k rounds is equal to L. As L does not contain a rainbow of size $k+1$, Bob can prevent Alice from winning within the first d_k rounds. This is a contradiction to Lemma 8 and completes the proof. □

The rest of this section is devoted to a proof of Lemma 8. We proceed with some definitions that will help us to talk about the k-queue game.

Let G be a k-tree designed by Alice during the game and let L be the linear order of $V(G)$ built by Bob. Given $x, y \in V(G)$, we say that x *lies left of* y in L, if $x < y$ in L, and we say that x *lies right of* y in L, otherwise. We denote the leftmost and the rightmost vertex of a subgraph H of G with respect to L by $\ell(H)$ and $r(H)$, respectively. An edge e of G *covers* a subgraph H of G in L if $\ell(e) \leqslant \ell(H) < r(H) \leqslant r(e)$ in L. The edge e *strictly covers* H if we have $\ell(e) < \ell(H)$ and $r(H) < r(e)$ in L. Suppose Alice chooses to stack on the clique C in her next move. Then we say that Bob *goes inside* C if he places the new vertex v_C such that $\ell(C) < v_C < r(C)$ in the layout. Otherwise, we say that Bob *goes outside* C. If Bob places v_C such that $r(C) < v_C$ in the layout, then he *goes to the right outside* of C.

We continue by developing a strategy for Alice to win the k-queue game within a finite number of rounds. Whenever we write that Alice *can force* Bob to make certain moves, then we mean that she has a strategy to win the game unless Bob does these moves.

Lemma 10. *For any k-clique C in the graph built by Alice and any positive number d, Alice can force Bob to go outside some k-clique C^*, which is covered by the edge $\ell(C)r(C)$, for at least d times.*

Proof. We describe a strategy for Alice to enforce the claimed behavior of Bob. First, Alice starts to stack on the clique C in her moves. If Bob does not go inside C for d rounds, then C fulfills the desired requirements.

So suppose that Bob goes inside C with the vertex v_C so that $\ell(C) < v_C < r(C)$ in L. Note that the vertices in the set $V(C) \setminus \{\ell(C)\} \cup \{v_C\}$ form a k-clique C'. For the next rounds, Alice keeps on stacking on C'. Again, if Bob does not go inside C' for d rounds, then we are done with clique C'. So suppose that he goes inside C' with the vertex $v_{C'}$. Then the vertices in $V(C') \setminus \{r(C)\} \cup \{v_{C'}\}$ form a k-clique C'' that is strictly covered by the edge $\ell(C)r(C)$.

Now observe that if Alice applies the above strategy to C'' instead of C, and Bob keeps on avoiding to go outside k-cliques as before, then a k-clique will be strictly covered by the edge $\ell(C'')r(C'')$ after several rounds. Clearly, if Alice repeats this strategy, then a rainbow of size $k+1$ appears in the layout unless Bob goes outside some k-clique being covered by $\ell(C)r(C)$ for at least d times, as claimed. ☐

Lemma 11. *Let C be a k-clique in the graph built by Alice consisting of vertices $v_1, \dots, v_k$ such that $v_1 < \dots < v_k$ in the layout. Assume that Alice can force Bob to go to the right outside of C at least $2k+1$ many times. Then Alice can enforce the existence of a vertex v_{k+1} in the layout such that*

1. *$v_k < v_{k+1}$ in the layout,*
2. *v_{k+1} is adjacent to C, and*
3. *Alice can force Bob to go to the right outside of C' arbitrary many times, where C' denotes the k-clique on the vertices $v_1, v_3, \dots, v_{k+1}$.*

Proof. By assumption, Alice can force Bob to place $2k+1$ vertices $p_1, \dots, p_k, v_{k+1}, q_1, \dots, q_k$, which are adjacent to C, to the right of C in the layout. Let us suppose that

$$p_k < \dots < p_1 < v_{k+1} < q_k < \dots < q_1$$

in the layout. For each $i \in \{1, \dots, k\}$, we let $e_i := p_i v_i$ and $e'_i := q_i v_i$. Observe that the edges $e_1, \dots, e_k$ and $e'_1, \dots, e'_k$ form rainbows of size k.

We claim that v_{k+1} fulfills the requirements of the statement. Clearly, v_{k+1} lies to the right of v_k in the layout and it is adjacent to C, so 1 and 2 hold. Denote the k-clique on the vertices $v_1, v_3, \dots, v_{k+1}$ by C'. Even stronger than condition 3, we can show that whenever Alice introduces a vertex $v_{C'}$ being adjacent to C', then Bob loses unless he puts $v_{C'}$ to the right of C' (that is, to the right of $v_{k+1} = r(C')$).

