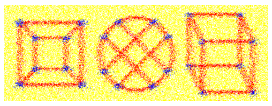

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 14, no. 2, pp. 245–267 (2010)

Minimum-Layer Upward Drawings of Trees

Muhammad Jawaherul Alam[1] *Md. Abul Hassan Samee*[2]
Mashfiqui Rabbi[3] *Md. Saidur Rahman*[1]

[1]Graph Drawing and Information Visualization Laboratory, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka-1000, Bangladesh
[2]Department of Computer Science, University of Illinois at Urbana-Champagne, Urbana, IL 61801-2302, USA
[3]Department of Computer Science, Dartmouth College, Hanover, NH 03755, USA

Abstract

An upward drawing of a rooted tree T is a planar straight-line drawing of T where the vertices of T are placed on a set of horizontal lines, called *layers*, such that for each vertex u of T, no child of u is placed on a layer vertically above the layer on which u has been placed. In this paper we give a linear-time algorithm to obtain an upward drawing of a given rooted tree T on the minimum number of layers. Moreover, if the given tree T is not rooted, we can select a vertex r of T in linear time such that an upward drawing of T rooted at r would require the minimum number of layers among all the upward drawings of T with any of its vertices as the root. We also extend our results on a rooted tree to give an algorithm for an upward drawing of a rooted ordered tree. To the best of our knowledge, there is no previous algorithm for obtaining an upward drawing of a tree on the minimum number of layers.

Submitted:	Reviewed:	Revised:	Accepted:
August 2009	January 2010	February 2010	March 2010
	Final: April 2010	Published: June 2010	
Article type: Regular paper		Communicated by: G. Liotta	

E-mail addresses: jawaherul@gmail.com (Muhammad Jawaherul Alam) mahassan@gmail.com (Md. Abul Hassan Samee) mrshuva@dartmouth.edu (Mashfiqui Rabbi) saidurrahman@cse.buet.ac.bd (Md. Saidur Rahman)

1 Introduction

An *upward drawing of a rooted tree* T is a planar straight-line drawing of T where the vertices of T are placed on a set of horizontal lines, called *layers*, such that for each vertex u of T, no child of u is placed on a layer vertically above the layer on which u has been placed. For example, Figures 1(b)–(e) illustrate four different planar straight-line drawings Γ_1, Γ_2, Γ_3 and Γ_4 of the rooted tree T_a of Figure 1(a) with the root vertex a. Among these, Γ_1 is not an upward

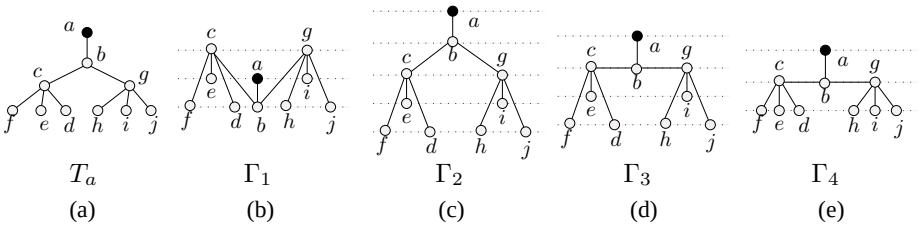

Figure 1: (a) A rooted tree T_a with the root vertex a, (b) a planar straight-line drawing of T_a, which is not upward, (c)–(e) upward drawings of T_a.

drawing of T_a since the vertices c and g are children of the vertex b although they are placed on a layer above the layer on which b is placed in Γ. However Γ_2, Γ_3 and Γ_4 are all upward drawings of T_a. Several works are found on strict upward drawings of G, where each vertex of T is placed on a layer strictly above all of its children [4, 8, 16]. However we are not considering such a strict model in this paper; our definition of upward drawing is consistent with that of [9, 14].

A *minimum-layer upward drawing of a rooted tree* T is an upward drawing of T that occupies the minimum number of layers among all the upward drawings of T. For example, as illustrated in Figure 1, the upward drawings Γ_2, Γ_3 and Γ_4 of T_a occupy five, four and three layers respectively. One can easily verify that there is no upward drawing of T_a on less than three layers. Therefore Γ_4 is a minimum-layer upward drawing of T_a.

An *upward drawing of an unrooted tree* T is an upward drawing of a rooted tree obtained by making any of the vertices of T the root. A *minimum-layer upward drawing of an unrooted tree* T is an upward drawing of T that requires the minimum number of layers among all the upward drawings of T rooted at any of its vertices. For example, Figure 2(a) illustrates an unrooted tree T and we obtain a rooted tree T_a from T with the root vertex a as illustrated in Figure 2(b). Figure 2(c) illustrates an upward drawing of T_a on three layers. Although T does not admit any upward drawing on less than three layers with the root vertex a, if we take the vertex b as the root of T to obtain another rooted tree T_b, T_b admits an upward drawing Γ_b on two layers as illustrated in Figure 2(e). One can also verify that there is no upward drawing of T on less than two layers rooted at any of its vertices. Thus Γ_b is a minimum-layer upward drawing of T.

