Characterization of Unlabeled Level Planar Trees

Alejandro Estrella-Balderrama*, J. Joseph Fowler**, Stephen G. Kobourov**

Department of Computer Science, University of Arizona
{aestrell,jfowler,kobourov}@cs.arizona.edu

Technical Report 06-03
August 28, 2006

Abstract. Consider a graph G drawn in the plane so that each vertex lies on a distinct horizontal line $\ell_j = \{(x, j) \mid x \in \mathbb{R}\}$. The bijection ϕ that maps the set of n vertices V to a set of distinct horizontal lines ℓ_j for $j \in \{1, 2, \ldots, n\}$ forms a *labeling* of the vertices. The graph G with the labeling ϕ is called an *n-level graph* and is said to be *n-level planar* if it can be drawn with straight-line edges and no crossings while keeping each vertex on its own level. In this paper, we consider the class of trees that are n-level planar regardless of their labeling. We call such trees *unlabeled level planar* (ULP). Our contributions are three-fold. First, we provide a complete characterization of ULP trees in terms of a pair of forbidden subtrees. Second, we show how to draw ULP trees in linear time. Third, we provide a linear time recognition algorithm for ULP trees.

1 Introduction

When drawing an n-vertex planar graph $G(V, E)$ in the xy-plane, a more restrictive form of planarity can be obtained by insisting on a predetermined y-coordinate for each vertex. In particular, suppose we have a set of k equidistant horizontal lines or *levels*, namely $\ell_j = \{(x, j) \mid x \in \mathbb{R}\}$ for $j \in \{1, 2, \ldots, k\}$ and each vertex is assigned to one of these k levels. Call this level assignment ϕ. The tuple $G(V, E, \phi)$ forms a *k-level graph*, and if ϕ is bijective so that each vertex is constrained to its own level, i.e., $k = n$, then $G(V, E, \phi)$ is an *n-level graph*. Further, suppose that when drawing G, each edge is a straight-line segment (or a continuous y-monotone polyline). If a planar drawing of G can be obtained in spite of these restrictions, then G is said to be *level planar* for level assignment ϕ. If G is an n-level graph that is level planar, then we say G is *n-level planar*.

In fact, if $k < n$, then it is NP-hard [13] to determine whether there even exists a k-level assignment of G in which G is level planar. If $k = n$, in which G is an n-level graph, such a level assignment gives a *labeling* of V since each vertex in V is uniquely numbered. A labeling of V whose level assignment preserves the planarity of G can be easily obtained from a plane drawing of G and a perturbation of the vertices to ensure unique x-coordinates [3]. One can use the x-ordering of the coordinates of a straight-line grid drawing [4] after perturbing the drawing so that no two vertices lies on the same horizontal line to obtain a labeling of V whose level assignment preserves the planarity of G. So while the question of the existence of a level planar assignment is clearly answered for

* Partially supported by CONACYT-México.
** This work is supported in part by NSF grants CCF-0545743 and ACR-0222920.

$k = n$, the existence of a *level non-planar assignment* is not. We call a n-level tree that is n-level planar for all possible labelings of its vertices an *Unlabeled Level Planar* (ULP) tree[1]. We characterize ULP trees in terms of a pair of forbidden subtrees and provide linear time recognition and drawing algorithms.

1.1 Background and Motivation

Visualizing hierarchical relationships using automated graph drawing has historically been a strong motivating factor in the study of the planarity of *level graphs*, i.e., graphs with a predetermined level assignment. Many hierarchical models such as those used in social networks [2] utilize either a constant or logarithmic number levels. Even when this is not necessarily the case, a natural underlying emphasis is to minimize the number of levels while preserving planarity whenever possible. For instance, Sugiyama's algorithm [17] for the display of *directed acyclic graphs* (DAG's) does so by favoring the use of short edges over long edges. Hence, given that often there are relatively few levels on which one wants to place vertices, considering levels graphs with $O(|V|)$ levels has not been of practical utility in this area.

However, geometric simultaneous embedding, which is related to geometric thickness [7, 9], has recently led to a new application of n-level graphs [3]. When simultaneously embedding a set of graphs $\mathcal{G}$, each with n vertices labeled by the numbers $\{1, 2, \ldots, n\}$ such that no number is used twice. A simultaneous embedding of $\mathcal{G}$ can be thought of as overlaying a set of transparencies on an overhead projector in which each graph of $\mathcal{G}$ is drawn on its own transparency, or *layer*. Furthermore, all the vertices with the same label from all the layers must coincide when overlaid. Ideally, one would like to draw all the graphs with no bends or edge crossings in any layer. Determining which sets of graphs have geometric simultaneous embeddings has proven difficult. For instance, it is unknown whether a path and a tree can aways be simultaneously embedded in this way for any possible labeling of vertices.

When simultaneously embedding an n-vertex path P with another n-vertex graph G, one can relabel the vertices of P sequentially 1 to n from one endpoint to the other. The corresponding relabeling of the vertices of G gives a natural level assignment ϕ for G. Eades *et al.* [8] have provided an $O(|V|)$ time algorithm for drawing any level planar graph with straight-line segments. Thus, if G is n-level planar for ϕ, it can easily be simultaneously embedded with P since the path merely zig-zags in a y-monotone fashion from one level to the next. The ability to characterize n-level graphs that are n-level planar regardless of the vertex-to-vertex labeling between the pair of graphs, proves useful in determining which obstructions prevent such an n-level planar drawing, which in turn can give additional insight into open problems in simultaneous embedding.

1.2 Previous Work

Jünger *et al.* [15] provide a linear time recognition algorithm for level planar graphs. This is based on the level planarity test given by Heath and Pem-

[1] A more appropriate name for these types of trees might be "unlabeled n-level planar trees" but for simplicity we call them unlabeled level planar or ULP trees.

maraju [11, 12], which in turn extends the more restricted PQ-tree level planarity testing algorithm of *hierarchies*—level graphs of DAGs in which all edges are between adjacent levels and all the source vertices are on the uppermost level—given by Di Battista and Nardelli [5]. Hierarchies are characterized in terms of level non-planar (LNP) patterns in [5] as well. Jünger and Leipert [14] further provide a linear time level planar embedding algorithm that outputs a set of linear orderings in the x-direction for the vertices on each level. However, to obtain a straight-line planar drawing one needs to subsequently run a $O(|V|)$ algorithm given by Eades *et al.* [8] who demonstrate that every level planar embedding has a straight-line drawing, though it may require exponential area. If the number of levels is constant, then Dujmović *et al.* [6] also provide a linear time level planarity testing algorithm using fixed parameter tractability instead of PQ-trees. Healy *et al.* [10] use LNP patterns to provide a set of minimum level non-planar subgraph patterns that characterize level planar graphs. These subgraph patterns are somewhat analogous to Kuratowski's result that any minimal non-planar graph is either a subdivided K_5 or $K_{3,3}$ [16]. It should be noted that these patterns are specific to a given level assignment and are not based solely on the underlying graph.