So suppose that Bob places $v_{C'}$ to the left of v_{k+1}. Then let $j \in \{1, \dots, k+1\}$ be minimal such that $v_{C'} < v_j$ in the layout (see Figure 2 illustrating an example with $k = 4$ and $j = 3$). We obtain that the edges $e'_1, \dots, e'_{j-1}, v_{C'}v_{k+1}, e_j, \dots, e_k$ form a rainbow of size $k+1$ in the layout (in Figure 2 this is the rainbow consisting of red edges), implying that Bob lost the game. Therefore, condition 3 holds. ☐

Later, we will show that Alice can reach a winning configuration in the k-queue game by using the previous lemma. This configuration is described in the following lemma.

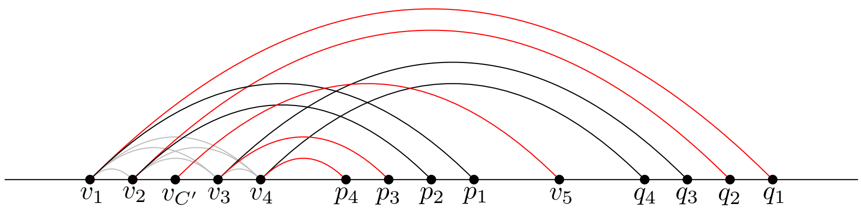


Figure 2: Situation in the 4-queue game (not all existing edges are depicted). If Bob places $v_{C'}$ to the left of v_5, then this creates a 5-rainbow.

Lemma 12. *Suppose that there are edges e, e', e'', and a k-clique C in the graph built by Alice such that*

$$\ell(e) \leqslant \ell(e') < r(e') < \ell(C) < r(C) < \ell(e'') < r(e'') \leqslant r(e)$$

in the layout built by Bob (see Figure 3 for an illustration of such a situation). Then Alice has a strategy to win the current k-queue game within a finite number of rounds.

Proof. Given the configuration of the statement, we describe a strategy for Alice to win the game. Alice starts by applying the strategy of Lemma 10 to enforce a k-clique C' being covered by the edge $\ell(C)r(C)$, such that Bob is forced to go outside C'. Note that C' and the edges e, e', e'' also build a configuration as described in the statement of the lemma. So we may assume that C is already the clique on which Bob is forced to go outside. In the following, let $v_1, \dots, v_k$ be the vertices of C such that $v_1 < \cdots < v_k$ in the layout.

Next, Alice keeps on stacking on C until there are $2k-1$ vertices adjacent to C that all lie to the left of C, or that all lie to the right of C (as Bob has to go outside C, this happens after at most $4k-3$ rounds). By symmetry, we may assume that these $2k-1$ vertices lie left of $\ell(C)$. Using the pigeonhole principle we obtain that either there are k such vertices lying to the left of $\ell(e)$, or k such vertices lying between $\ell(e)$ and $\ell(C)$.

Consider the first case. So we have k vertices $p_1, \dots, p_k$ adjacent to C such that

$$p_k < \cdots < p_1 < \ell(e) \leqslant \ell(e') < r(e') < v_1 < \cdots < v_k$$

in the layout. Then the edges $e', p_1v_1, \dots, p_kv_k$ form a rainbow of size $k+1$, and hence Alice wins the game.

In the second case, Bob has placed k vertices $p_1, \dots, p_k$ being adjacent to C such that

$$\ell(e) < p_k < \cdots < p_1 < v_1 < \cdots < v_k < r(e)$$

in the layout. However, in this case the edges $p_1v_1, \dots, p_kv_k, e$ form a rainbow of size $k+1$.

This shows that Alice has a winning strategy once the configuration in the statement of the lemma occurs during the game. □

We are now ready to combine the previous lemmas to prove Lemma 8.

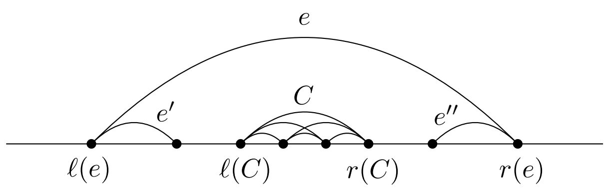


Figure 3: Winning configuration for Alice.