There are some previously known results that provide upper bounds on the

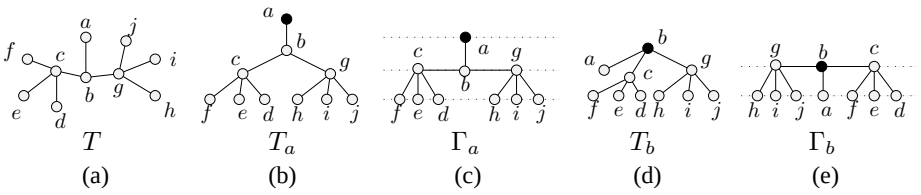

Figure 2: (a) An unrooted tree T, (b) a rooted tree T_a obtained from T with the root vertex a, (c) a minimum-layer upward drawing of T_a, (d) another rooted tree T_b obtained from T with the root vertex b, (e) a minimum-layer upward drawing of T_b and T.

area in upward drawings of trees [8, 9, 14] but none of these algorithms focused on minimizing the number of layers in the drawing. However, there are some papers that addressed this issue for "layered drawings" of trees [15, 17]. A *layered drawing* of a tree T is a planar straight-line drawing of T such that the vertices are drawn on a set of layers. Thus an upward drawing of a rooted tree T is a layered drawing of T with the additional constraint that no vertex u of T is placed on a layer vertically above the layer on which the parent of u is placed. Layered drawings have important applications in VLSI layouts [12], DNA-mapping [18], information visualization [5, 11] etc. In some application areas such as in the "standard cell" technology for VLSI layout design, it is often desirable to draw a tree on the minimum number of layers. However, there is no known algorithm to obtain a layered drawing of a tree on the minimum number of layers. Linear-time algorithms are known [2, 6] for determining whether a tree admits a layered drawing on at most two layers. Felsner *et al.* [6] gave a necessary condition for a tree to admit a layered drawing on k layers for $k > 2$. For a tree T with pathwidth h, Suderman gave a linear-time algorithm to draw T on $\lceil 3h/2 \rceil$ layers but this bound is not tight [15]. Moreover, none of these known algorithms focuses on producing upward drawings of trees although some applications like organization charts, software class hierarchies, phylogenetic evolutions, programming language parsing etc. require upward drawings of rooted trees [7]. In this paper, we give a linear-time algorithm for obtaining a minimum-layer upward drawing of a rooted tree T. In case T is not rooted, our algorithm can select a vertex r of T so that a minimum-layer upward drawing of T rooted at r gives a minimum-layer upward drawing of the unrooted tree T. We also extend the result on a rooted tree to give an algorithm for minimum-layer upward drawing of a rooted ordered tree, where a given left-to-right ordering of the children for each vertex of the tree is preserved in the drawing.

Apart from minimizing the number of layers in an upward drawing of a rooted tree T, our results presented in this paper are also significant regarding the area bounds for upward drawings of T. We have shown that our drawing algorithm produces an upward drawing of T with an area bound of $O(n \log n)$. This bound matches previous ones [3, 7, 13]. Although there is an algorithm which gives an $O(n \log \log n)$ area upward drawings of trees, the algorithm works

only on trees with bounded degrees [14].

The rest of this paper is organized as follows. In Section 2, we give some preliminary definitions and present a brief outline of our algorithm. Section 3 gives a linear-time algorithm for obtaining a minimum-layer upward drawing of a rooted tree. In Section 4 we give our algorithm to obtain a minimum-layer upward drawing of an unrooted tree. In Section 5, we give an algorithm to obtain a minimum-layer upward drawing of a rooted ordered tree. Finally, Section 6 is a conclusion. A preliminary version of this paper has been presented previously [1].

2 Preliminaries

In this section, we give some definitions and present an outline of our algorithm.

Let $G = (V, E)$ be a simple graph with the vertex set V and the edge set E. A subgraph of G is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. Let (u, v) denote an edge of G joining two vertices u and v of G. The edge (u, v) is said to be incident to the two vertices u and v of G. A vertex u of G is a *neighbor* of another vertex v of G (and vice versa) if G has an edge (u, v). The *degree* of a vertex v in G is the number of neighbors of v in G. Let v be a vertex in G. We denote by $G - v$ the graph obtained by deleting from G, the vertex v and all its incident edges in G. A *path* P in G is a sequence v_0, v_1, ..., v_n of vertices of G such that G contains an edge (v_{i-1}, v_i) for each $i, (1 \leq i \leq n)$ and each vertex $v_i, (0 \leq i \leq n)$ is distinct. Such a path P is also called a v_0, v_n-path of G. The vertices v_0 and v_n of the path P are called the *end-vertices* of P.