1.3 Our Contribution

Our contributions are three-fold.

1. First, we provide a forbidden subdivision characterization for unlabeled level planar (ULP) trees in terms of two minimal ULP trees, T_1 and T_2; see Fig. 1.

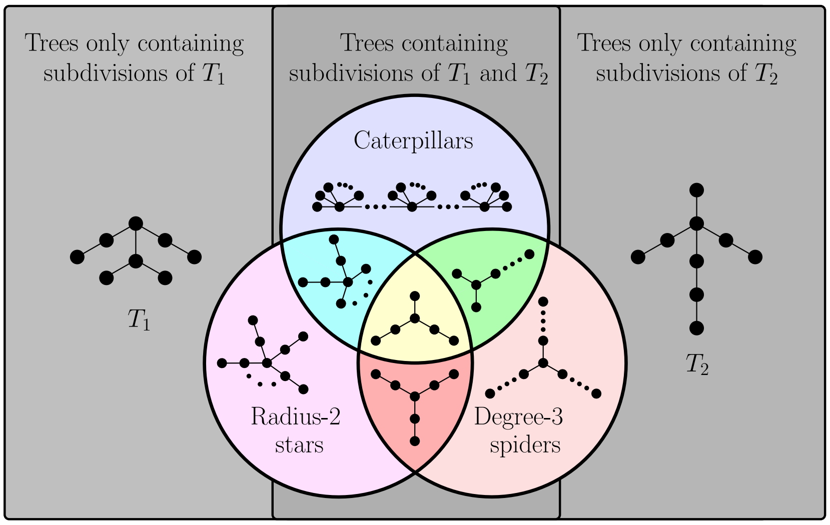

Fig. 1. A Venn diagram for the universe of trees characterized by the two forbidden subtrees T_1 and T_2. Graphs that do not contain either a subdivision of T_1 or T_2 are caterpillars, radius-2 stars, and degree-3 spiders. These three categories comprise the universe of unlabeled level planar trees, which are depicted in the three overlapping circles. Examples of trees that falling into multiple categories are illustrated in the intersections of the circles.

2. Second, we characterize any tree without a subdivision of either T_1 or T_2 as either (i) a *caterpillar*, a tree in which the removal of all degree-1 vertices yields a path, which forms its *spine*, or (ii) a *radius*-2 *star*, a $K_{1,k}$ in which every edge is subdivided at most once, or (iii) a *degree*-3 *spider*, an arbitrary subdivision of $K_{1,3}$. Examples of each of these are also shown in Fig. 1. We show that these three classes are n-level planar by virtue of an $O(|V|)$ time algorithm for constructing straight-line n-level planar drawings of ULP trees.
3. Third, if a tree is not a caterpillar, then it must contain a *lobster* (a graph in which the removal of all degree-1 vertices yields a caterpillar). Using minimal lobsters we show that trees that are not radius-2 stars or degree-3 spiders must contain subdivisions of T_1 or T_2, which completes the characterization. We also provide a $O(|V|)$ time algorithm for testing whether a tree falls into one of these three categories, thus yielding a linear time recognition algorithm for ULP trees.

2 Preliminaries

In this paper we try to use the established notation for level graphs whenever possible. The following definitions for levels graphs are predominantly taken from [1, 5, 10]. A *k-level graph* $G(V, E, \phi)$ on n vertices is a DAG with a level assignment $\phi : V \rightarrow [1..k]$ such that the induced partial order is strict: $\phi(u) < \phi(v)$ for every $(u, v) \in E$. A k-level graph is a k-partite graph in which ϕ partitions V into k independent sets $V_1, V_2, \ldots, V_k$, which form the k *levels* of G. A *level-j* vertex v is on the j^{th} *level* V_j of G if $\phi(v) = j$ where $V_j = \phi^{-1}(j)$.

If ϕ is an injection, each level contains at most one vertex, i.e., $|V_i| \leq 1$ for $i \in [1..k]$, hence, $k \geq n$. W.l.o.g., we can assume in such instances that $k = n$ in which case ϕ is a bijection that forms a topological sort of the DAG $G(V, E)$. Unless noted otherwise, an n-level graph $G(V, E, \phi)$ is assumed to have a bijective level assignment ϕ, i.e., $\phi : V \xrightarrow[\text{onto}]{1:1} [1..n]$. Such a bijective level assignment is equivalent to a *labeling* of the vertices from 1 to n.

A level graph G has a *level drawing* if there exists a drawing such that every vertex in V_j is placed along the horizontal line $\ell_j = \{(x, j) \,|\, x \in \mathbb{R}\}$ and the edges are drawn as strictly y-monotone polylines. The order that the vertices of V_j are placed along each ℓ_j in a level drawing induces a family of linear orders $(\leq_j)_{1 \leq j \leq k}$ along the x-direction, which form a *linear embedding* of G. A level drawing, and consequently its level embedding, is *level planar* if it can be drawn without edge crossings. A level graph G is level planar if it admits a level planar embedding. The more restrictive definition of level drawings allowing only straight-line segments for edges is equivalent, as shown by Eades *et al.* [8] by demonstrating that every level planar graph has a straight-line planar drawing.

A planar graph H is *realize*d if it can be drawn with straight-line edges without crossings. Such a plane graph is a *realization* of H. A n-level graph $G(V, E, \phi)$ is *n-level realize*d if it is *realize*d such that each vertex v lies on its level $\phi(v)$, which is a certificate of the n-level planarity of G.

A *chain* of a k-level graph $G(V, E, \phi)$ is a nonrepeating sequence of vertices $v_1, v_2, \ldots, v_t$ of V such that $t > 1$ and either $(v_i, v_{i+1}) \in E$ or $(v_{i+1}, v_i) \in E$ for every $i \in \{1, 2, \ldots, t-1\}$, i.e., a path in the underlying undirected graph of G. If C is a chain, let $\textsc{lower}(C)$ and $\textsc{upper}(C)$ denote the lowermost and uppermost vertices, respectively, with respect to ϕ. Also let $\textsc{min}(C) = \phi\big(\textsc{lower}(C)\big)$ and $\textsc{max}(C) = \phi\big(\textsc{upper}(C)\big)$ be the minimum and maximum levels of C, thus, $\textsc{min}(C) \leq \phi(v) \leq \textsc{max}(C)$ for every v of C.