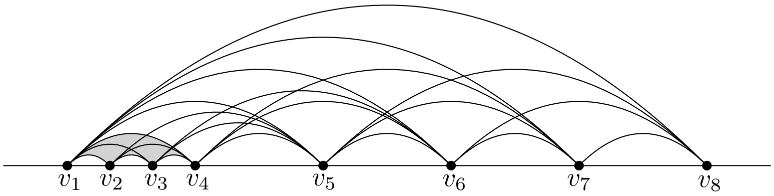


Figure 4: Situation in the 4-queue game after applying Lemma 11 four times (starting with the 4-clique on v_1, v_2, v_3, v_4). It contains the winning configuration of Figure 3 that is formed by the edges v_1v_8, v_1v_2, v_7v_8 and the clique on v_3, v_4, v_5, v_6.

Proof of Lemma 8. We describe a strategy for Alice to win the k-queue game. Using a k-clique of the initial graph in the game and the strategy of Lemma 10, Alice can enforce a k-clique C_1 on which Bob has to go outside arbitrary many times. Next, Alice keeps on stacking on C_1 until Bob has placed $2k+1$ of the newly introduced vertices either to the left of C_1, or to the right of C_1. By symmetry, we may assume that the latter occurs.

Observe that C_1 fulfills the assumptions of Lemma 11. Starting with C_1, we now describe how Alice can iteratively apply the strategy of this lemma. Let $v_1, \dots, v_k$ be the vertices of C_1 such that $v_1 < \cdots < v_k$ in the layout. Then by Lemma 11 Alice can enforce a vertex v_{k+1} to the right of v_k, such that v_{k+1} is adjacent to C_1 and Bob is forced to go to the right outside of the k-clique C_2 consisting of the vertices $v_1, v_3, \dots, v_{k+1}$.

Clearly, Alice can now apply the strategy of Lemma 11 to C_2. So suppose that Alice goes on like this for another three times starting with C_2, and denote the three newly enforced vertices by v_{k+2}, v_{k+3}, and v_{k+4}. Then we have $v_1 < \cdots < v_{k+4}$ in the layout, and with their introduction the new vertices became adjacent to the following vertices: vertex v_{k+2} to $v_1, v_3, \dots, v_{k+1}$, vertex v_{k+3} to $v_1, v_4, \dots, v_{k+2}$, and vertex v_{k+4} to $v_1, v_5, \dots, v_{k+3}$. Figure 4 shows this situation for $k = 4$.

Next we show that the resulting layout contains the winning configuration of Lemma 12. To see this, let $e := v_1v_{k+4}$, $e' := v_1v_2$, and $e'' := v_{k+3}v_{k+4}$. Now note that e, e', e'' and the k-clique formed by the vertices $v_3, \dots, v_{k+2}$ build such a winning configuration.

Therefore, Alice can apply the strategy of Lemma 12 and wins the k-queue game. By the arguments used for the proofs of Lemmas 10–12, it is also clear that Alice can exploit her winning strategy within a number of rounds that only depends on k. This completes the proof. □

5 Open Problems

In this paper we showed a single exponential upper bound on the queue-number of graphs with tree-width at most k. It remains open whether this bound can be reduced to a bound that is polynomial in k. Regarding our theorem on the lower bound, it seems unlikely that $k+1$ is the right answer for the maximal queue-number of k-trees. A quadratic lower bound would already be an exciting improvement.

As mentioned in the introduction, it remains open whether planar graphs have bounded queue-number. The current best upper bound of $\mathcal{O}(\log n)$ is due to Dujmovic [7]. From below we showed the existence of planar graphs with queue-number at least 3. This is a surprisingly large gap for such a popular class of graphs.

Concerning the track-number, the analogous upper bound problems are unsolved as well.

Acknowledgments

We are very grateful to Piotr Micek for the many discussions on the topic. Moreover, we would like to thank an anonymous referee for their numerous helpful remarks.

References

[1] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k- trees. *Discrete Appl. Math.*, 23(1):11–24, 1989.

[2] Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *J. Comb. Theory, Ser. B*, 27:320–331, 1979.

[3] Prosenjit Bose, Jurek Czyzowicz, Pat Morin, and David R. Wood. The maximum number of edges in a three-dimensional grid-drawing. *J. Graph Algorithms Appl.*, 8(1):21–26, 2004.

[4] Robert F. Cohen, Peter Eades, Tao Lin, and Frank Ruskey. Three-dimensional graph drawing. *Algorithmica*, 17(2):199–208, 1997.

[5] Giuseppe Di Battista, Fabrizio Frati, and János Pach. On the queue number of planar graphs. *SIAM J. Comput.*, 42(6):2243–2285, 2013.

[6] Emilio Di Giacomo, Giuseppe Liotta, and Henk Meijer. Computing straight-line 3D grid drawings of graphs in linear volume. *Comput. Geom.*, 32(1):26–58, 2005.