A graph G is *connected* if there exists a u, v-path in G for every pair of vertices u and v of G. A *component* in a graph G is a maximal connected subgraph of G. A *tree* is a graph which contains exactly one u, v-path for every pair of vertices u and v of G. A tree T is called a *rooted tree* if a vertex r of T is designated as the *root* of T; otherwise, T is called an *unrooted tree*. In this paper, we use the notation T_r to denote the rooted tree obtained from an unrooted tree T by considering a vertex r as the root of T. Let u and v be two vertices of a rooted tree T_r. If the r, v-path in T_r contains u, then u is called an *ancestor* of v and v is called a *descendant* of u. u is the *parent* of v and v is a *child* of u if u immediately precedes v in the r, v-path in T_r. A vertex u of T_r is called a *leaf* of T_r if u has no children in T_r. Otherwise, u is called a *non-leaf vertex* of T_r. A *subtree* of T_r rooted at a vertex u is the rooted tree obtained from the subgraph of T_r induced by the descendants of u such that u is the root. In the remainder of this paper, we use the notation T_u^r to denote the subtree of T_r rooted at u. A rooted tree T_r is called a *rooted ordered tree* if for each vertex u in T_r, there is a fixed left-to-right ordering of the children of u in T. Let T_r be a rooted ordered tree and u be a vertex of T_r. Then the *leftmost* (*rightmost*) child of u in T_r is one that comes first (last) in the left-to-right ordering of the children of u in T_r.

An *upward drawing of a rooted tree* T_r is a planar straight-line drawing of T_r where the vertices of T_r are placed on a set of horizontal lines, called *layers*

and for each vertex u of T_r, no child of u is placed on a layer vertically above the layer on which u has been placed. If T_r is a rooted ordered tree, then in an *upward drawing of* T_r, there is an additional constraint that for each vertex u of T_r, a given left-to-right ordering of the children of u in T_r is preserved in the drawing. A planar straight-line drawing Γ of an unrooted tree T is called *an upward drawing of* T if there is some vertex r in T such that Γ is an upward drawing of T_r. An upward drawing Γ of a rooted or an unrooted tree T is also called *a k-layer upward drawing of* T if the number of layers in Γ is equal to k. A k-layer upward drawing of T is called a *minimum-layer upward drawing of* T if any upward drawing of T requires at least k layers.

We now give an outline of our algorithm for finding a minimum-layer upward drawing of a tree. We first consider the scenario where the given tree is a rooted tree T_r with the root vertex r as illustrated in Figure 3(b). In order to

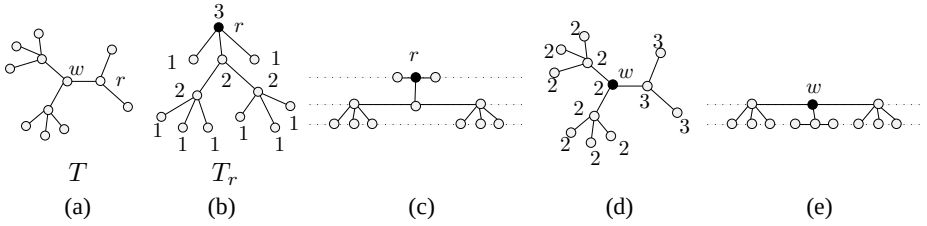

Figure 3: (a) An unrooted tree T and a vertex r of T, (b) line-labelings of all the vertices in T_r, (c) a minimum-layer upward drawing of T_r, (d) line-labelings $L_u(u)$ of u in T_u for all the vertices u of T, and (e) a minimum-layer upward drawing of T.

compute the minimum number of layers required for an upward drawing of T_r, we first perform a bottom-up traversal of T_r during which we label every vertex u of T_r with an integer, called the "line-labeling" of u in T_r, that represents the minimum number of layers required for any upward drawing of T_u^r. (See Figure 3(b)). Let k be the line-labeling of r in T_r. Then k is the minimum number of layers required for any upward drawing of T_r. In the next step, we obtain a k-layer upward drawing of T_r in linear time. (See Figure 3(c)).