For an n-level graph $G(V, E, \phi)$, let $<_Y$ denote the strict linear ordering given by the level assignment ϕ, i.e., for every $u, v \in V$, we have $u <_Y v$ iff $\phi(u) < \phi(v)$. Let $<_X$ (and $\leq_X$) denote the strict (and weak) linear ordering induced by the x-coordinate of the placement of a level-i vertex u and a level-j vertex v along their respective horizontal lines ℓ_i and ℓ_j in the particular level drawing under consideration. Finally, for vertex subsets $U, W \subseteq V$, let $U <_X W$ (and $U <_Y W$) iff $u <_X w$ (and $u <_Y w$) for every $u \in U$ and $w \in W$. Often we will represent an edge $(u, v) \in E$ as $u-v$ and a chain of vertices, $v_1, v_2, \ldots, v_t$ for some $t > 1$ as $v_1-v_2- \cdots -v_t$.

Finally, we recall a few standard graph theory definitions. In a graph $G(V, E)$, *subdividing* an edge $(u, v) \in E$ is the operation of replacing (u, v) with the pair of edges (u, w) and (w, v) in E by adding w to V. A *subdivision* of G is a graph obtained by performing a series of successive edge subdivisions of G.

3 Characterization of Unlabeled Level Planar Trees

First, we introduce the forbidden subdivisions T_1 and T_2 together with explicit level assignments in which the resulting graphs are level non-planar. Then we show how to compute a n-level realization for each of the three remaining types of trees—caterpillars, radius-2 stars and degree-3 spiders—in linear time given a labeling of the vertices. Next, we show that if a tree does not contain a subdivision of T_1 or T_2, then it must fall into at least one of the three categories of unlabeled level planar trees. Finally, we give a simple $O(|V|)$ time recognition algorithm for ULP trees.

3.1 Forbidden Trees

Lemma 1 *There exist labelings that prevent T_1 and T_2 from being level planar.*

Proof. Fig. 2(a) gives one of 8 distinct labelings that satisfies the y-partial order $\{c, g\} >_Y d >_Y f >_Y a >_Y b >_Y \{e, h\}$ (or its dual in which the ordering is reversed). Each labeling gives a bijective n-level assignment in which T_1 does not have a n-level realization. This can be seen as follows: To prevent paths $a-b-c$ and $a-d-e$ from crossing, one path must go to the left and the other to the right. Assume w.l.o.g. $c-b <_X d-e$. This forces $c <_X f <_X d$ so that $a-f$ does not cross $b-c$, or $a-f-g$ does not cross $a-d-e$. However, then $f-h$ will cross either $a-d$ or $a-b-c$. This concludes the argument for T_1.

Next, consider T_2. Fig. 2(b) gives one of 8 distinct labelings that satisfies the y-partial order $\{c, h\} >_Y d >_Y f >_Y a >_Y b >_Y g >_Y \{e, i\}$ (or its dual in which the ordering is reversed). Each labeling gives a bijective n-level assignment in which T_2 does not have a n-level realization. This can be seen as follows: To prevent paths $a-b-c$ and $a-d-e$ from crossing, one path must go to the left

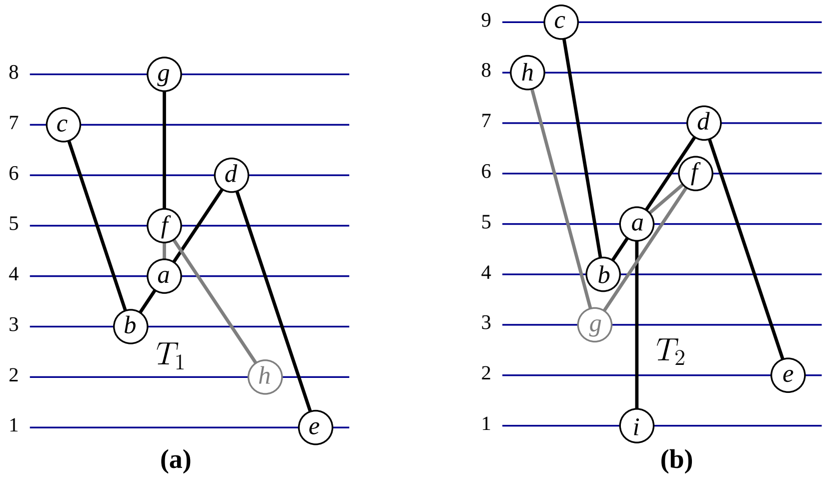

Fig. 2. Labelings that prevent T_1 and T_2 from being ULP.

and the other to the right. Assume again w.l.o.g. $c–b <_X d–e$. Then $a–i$ must be drawn below and to the right of $a–b$ and to the left of $d–e$, otherwise it will cross $b–c$ or $d–e$. To prevent the edge $a–f$ from crossing $a–b–c$ or $a–d–e$, f must with be drawn (i) so that $a <_X f <_X e$ in which case $f–g–h$ will then cross $a–i$ or $d–e$, or (ii) so that $c <_X f <_X d$ in which case $f–g$ will cross $a–b–c$, $a–d$, or $a–i$. This completes the argument about T_2 and the overall claim. □

As a consequence, we have the following corollary.

Corollary 2 *If a tree $T(V, E)$ contains a subdivision of T_1 or T_2, then it cannot be unlabeled level planar.*

Proof. Assume that the tree T contains a subdivision of T_1 (or T_2). Let $T'(V', E')$ be a subtree of T that is a subdivision of T_1 (or T_2). Label the 8 (or 9) vertices of V' in the same order as shown in Fig. 2. Note that the values of the labels need to be adjusted in order to accommodate any intermediate vertices along a subdivided edge of T_1 (or T_2). Any extra vertex w along a subdivided edge (u, v) can be assigned to a unique level that preserves the y-ordering $u <_Y w <_Y v$. This works since level planarity is defined in terms of y-monotone polylines. An edge drawn from level $\phi(u)$ to level $\phi(v)$ is allowed any number of bends so long as the edge proceeds in a y-monotone fashion. In particular, it can bend at w on level $\phi(w)$, which is equivalent to subdividing the edge $u–v$ into $u–w–v$.