[7] Vida Dujmović. Graph layouts via layered separators. *J. Combin. Theory Ser. B*, 110:79–89, 2015.

[8] Vida Dujmović, Pat Morin, and David R. Wood. Layout of graphs with bounded tree-width. *SIAM J. Comput.*, 34(3):553–579, 2005.

[9] Vida Dujmović and David R. Wood. On linear layouts of graphs. *Discrete Math. Theor. Comput. Sci.*, 6(2):339–357, 2004.

[10] Vida Dujmović and David R. Wood. Three-dimensional grid drawings with subquadratic volume. In *Graph drawing*, volume 2912 of *Lecture Notes in Comput. Sci.*, pages 190–201. Springer, Berlin, 2004.

[11] Vida Dujmović and David R. Wood. Stacks, queues and tracks: layouts of graph subdivisions. *Discrete Math. Theor. Comput. Sci.*, 7(1):155–201, 2005.

[12] Vida Dujmović and David R. Wood. Upward three-dimensional grid drawings of graphs. *Order*, 23(1):1–20, 2006.

[13] Vida Dujmović, Pat Morin, and David R. Wood. Layered separators in minor-closed families with applications, 2013. arXiv:1306.1595.

[14] Stefan Felsner, Giuseppe Liotta, and Stephen Wismath. Straight-line drawings on restricted integer grids in two and three dimensions. *J. Graph Algorithms Appl.*, 7(4):363–398 (electronic), 2003.

[15] Joseph L. Ganley and Lenwood S. Heath. The pagenumber of k-trees is $O(k)$. *Discrete Appl. Math.*, 109(3):215–221, 2001.

[16] Petr Gregor, Riste Škrekovski, and Vida Vukašinović. Queue layouts of hypercubes. *SIAM J. Discrete Math.*, 26(1):77–88, 2012.

[17] Rudolf Halin. Tree-partitions of infinite graphs. *Discrete Math.*, 97(1-3):203–217, 1991.

[18] Lenwood S. Heath, Frank Thomson Leighton, and Arnold L. Rosenberg. Comparing queues and stacks as mechanisms for laying out graphs. *SIAM J. Discrete Math.*, 5(3):398–412, 1992.

[19] Lenwood S. Heath and Arnold L. Rosenberg. Laying out graphs using queues. *SIAM J. Comput.*, 21(5):927–958, 1992.

[20] Jaroslav Nešetřil, Patrice Ossona de Mendez, and David R. Wood. Characterisations and examples of graph classes with bounded expansion. *European J. Combin.*, 33(3):350–373, 2012.

[21] János Pach, Torsten Thiele, and Géza Tóth. Three-dimensional grid drawings of graphs. In *Advances in discrete and computational geometry (South Hadley, MA, 1996)*, volume 223 of *Contemp. Math.*, pages 251–255. Amer. Math. Soc., Providence, RI, 1999.

[22] Sriram Venkata Pemmaraju. *Exploring the powers of stacks and queues via graph layouts*. ProQuest LLC, Ann Arbor, MI, 1992. Thesis (Ph.D.)–Virginia Polytechnic Institute and State University.

[23] Bruce A. Reed. Algorithmic aspects of tree width. In *Recent advances in algorithms and combinatorics*, pages 85–107. New York, NY: Springer, 2003.

[24] S. Rengarajan and C. E. Veni Madhavan. Stack and queue number of 2-trees. In *Computing and combinatorics (Xi'an, 1995)*, volume 959 of *Lecture Notes in Comput. Sci.*, pages 203–212. Springer, Berlin, 1995.

[25] Detlef Seese. Tree-partite graphs and the complexity of algorithms. Fundamentals of computation theory, Proc. 5th Int. Conf., Cottbus/Ger. 1985, Lect. Notes Comput. Sci. 199, 412-421 (1985), 1985.

[26] David R. Wood. Queue layouts, tree-width, and three-dimensional graph drawing. In *FST TCS 2002: Foundations of software technology and theoretical computer science (Kanpur)*, volume 2556 of *Lecture Notes in Comput. Sci.*, pages 348–359. Springer, Berlin, 2002.

[27] David R. Wood. Bounded-degree graphs have arbitrarily large queue-number. *Discrete Math. Theor. Comput. Sci.*, 10(1):27–34, 2008.

[28] Mihalis Yannakakis. Embedding planar graphs in four pages. *J. Comput. System Sci.*, 38(1):36–67, 1989. 18th Annual ACM Symposium on Theory of Computing (Berkeley, CA, 1986).