We now consider the case where the given tree T is not a rooted tree as illustrated in Figure 3(a). In this case we first select a vertex w of T such that an upward drawing of T_w would require the minimum number of layers among all the upward drawings of T rooted at any vertex of T. For this purpose, we first consider T to be rooted at an arbitrary vertex r to obtain a rooted tree T_r and compute the line-labelings of all the vertices in T_r in the same way as described above. (See Figure 3(b).) After this, we perform a top-down traversal of T_r, during which, for every vertex u of T, we compute the line-labeling of u in T_u. The value of these line-labelings are illustrated in Figure 3(d). Let w be a vertex such that the line-labeling of w in T_w is the minimum among all the line-labelings of u in T_u for all the vertices u of T. We then obtain a minimum-layer upward drawing of T_w as illustrated in Figure 3(e). Our algorithm for

obtaining a minimum-layer upward drawing of a rooted ordered tree is similar to the one for a rooted tree.

3 Upward Drawings of Rooted Trees

In this section, we give an algorithm to obtain a minimum-layer upward drawing of a rooted tree.

Let T_r be a rooted tree with root vertex r and let the height of T_r be h. It is trivial to see that the minimum number of layers required for a strictly upward drawing of T_r is $h + 1$ where the strict upwardness restricts the placement of each vertex of T_r on a layer strictly above all its children. However if we relax the definition of upwardness to allow any child of a vertex v of T_r to be placed on the same layer as v, then an upward drawing of T_r requires fewer number of layers than $h + 1$. We therefore define a new parameter called "line-labeling" to represent the minimum number of layers in an upward drawing of T_r. Before proceeding further, we have the following observations.

Let T_r have three child subtrees such that the minimum number of layers required for each of these subtrees is equal to k as illustrated in Figure 4(a). Then T_r requires at least $k + 1$ layers for any upward drawing since it is not possible to place all the three child subtrees along with the vertex r on the same k layers and add the edges from r to the roots of these subtrees while keeping

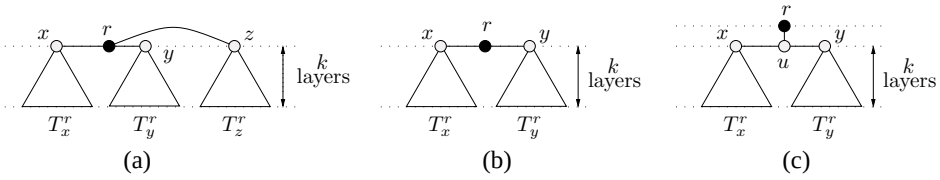

Figure 4: Illustration for the number of layers required for an upward drawing.

planarity. On the other hand if T_r has exactly two subtrees that require at least k layers for any upward drawing, then T_r may have an upward drawing on k layers as illustrated in Figure 4(b). However it is interesting to note that both the left and the right side of r on the topmost layer is occupied in such a drawing. Thus such a rooted tree is in a sense "saturated". Finally consider a rooted tree T_r with a child subtree that is "saturated" on k layers as illustrated in Figure 4(c). Then it is obvious to see that any upward drawing of T_r requires at least $k + 1$ layers. We bring these observations into consideration while we define the notion of "line-labelings of the vertices in a rooted tree T_r" in the following.

Let u be a vertex of T_r. Then the *line-labeling of a vertex u in T_r*, which we have denoted by $L_r(u)$ in the remainder of this paper, is defined as follows.

(a) If u is a leaf of T_r, then $L_r(u) = 1$.

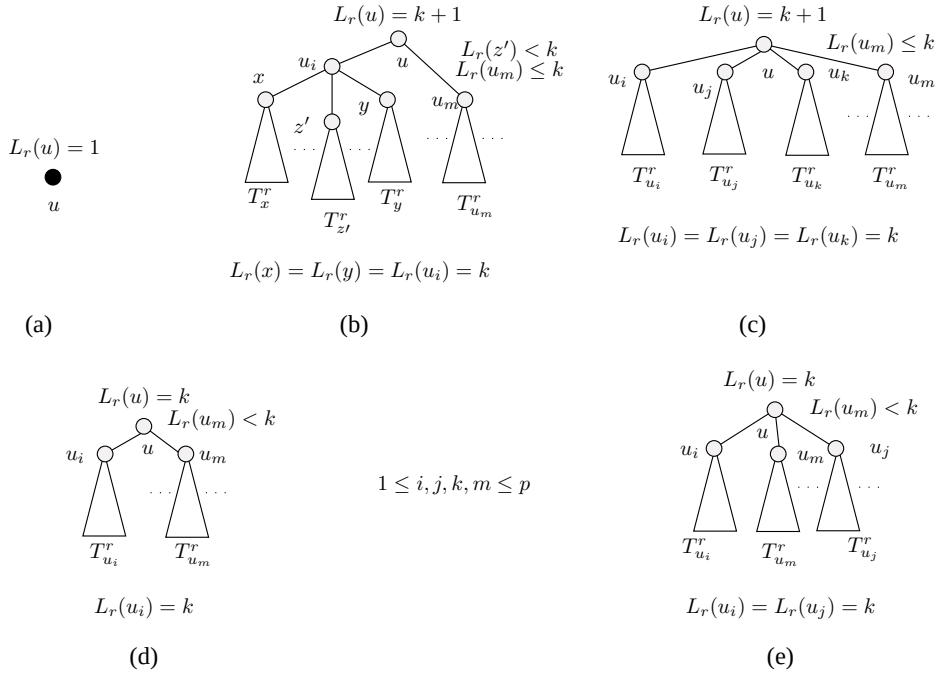

Figure 5: Definition of the line-labeling $L_r(u)$.