This gives a labeling of the vertices of T' using the labels $\{1, 2, \ldots, |V'|\}$ such that the 8 (or 9) vertices corresponding to T_1 (or T_2) satisfy one of the y-partial orders from Lemma 1. Hence, the arguments used in Lemma 1 can be directly applied to this y-ordering. Thus, T' is not n-level planar, and as a consequence, neither is T, regardless of the labels for the remaining vertices. □

3.2 ULP Trees—Caterpillars, Radius-2 Stars, and Degree-3 Spiders

The following three lemmas explicitly show all the trees that are unlabeled level planar and how to n-level realize them in linear time.

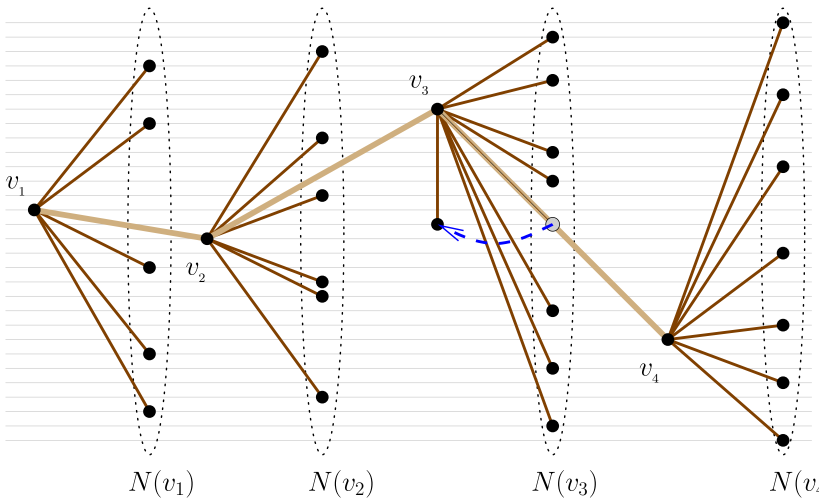

Fig. 3. A n-level realization of a 30-level caterpillar on a 8×30 grid. The eight vertical grid lines are omitted for readability.

Lemma 3 *(Brass* et al. *[3]) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[onto]{1:1} \{1, 2, \ldots, n\}$.*

Proof. The following proof of the claim is a shorter version and is an improvement over the original proof in [5] that spans 3 pages. Note that the original claim had the slightly weaker result of using a $2n \times n$ grid.

Let $T(V, E, \phi)$ be an n-level caterpillar with spine $S(V', E')$ such that S is isomorphic to $P_{|V'|}$. In particular, let the vertices of V' be labeled according to their relative distance from the end point v_1, and the edges $E' = \big\{(v_1, v_2), (v_2, v_3), \ldots, (v_{|V'|-1}, v_{|V'|})\big\}$. Let the degree-1 leaves of v be denoted by $N(v) = \big\{u \,|\, (u, v) \in E \text{ and } (u, v) \notin E'\big\}$ for each $v \in V'$. Then for each $i \in \{1, 2, \ldots, n\}$ place $v_i \in V'$ at the coordinate $\big(2i - 1, \phi(v_i)\big)$ and place each $u \in N(v_i)$ at the coordinate $\big(2i, \phi(u)\big)$ unless u would lie on the straight-line edge segment v_i–v_{i+1} in which case place u directly under v_i at the coordinate $\big(2i - 1, \phi(u)\big)$ instead. All this can be done in $O(n)$ time. This drawing is an n-level planar since S is drawn in a strictly left to right fashion and each incident edge to the spine is either drawn either directly above or below the spine or immediately to its right. Clearly, this drawing uses only straight-line edge segments in which there are no crossings forming a n-level realization. □

Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n + 3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[onto]{1:1} \{1, 2, \ldots, n\}$.*

Proof. Let r be the root of an n-level radius-2 star $T(V, E, \phi)$ located at the coordinate $\big(n+2, \phi(r)\big)$. For every leaf ℓ that is at a distance of 1 from r, place it at the coordinate $\big(n+1, \phi(\ell)\big)$, which is one x-coordinate to the left of r. For each remaining leaf ℓ, let $adj(\ell)$ denote its adjacent vertex, and let $L \subseteq V$ denote this set of leaves at a distance 2 from r. Then $L_d = \big\{\ell \,|\, \ell \in L \text{ and } \phi(adj(\ell)) > \phi(\ell)\big\}$

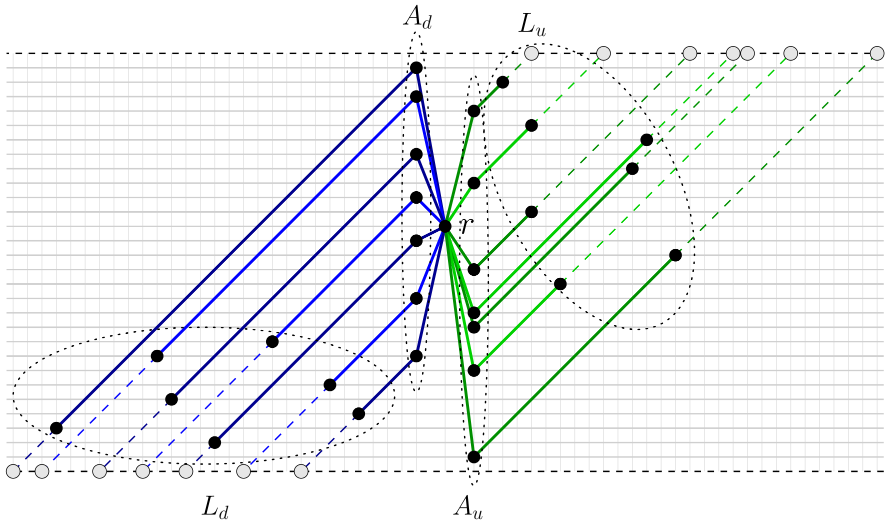

Fig. 4. A n-level realization of an 29-level radius-2 star on a 61×29 grid. The small gray circles are the intersection points of slope-1 rays emanating from each vertex in A_d and A_u to the imagined 0-level and imagined $(n+1)$-level, respectively, which are drawn with dashed lines.

and $L_u = \big\{\ell \,|\, \ell \in L \text{ and } \phi(adj(\ell)) < \phi(\ell)\big\}$ partition L according to whether the adjacent vertex of the leaf is to be drawn above or below it, i.e., whether the incident edge goes down or up. Let $A_d = \big\{adj(\ell) \,|\, \ell \in L_d\big\}$ and $A_u = \big\{adj(\ell) \,|\, \ell \in L_u\big\}$ be the adjacent vertices of degree 2 to the leaf vertices of T.

Place each $u \in A_d$ at the coordinate $\big(n+1, \phi(u)\big)$ immediately to the left of r, and each $u \in A_u$ at the coordinate $\big(n+3, \phi(u)\big)$, immediately to the right of r. For each $\ell \in L_d$, place it at the grid point that corresponds to the intersection of the $\phi(\ell)$-level and the line segment connecting the points $\big(n+1, \phi(adj(\ell))\big)$, the coordinate of its adjacent vertex, and $\big(n - \phi(adj(\ell)) + 1, 0\big)$, the point that an emanating ray from $adj(\ell)$ with a slope of 1 in the *negative* x-direction intersects an imagined 0-level; see Fig. 4. Since the ray has slope 1, this intersection will always be an integer grid point. In a similar fashion, place each $\ell \in L_u$ at the grid point that corresponds to the intersection of the $\phi(\ell)$-level and the line segment connecting the points $\big(n+3, \phi(adj(\ell))\big)$, the coordinate of its adjacent vertex, and $\big(n - \phi(adj(\ell)) + 1, n+1\big)$, the point that an emanating ray from $adj(\ell)$ with a slope of 1 in the *positive* x-direction intersects an imagined $(n+1)$-level. All this can be done in linear time since $O(1)$ time is spent locating each vertex.

This produces a n-level realization since every vertex that is adjacent to r is either placed immediately to its right or left, and every other leaf is placed so that its incident edge has a slope of 1, which prevents any edge from crossing. $\square$

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[onto]{1:1} \{1, 2, \ldots, n\}$.*

Proof. We start by noting that unlike in the case of a caterpillar or a radius-2 star, a degree-3 spider can require exponential area for a n-level realization. We omit details regarding area for this abstract.

The proof is in three parts. First, we show how to reduce an arbitrary degree-3 spider to one in which all the legs zig-zag between successively lower and higher levels. Second, we skip ahead and show how to place the vertices of the original spider when processing an edge of the reduced spider. Finally, we show an $O(n)$ time algorithm for greedily drawing the reduced degree-3 spider.

Part 1: Let r be the root vertex of an n-level degree-3 spider $T(V, E, \phi)$. Let X, Y, and Z be the three subtrees of r, each of which forms a chain. First, we show how to reduce T to a special form in which X, Y, and Z each zig-zag back and forth between successively lower and higher levels. Call $T'(V', E', \phi')$ a *strictly expanding* degree-3 spider if the level assignment ϕ' on the vertices $r, v_2, \ldots, v_{|C|}$ of each chain C of T' obeys the following two properties:

$$\phi(v_{i-1}) < \phi(v_i) > \phi(v_{i+1}) \ \text{ or } \phi(v_{i-1}) > \phi(v_i) < \phi(v_{i+1}), \tag{1}$$

and

$$\begin{aligned} &\left[\phi(v_{i-1}) < \phi(v_i) \Rightarrow \phi(v_{i-1}) > \phi(v_{i+1})\right] \ \text{ and} \\ &\left[\phi(v_{i-1}) > \phi(v_i) \Rightarrow \phi(v_{i-1}) < \phi(v_{i+1})\right] \end{aligned} \tag{2}$$

for $1 < i < |C|$. We call a chain that satisfies property (1) a *zig-zagging chain* since it cannot have any monotonically increasing or decreasing sequences of vertices. We note that if a portion of a chain were to be monotonic, then the locations of the vertices of that portion could be trivially computed. Hence, this property is easily satisfied by extracting a subchain from T, namely the vertices in which the level assignment changes direction, which are precisely the vertices of T for which (1) holds. Assuming that we only have zig-zags for chains, then a zig-zagging chain that also satisfies property (2) is *strictly expanding* as the next level reached by the chain is either greater than any previous level or less than any previous level.

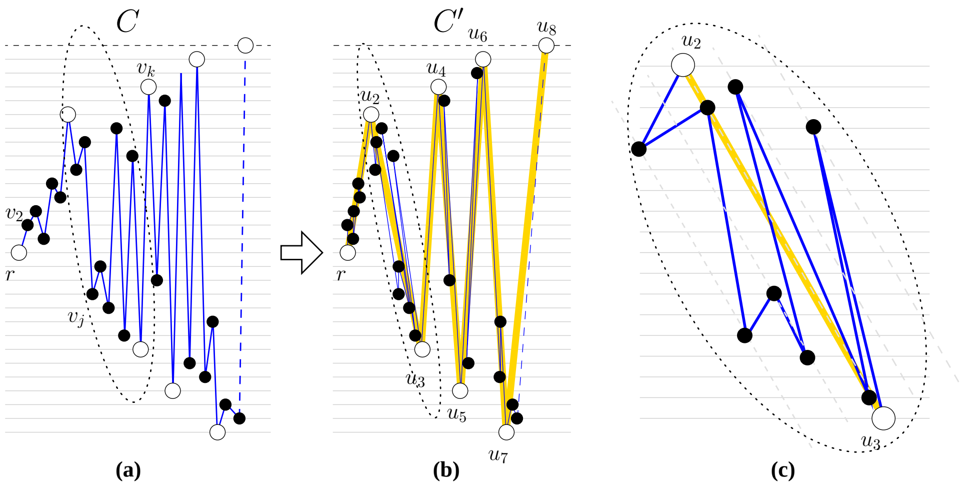

Fig. 5. An example of a chain C of (a) in which the strictly expanding zig-zagging subchain C' of white vertices is extracted to give (b). Then (c) shows an example n-level realization of the intermediate edges and vertices for the second edge of (b).

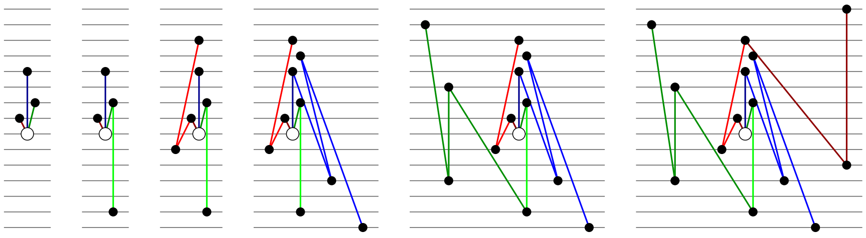

Fig. 6. Six iterations of the greedy strategy to n-level realize a degree-3 spider.