(b) If u is a non-leaf vertex of T_r with children u_1, u_2, …, u_p and k is the maximum among the values $L_r(u_1)$, $L_r(u_2)$, …, $L_r(u_p)$, then $L_r(u)$ is defined according to (i)–(iii) as follows.

(i) If u has at least one child u_i in T_r such that $L_r(u_i) = k$ and u_i has at least two children, each of which has line-labeling k in T_r, then $L_r(u) = k+1$. We call a child u_i of u in T_r a *saturated child of* u if u_i has at least two children with the same line-labeling as u_i in T_r.

(ii) If u has no saturated child with line-labeling k in T_r but has three or more children with line-labeling k in T_r, then $L_r(u) = k+1$.

(iii) Otherwise, $L_r(u) = k$. (Note that in this case, u has at most two children with line-labeling k in T_r and neither of them is a saturated child.)

Figure 5 illustrates different cases of this definition. The definition above implies that a saturated child of u with line-labeling k in T_r has exactly two children with line-labeling k in T_r. Furthermore, the following two facts hold directly from the definition above.

Fact 1 *Let T_r be a rooted tree with the root vertex r and let u be a vertex in T_r. Assume that T^r_u is the subtree of T_r rooted at u. Then the line-labeling of u in T_r is the same as the line-labeling of u in T^r_u.*

Fact 2 *Let T_r be a rooted tree with the root vertex r and let u and v be two vertices of T_r such that v is a descendant of u in T_r. Then $L_r(u) \geq L_r(v)$.*

We now have the following lemma, which shows that the line-labeling $L_r(r)$ of the root vertex r in a rooted tree T_r gives a lower bound on the number of layers in an upward drawing of T_r.

Lemma 1 *Let T_r be a rooted tree with the root vertex r. Then any upward drawing of T_r requires at least $L_r(r)$ layers.*

Proof: Let n denote the number of vertices in T_r. We prove this claim by induction on n. The claim is obvious for $n = 1$ since any drawing of a single-vertex tree requires at least one layer as illustrated in Figure 6(a) and $L_r(r) = 1$ by definition. We thus assume that $n > 1$ and the claim is true for any rooted tree with less than n vertices. We now prove the claim for the rooted tree T_r with n vertices.

Let k be the maximum value among the line-labelings of all the children of r in T_r. We first assume that r has a saturated child u in T_r with line-labeling k; that is, $L_r(u) = k$ and u has two children x and y in T_r such that $L_r(x) = L_r(y) = k$. According to the definition, in this case, $L_r(r) = k + 1$ and we claim that any upward drawing of T_r requires at least $k + 1$ layers. Assume for a contradiction that T_r has a k-layer upward drawing Γ. Let Γ_x and Γ_y be the upward drawings of T_x^r and T_y^r respectively, contained in Γ. According to Fact 1, the line-labelings of x and y in T_x^u and T_y^u respectively, are also k. Therefore, from the induction hypothesis, at least k layers are required for any upward drawing of T_x^u and T_y^u. Therefore each of the drawings Γ_x and Γ_y occupies all k layers in Γ. These two drawings restrict the placement of the vertex u in Γ between x and y on the topmost of these k layers of Γ. Then r cannot be placed on any of these k layers in Γ while retaining the upwardness of the drawing, which is a contradiction. (See Figure 6(b).) Hence, at least $k + 1$ layers are required for any upward drawing of T_r.

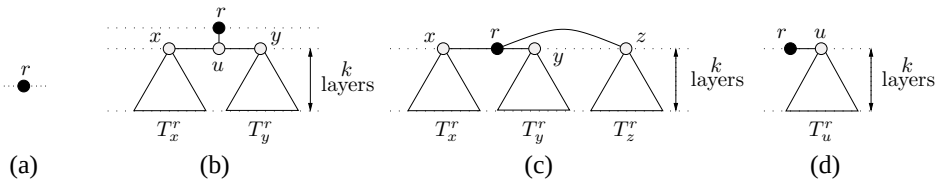

Figure 6: Any upward drawing of T_r requires at least $L_r(r)$ number of layers.