A zig-zagging chain C can be made strictly expanding by keeping track of the minimum and maximum levels encountered by the chain so far. Assume w.l.o.g. that $\phi(r) < \phi(v_2)$, i.e., the chain C begins by going upwards. We extract from C, the strictly expanding subchain C', which we will label its vertices as $r - u_2 - u_3 - \cdots - u_{C'}$, by first prepending r to C'. Then we set $min_C = \phi(r)$ and find the first vertex v_j along C such that $\phi(v_j) < min_C$. Next we append $u_2 = \text{UPPER}(r - v_2 - v_3 - \cdots - v_{j-1})$ to C', and set $max_C = \phi(u_2)$. Then we look for the next vertex v_k along C such that $\phi(v_k) > max_C$. Afterward, we append $u_3 = \text{LOWER}(v_j - v_{j+1} - \cdots - v_{k-1})$ to C', and set $min_C = \phi(u_3)$, and repeat this process until all the vertices of C are exhausted. If the last vertex encountered is not greater than min_C or less than max_C, then we add an extra vertex to the end of C' satisfying this condition; see Fig. 5 for an illustration of this process.

Part 2: For each vertex u_i in C', we keep a linked list of the subpath $P = u_i - w_{i_1} - w_{i_2} - \cdots - w_{i_{|P|-2}} - u_{i+1}$ of C that was replaced by the edge $u_i - u_{i+1}$ where $\text{MIN}(P) = \phi(u_i)$ and $\text{MAX}(P) = \phi(u_{i+1})$ (or $\text{MAX}(P) = \phi(u_i)$ and $\text{MIN}(P) = \phi(u_{i+1})$). This linked list will be used to place edges of T as we process edges from T'. Fig. 5(c) visually illustrates how this might be done. Here, any particular subpath P of C for a given edge of C' can be drawn arbitrary close to C. We omit the details of this particular point, noting only that the intuitive idea of compressing the zig-zagging chain allows us to greedily draw the edges of T without crossings for each edge of T' that is processed. We finish this section of the proof by observing that both the extraction of C' from C *and* the ability to draw the edges of T once we have processed the edges of T' can be done in linear time.

Part 3: Now that we have our degree-3 spider in the proper form, we can apply a simple greedy algorithm that can be used to give a n-level realization of T'. We complete the proof of the lemma by giving the details regarding this linear time algorithm.

Let the vertices of chain X be denoted by $x_0 - x_1 - \cdots - x_{|X|-1}$ in which $x_0 = r$ and $(x_i, x_{i+1}) \in E$ for $0 \leq i < |X| - 2$. Similarly, let the vertices of Y and Z be $y_0 - y_1 - \cdots - y_{|Y|-1}$ and $z_0 - z_1 - \cdots - z_{|Z|-1}$. Finally, let $A_1 = \{x_1, y_1, z_1\}$ be the first vertices along each chain immediately following r.

There exist two possibilities for the strictly expanding degree-3 spider T': Either (i) $A_1 >_Y r$ (or $A_1 <_Y r$) or (ii) $\text{MIN}(A_1) <_Y r <_Y \text{MAX}(A_1)$. We show

how to draw T for case (i) assuming that $A_1 >_Y r$. The other case is similar. We start the first iteration by drawing A_1 so that the vertex of maximum index with respect to ϕ' lies between the other two vertices of A_1 along the x-coordinate.

At any one point in this greedy strategy, we maintain the invariant that the last vertex along a chain that we placed either lies above or below any of the other vertices that have been drawn so far. Property (2) allows us to do this. If we encounter the end of a chain in which this invariant does not hold for its last vertex, then we can easily draw the remaining two chains without crossings since the invariant holds for their most recently placed vertex. We do this by drawing one of the two chains monotonically to the right until we reach its end, and do the same for the other chain monotonically to the left.

For iteration $i > 1$, we arbitrarily pick one of the chains whose most recently placed vertex is neither the maximum nor the minimum vertex drawn so far. We greedily extend the chain either to the right or left until we reach a vertex whose level assignment is either above or below all the ones drawn so far. This enlarges the set of processed vertices from A_{i-1} to A_i.

Note that we can always extend a chain C to the right or left. This follows from the fact that during the previous iteration, *before the vertices of some other chain C' were processed*, the last vertex v of C was either minimum or maximum. Suppose w.l.o.g. v was the maximum vertex processed so far until a vertex v' of C' becomes the new maximum vertex without any of the previous vertices of C' having level assignments less than the minimum vertex v'' of the other chain C''. As a result, we temporarily suspend drawing C'. Suppose further w.l.o.g. that the edge incident to v' was drawn to the right of v. This allows us to continue extending C to the left. In fact, this must be the next course of action since the third chain C'' is not eligible to be drawn at this time. This is because v'' must still be the minimum vertex processed so far. If at any point a previous vertex of C' were to have a level assignment less than v'', then we would have stopped drawing C' at that point, and continued drawing C'' in which case C would not be eligible to be drawn. This is because v would still be the maximum vertex processed so far. See Fig. 6 for an example of this greedy algorithm.

Since we can always greedily place a vertex without introducing a crossing, this strategy succeeds in producing a n-level realization of T in $O(n)$ time (constant time per vertex), which shows that T is indeed ULP. Simple geometry can be used to construct such an drawing using only straight-line edge segments for T', which can be used to produce a plane drawing of T as detailed above, though, as we initially noted, this can require exponential area. $\square$

Now that we have shown which trees are ULP, we need to show that our characterization is complete. First, we show that T_1 and T_2 are minimal unlabeled level non-planar trees with the following lemma.