We next assume that r has no saturated child with line-labeling k in T_r. If r has three (or more) children in T_r with line-labeling k in T_r, then $L_r(r) = k+1$ according to the definition and we again claim that any upward drawing of T_r requires at least $k + 1$ layers. Assume for a contradiction that T_r has a k-layer upward drawing Γ. Let x, y and z be three children of u in T_u such that $L_r(x) = L_r(y) = L_r(z) = k$. Let Γ_x, Γ_y and Γ_z be the upward drawings of T_x^r, T_y^r and T_z^r

respectively, contained in Γ. According to Fact 1, the line-labelings of x, y and z in T_x^r, T_y^r and T_z^r respectively, are also k and hence by the induction hypothesis, each of Γ_x, Γ_y and Γ_z occupies all the k layers in Γ. Then as illustrated in Figure 6(c), we cannot place u on any of these k layers and draw the edges from u to x, y and z using straight-line segments and keeping planarity, which is a contradiction. Therefore, at least $k+1$ layers are necessary for any upward drawing of T_r in this case also.

We finally assume that r has at most two children in T_r with line-labeling k in T_r and none of these children of r is a saturated child of r with line-labeling k in T_r. Then according to our definition, $L_r(r) = k$. Let x be a child of r in T_r with line-labeling k in T_r and let T_x^r be the subtree of T_r rooted at x. According to Fact 1, the line-labeling of x in T_x^r is also k and hence by the induction hypothesis, any upward drawing of T_x^r requires at least k layers. Therefore any upward drawing of T_r also requires at least k layers since T_x^r is a subtree of T_r as illustrated in Figure 6(d). □

Lemma 1 implies that any upward drawing of T_r requires at least k layers, where $L_r(r) = k$. We now constructively prove that there is also a k-layer upward drawing of T_r. Before proceeding further, we need to define the notion of "skeleton subgraph".

Let T_r be a rooted tree and let $L_r(r) = k$. Then the *skeleton subgraph* of T_r is defined as the subgraph of T_r induced by the vertices of T_r with line-labeling k in T_r and is denoted by $skel(T_r)$ in the remainder of this paper. The skeleton subgraph of T_r in Figure 8(a) is shown in Figure 8(b). We now have the following lemma.

Lemma 2 *Let T_r be a rooted tree with the root vertex r. Then the skeleton subgraph $skel(T_r)$ of T_r is a path.*

Proof: We first prove the claim that $skel(T_r)$ is a connected subgraph of the tree T_r and hence is a tree. To prove the claim, it is sufficient to show that for each vertex u in the skeleton subgraph $skel(T_r)$, there is a path from r to u in $skel(T_r)$. Let u be a vertex in $skel(T_r)$ with $L_r(r) = k$. Then $L_r(u) = k$ by the definition of the skeleton subgraph. Let w be any vertex on the r,u-path in T_r as illustrated in Figure 7(a). Then w is an ancestor of the vertex u in T_r and by Fact 2, $L_r(w) \geq L_r(u) = k$. Since the root vertex r is the ancestor of all the vertices in T_r, by Fact 2, $L(r) = k \geq L_r(w)$. Therefore $L_r(w) = k$ and hence the vertex w is also in $skel(T_r)$. Since every vertex on the r,u-path in T_r is also in $skel(T_r)$, there is a path from r to u in $skel(T_r)$.

We now prove that the degree of a vertex in the skeleton subgraph $skel(T_r)$ of T_r is at most two. Assume for a contradiction that there is a vertex w with degree at least three in $skel(T_r)$. Let x, y and z be three neighbors of w in $skel(T_r)$. Then, by the definition of skeleton subgraph, $L_r(x) = L_r(y) = L_r(z) = L_r(w) = k$; where $L(r) = k$. If all the three vertices x, y and z are children of w in T_r as illustrated in Figure 7(b), then the line-labeling of at least three children of w in T_r is k and hence the line-labeling of w in T_r is $k+1$, which is a contradiction. We thus assume that w does not have three children

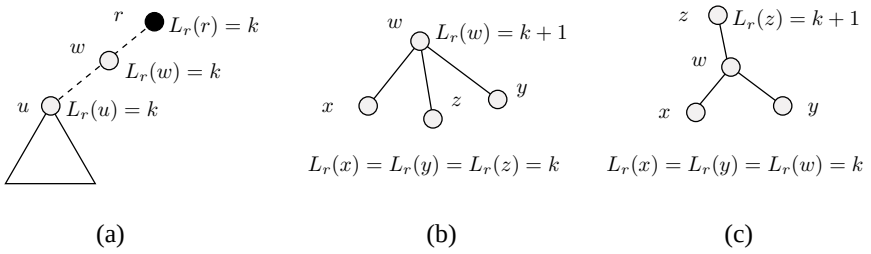

Figure 7: (a) The connectivity of $skel(T_r)$, and (b)–(c) The maximum degree of a vertex in $skel(T_r)$ is two.

with line-labeling k in T_r. Then exactly one of the three vertices x, y and z of w is the parent of w and the other two are children of w in T_r. We may assume that z is the parent of w in T_r as illustrated in Figure 7(c). Then w is a saturated child of z with line-labeling k in T_r and the line-labeling of z in T_r is $k + 1$, which is also a contradiction. Hence, there is no such vertex with degree three or more in $skel(T_r)$ and $skel(T_r)$ is a path. □

We are now ready to prove the following lemma on an upward drawing of the rooted tree T_r.

Lemma 3 *Let T_r be a rooted tree with the root vertex r and let $L_r(r) = k$ be the line-labeling of r in T_r. Then there is a k-layer upward drawing of T_r.*

Proof: The proof is by induction on k. We first assume that $k = 1$. In this case for every vertex u of T_r, $L_r(u) = 1$ and the skeleton subgraph of T_r is the tree T_r itself. By Lemma 2 T_r is a path and hence T_r can be drawn on a single layer. We thus assume that $k > 1$ and there is a k'-layer upward drawing of any rooted tree where the root of the tree has line-labeling $k' < k$ in the tree. We now construct a k-layer upward drawing of T_r.

We first draw the skeleton subgraph $skel(T_r)$ of T_r. By Lemma 2, the skeleton subgraph $skel(T_r)$ is a path. Let us assume that the vertices of the path are $v_1, v_2, \ldots, v_f$ in this order. We draw the skeleton subgraph $skel(T_r)$ on the topmost layer such that the x-coordinate of v_i is one plus the x coordinate of v_{i-1} for each $2 \leq i \leq f$ as illustrated in Figure 8(c). Deleting the skeleton subgraph leaves T_r with several components, each of which is a rooted tree T' with the root having line-labeling less than k in T'. For each of these rooted trees T', there is an edge between the root of T' and exactly one of the vertices in $skel(T_r)$. We call such a tree T' *a v_i-tree* if there is an edge between the root of T' and the vertex v_i in $skel(T_r)$. From the induction hypothesis, each of these trees has a k'-layer upward drawing where $k' < k$. We place the drawings of these trees on the bottommost $k - 1$ layers in such a way that the x-coordinate of the leftmost vertex in the drawing of the leftmost v_i-tree is one plus the x-coordinate of the rightmost vertex in the drawing of the rightmost v_{i-1}-tree for $2 \leq i \leq f$ as illustrated in Figure 8(d). Then, it is possible to join the

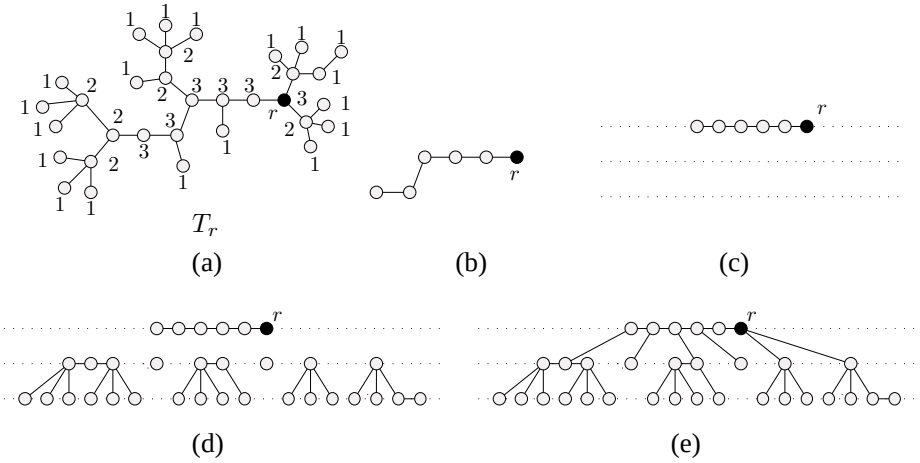

Figure 8: (a) Line-labelings of all the vertices in T_r, (b) the skeleton subgraph $skel(T_r)$ of T_r, and (c)–(e) construction of a k-layer upward drawing T_r.

edges from the vertices of $skel(T_r)$ to the root of these trees without violating planarity as illustrated in Figure 8(e). Thus, we can obtain a k-layer upward drawing of T_r. □

The constructive proof of Lemma 3 immediately gives an algorithm for a k-layer upward drawing of a rooted tree T_r where $L_r(r) = k$. For the remainder of this paper, we call this algorithm **Draw-Rooted**. We now analyze the area requirement of the upward drawing of a rooted tree T_r obtained by Algorithm **Draw-Rooted**. For this purpose, we establish a relationship between the line-labeling of r in T_r and the number of vertices of T_r in the following lemma.