Lemma 6 *Removing any edge from T_1 or T_2 yields a forest of ULP trees.*

Proof. If removing an edge from T_1 decreases the degree of one of the two degree-3 vertices, call them x and y, then the resulting graph is a forest consisting of a degree-3 spider and a possible lone edge; see Fig. 7(a). Removing the edge $x{-}y$ yields two paths. The only possibility (up to isomorphism) in removing

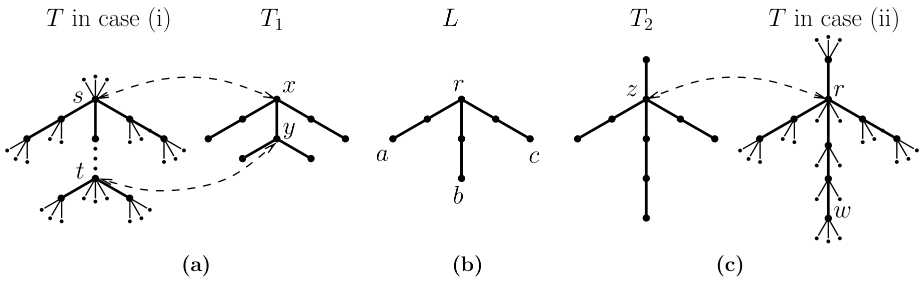

Fig. 7. Finding copies of T_1 and T_2 in trees that are not caterpillars, radius-2 stars, or degree-3 spiders.

an edge without affecting the degree of x and y, yields a caterpillar with a spine of length 5. Moving onto T_2, if its vertex z of degree 4 maintains its degree after the edge removal, then the resulting graph must be a forest consisting of either a caterpillar, if the removed edge was incident to a leaf vertex at a distance 2 from z, or a radius-2 star and a possible lone edge, otherwise. On the other hand, if the degree of z decreases to 3, then the resulting graph is a degree-3 spider and, possibly, a path; see Fig. 7(c). □

The next theorem completes the characterization of ULP trees.

Theorem 7 *Every tree either contains a subdivision of T_1 or T_2 in which case it is not ULP, or it is a caterpillar, a radius-2 star, or a degree-3 spider in which case it is ULP. Hence, T_1 and T_2 give a minimal forbidden subtree characterization of ULP trees.*

Proof. First, we argue that neither T_1 nor T_2 is a caterpillar, a radius-2 star, or a degree-3 spider to show that if a tree T contains a subdivision of T_1 or T_2, it cannot be one of those three. Clearly, both T_1 and T_2 are lobsters and not caterpillars since removing all their degree-1 vertices yields caterpillars rather than paths. The two vertices of degree 3 prevent T_1 being a radius-2 star or a degree-3 spider. Since T_2 has radius 3 and a vertex of degree 4, it cannot be a radius-2 star or a degree-3 spider either.

Now we show that caterpillars, radius-2 stars, and degree-3 spiders are the only types of ULP trees. We do this by showing that any tree that does not fit into at least one of these categories must contain either a subdivision of T_1 or T_2. Then by Corollary 2 any tree T that contains a subdivision of T_1 or T_2 cannot be ULP. By Lemma 6, T_1 and T_2 are minimal.

Assume that $T(V, E)$ is a tree that is not a caterpillar, radius-2 star, or degree-3 spider. Since T is not a caterpillar, it must contain a minimal lobster L, i.e., the unique tree that cannot have any more edges removed without becoming a caterpillar (and possibly a lone edge); see Fig. 7(b). It has one vertex r of degree 3 and three leaf vertices a, b, c at a distance 2 from r, which is the minimal requirement for a tree to be a lobster. Any other lobster can have its edges trimmed away until L is all that remains, which is what makes L minimal.

Since T is not a degree-3 spider, there are two cases to consider: either (i) T has two vertices s and t of degree at least 3 or (ii) T has one vertex of degree k greater than 3.

Assuming case (i) holds, we show how to find a subdivision of T_1 in T. Let x and y be the two vertices of degree 3 in T_1 where x is the one without an adjacent leaf vertex; see Fig. 7(a). At least one of the two vertices s and t of degree at least 3 in T must correspond to the root vertex r in the subtree L that forms the minimal lobster in T. Assume w.l.o.g. this vertex is s. Then we map s in T to vertex x in T_1, and the other vertex of degree at least 3, t in T to vertex y in T_1. Since t has degree at least 3, there exists two neighbors of t not along the path from s to t, which we can map to the two corresponding leaf vertices in T_1 that are adjacent to y. Only one of the three leaf vertices a, b, c of L in T can be contained in the subtree of s containing t. Suppose w.l.o.g. it is a. Then the other two vertices b and c in T can be mapped to the two remaining leaf vertices in T_1. This completes the mapping of vertices of T_1, showing that T contains T_1 as a subdivision where the only edge in T_1 that is subdivided is $x{-}y$ that maps to the path s to t in T.

Next we consider case (ii) in which we show how to find the subtree T_2 in T. The one vertex of degree k greater than 3 must be the corresponding vertex r of L in T; see Fig. 7(c). Otherwise, if there were separate vertices of degree greater than 3, case (i) would apply. Let r be mapped to the degree-4 vertex z of T_2. Since T is not a radius-2 star, there exists a vertex w at a distance 3 from r, which can be mapped to the leaf vertex in T_2 at a distance 3 from z. Only one of the three vertices a, b, c of L in T can be along the path from r to w. Suppose w.l.o.g. it is a. The other two vertices b and c in T can be mapped to the other two leaf vertices in T_2. The remaining leaf vertex of T_2 that is directly adjacent to z can be mapped to the endpoint of the fourth edge incident to r in T since it has degree greater than 3. This completes the mapping of vertices of T_2, showing that T contains T_2 as a subtree. □

3.3 Linear Time Recognition of ULP Trees

First, we need a few simple observations regarding the degree sequences of caterpillars, radius-2 stars, and degree-3 spiders, which we state as lemmas whose proofs we omit in this abstract.

Lemma 8 *If a tree T has a degree sequence of the form $2, \dots, 2, 1, 1$ or $1, 1$, i.e., a path, after the removal of all degree-1 vertices, then T must be a caterpillar.*

Lemma 9 *If a tree T has a degree sequence of the form $k, 2, \dots, 2, 1, 1, \dots 1$ for some $k > 2$, i.e., T is an arbitrarily subdivided $K_{1,k}$, and after the removal of all degree-1 vertices, the degree sequence then becomes $\ell, 1, \dots, 1$ for some $\ell \leq k$, i.e., T becomes a $K_{1,\ell}$, then T must be a radius-2 star.*

Lemma 10 *If a tree T has a degree sequence of the form $3, 2, \dots, 2, 1, 1, \dots 1$, i.e., T has maximum degree of 3 with only one vertex of degree 3, then T must be a degree-3 spider.*

Theorem 7 together with the above Lemmas, lead to a simple linear time recognition algorithm for ULP trees summarized in the following corollary:

Corollary 11 *The class of* ULP *trees can be recognized in linear time. That is, given an arbitrary n-vertex tree T, one can decide in $O(n)$ time whether or not it is always possible to n-level realize T for any possible labeling.*

Proof. In order to determine whether a tree $T(V,E)$ is ULP, we merely need to determine whether it is a caterpillar, radius-2 star, or degree-3 spider by Theorem 7. We do this using degree sequences. We note that it is possible to obtain a sorted degree sequence of a graph given an adjacency matrix representation since it is merely the size of each vertex list. The sorting of the degree sequence can be done in linear time using counting sort since there are at most n positive integer values that lie strictly in the range of 1 and $n-1$.

If the degree sequence of T is of the form $3, 2, \ldots, 2, 1, \ldots, 1$, then by Lemma 10, T must be a degree-3 spider. Else we do a single pass through all the doubly-connected adjacency lists and remove all degree-1 vertices, which can be done in $O(n)$ time by using a companion adjacency matrix with pointers into the adjacency lists. If the resulting graph is empty, then T is a path, i.e., a caterpillar. Otherwise, if its degree sequence is of the form $1, 1$ or $2, \ldots, 2, 1, 1$, then by Lemma 8, T must be a caterpillar. Otherwise, if degree sequence is of the form $k, 1, \ldots, 1$, then T must be a radius-2 star by Lemma 9.

If none of these is the case, then T cannot be one of these three types of trees. Since we can make these evaluations of degree sequences in $O(n)$ time, this algorithm only takes linear time to determine whether a tree is ULP. □

4 Conclusion and Future Work

We described a complete characterization of unlabeled level planar trees. We provided a linear time algorithm to n-level realize the three classes of ULP trees which can also be used for simultaneously embedding a ULP tree T with any path P. Clearly, an n-level graph G can be simultaneously embedded with a path P, provided the level assignment of G corresponding to the labeling of the vertices of P yields a n-level realization. Since this is always the case for ULP trees, have also given a mechanism for simultaneously embedding a ULP tree T with any path P.

Finally, we provided a linear time recognition algorithm for ULP trees. What is missing from the recognition algorithm is a certificate of unlabeled level non-planarity, i.e., the 8 (or 9) vertices corresponding T_1 (or T_2) if they exist.

We can easily find some set of vertices corresponding to a copy of T_1 or T_2 in an unlabeled level non-planar tree T in $O(n^2)$ time running depth-first search from each vertex of degree at least 3 determining if it either corresponds to one of two vertices of degree 3 in T_1 or the vertex of degree 4 in T_2. Developing a more efficient, perhaps linear time, version of this algorithm is left for future work. Even so, the copy of T_1 or T_2 found in this way may not be the offending one that prevents T from being n-level non-planar for a given level assignment ϕ, which is more desirable.

In this case, we can construct an $O(n^2)$ time algorithm to produce such a certificate. We do this by using the linear time recognition algorithm for level planar graphs of Jünger, Leipert, and Mutzel (JLM) [15]. For every edge e of a n-level non-planar tree T, try removing it to form T'. If T' is n-level planar, then leave e in T, otherwise delete it from T, and repeat this process until all the edges of T have been considered. By Corollary 2, the resulting tree T'' that is still n-level non-planar, must contain a subdivision of T_1 or T_2. Furthermore, it can only contain one such subdivision, since any edge that would have only contributed to one, but not the other, would have been removed. Finally, we can process T'' in $O(n^2)$ time attempting to "unsubdivide" each pair of edges incident to vertices of degree 2. We process these vertices from the ones with least to the ones with greatest level assignment. Since there are at most $O(n)$ such pairs of edges, and we can test if the resulting tree is still level non-planar in $O(n)$ time, this algorithm takes $O(n^2)$ time altogether. This process is guaranteed to yield vertices in T corresponding to either a copy of T_1 or T_2 with a n-level non-planar level assignment.

The problem with this approach aside from the running time is that it depends on the complicated JLM algorithm. Giving an algorithm that can directly yield a violating certificate given a particular level assignment in which T is n-level non-planar would be preferable. This remains to be done and is also left for future work.

Another future task is to provide a forbidden subgraph characterization for general unlabeled level planar graphs as we have done for ULP trees.

References

1. C. Bachmaier, F. J. Brandenburg, and M. Forster. Radial level planarity testing and embedding in linear time. *J. Graph Algorithms Appl.*, 9(1):53–97, 2005.
2. U. Brandes, P. Kenis, and D. Wagner. Communicating centrality in policy network drawings. *IEEE Transactions on Visualization and Computer Graphics*, 09(2):241–253, 2003.
3. P. Brass, E. Cenek, C. A. Duncan, A. Efrat, C. Erten, D. Ismailescu, S. G. Kobourov, A. Lubiw, and J. S. B. Mitchell. On simultaneous graph embedding. In *8th Workshop on Algorithms and Data Structures*, pages 243–255, 2003.
4. H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.
5. G. Di Battista and E. Nardelli. Hierarchies and planarity theory. *IEEE Trans. Systems Man Cybernet.*, 18(6):1035–1046 (1989), 1988.
6. V. Dujmovic, M. R. Fellows, M. T. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. A. Rosamond, M. Suderman, S. Whitesides, and D. R. Wood. On the parameterized complexity of layered graph drawing. In *European Symposium on Algorithms*, pages 488–499, 2001.
7. C. Duncan, D. Eppstein, and S. G. Kobourov. The geometric thickness of low degree graphs. In *20th ACM Symposium on Computational Geometry*, pages 340–346, 2004.
8. P. Eades, Q. Feng, X. Lin, and H. Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. *Algorithmica*, 44(1):1–32, 2006.
9. D. Eppstein. Separating thickness from geometric thickness. In *Proceedings of the 10th Symposium on Graph Drawing (GD)*, pages 150–161, 2002.
10. P. Healy, A. Kuusik, and S. Leipert. A characterization of level planar graphs. *Discrete Math.*, 280(1-3):51–63, 2004.

11. L. S. Heath and S. V. Pemmaraju. Recognizing leveled-planar dags in linear time. In *4th Symposium on Graph Drawing (GD)*, pages 300–311. 1996.
12. L. S. Heath and S. V. Pemmaraju. Stack and queue layouts of directed acyclic graphs. II. *SIAM J. Comput.*, 28(5):1588–1626, 1999.
13. L. S. Heath and A. L. Rosenberg. Laying out graphs using queues. *SIAM J. Comput.*, 21(5):927–958, 1992.
14. M. Jünger and S. Leipert. Level planar embedding in linear time. *J. Graph Algorithms Appl.*, 6:no. 1, 67–113, 2002.
15. M. Jünger, S. Leipert, and P. Mutzel. Level planarity testing in linear time. In *6th Symposium on Graph Drawing (GD)*, pages 224–237. 1998.
16. C. Kuratowski. Sur les problèmes des courbes gauches en Topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.
17. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems Man Cybernet.*, 11(2):109–125, 1981.