Lemma 4 *Let T_r be a rooted tree with the root vertex r and let the line-labeling of r in T_r be $L_r(r) = k \geq 2$. Then T_r has at least 2^k vertices.*

Proof: The proof is by induction on k. We first assume that $k = 2$. The claim is true in this case, namely, T_r has at least $2^2 = 4$ vertices, since if T_r had three or less number of vertices, then T_r would have been a path and the line-labeling of r in T_r would be one, which is a contradiction. We thus assume that $k > 2$ and that any rooted tree T' has at least $2^{k'}$ vertices where the line-labeling of the root of T' is $k' < k$.

The skeleton subgraph $skel(T_r)$ of T_r is a path according to Lemma 2. An end-vertex r' of $skel(T_r)$ has line-labeling $L_r(r') = k$ in T_r but each of its children has line-labeling less than k. Therefore, the vertex r' either has at least a saturated child with line-labeling $k-1$ in T_r or has at least three children with line-labeling $k-1$ in T_r.

Let us first assume that r' has a saturated child u with line-labeling $L_r(u) = k-1$ in T_r. Let x and y be the two children of u in T_r such that $L_r(x) = L_r(y) =$

$k-1$. Let T_x^r and T_y^r be the subtrees of T_r rooted at x and y respectively. Then by Fact 1, the line-labelings of x and y in T_x^r and T_y^r respectively are also $k-1$. Thus from the induction hypothesis, both these subtrees have at least 2^{k-1} vertices. Then, the number of vertices in $T_r > 2^{k-1} + 2^{k-1} = 2^k$.

We thus assume that r' does not have any saturated child with line-labeling $k-1$ in T_r. Then r' has at least three children x, y and z in T_r such that $L_r(x) = L_r(y) = L_r(z) = k-1$. Let T_x^r, T_y^r and T_z^r be the subtrees of T_r rooted at x, y and z respectively. By Fact 1, the line-labelings of x, y and z in T_x^r, T_y^r and T_z^r respectively are also $k-1$. Then from the induction hypothesis, all these subtrees have at least 2^{k-1} vertices. Thus, the number of vertices in $T_r > 2^{k-1} + 2^{k-1} + 2^{k-1} > 2^k$.

Hence, for both these cases, T_r has at least 2^k vertices. □

An immediate consequence of the above lemma is that the line-labeling of the root vertex r of an rooted tree T_r with n vertices is $L_r(r) = O(\log n)$. We now have the following theorem.

Theorem 1 *Let T_r be a rooted tree with n vertices. Then Algorithm **Draw-Rooted** finds a minimum-layer upward drawing of T_r on a grid of area $O(n \log n)$ in $O(n)$ time.*

Proof: Let k be the line-labeling of r in T_r. Then Algorithm **Draw-Rooted** gives a k-layer upward drawing of T_r. By Lemma 1, k also represents the minimum number of layers required for any upward drawing of T_r. The drawing obtained by Algorithm **Draw-Rooted** is thus a minimum-layer upward drawing of T_r. Furthermore, Lemma 4 implies that $k = O(\log n)$.Therefore the height of the drawing is $O(\log n)$. Clearly the width of the drawing is $O(n)$. The area of the drawing is thus $O(n \log n)$. Finally, it is easy to see that Algorithm **Draw-Rooted** can be implemented in $O(n)$ time. □

4 Upward Drawings of Unrooted Trees

In the previous section, we gave a linear-time algorithm to obtain a minimum-layer upward drawing of a rooted tree. In this section, we present a similar result for an unrooted tree.

Let T be an unrooted tree with n vertices. A k-layer upward drawing of T is called a *minimum-layer upward drawing of* T if for each vertex r of T, a minimum-layer upward drawing of T_r requires at least k layers. There are n different rooted trees obtained from T by taking each of its n vertices as the root. A naive approach for obtaining a minimum-layer upward drawing of T would thus compute the minimum-layer upward drawing for all the n rooted trees obtained from T and find the drawing which takes the minimum number of layers among all these drawings. This approach takes $O(n^2)$ time. In this section we give an elegant algorithm to obtain a minimum-layer upward drawing of an unrooted tree T in $O(n)$ time.

We now define the "line-labeling" of a tree T. The *line-labeling of a tree* T, which we have denoted by L_T in the remainder of this paper, is the minimum value among the line-labelings $L_u(u)$ of u in T_u for each of the vertices u of T, that is, $L_T = \min_{u \in V} L_u(u)$, where V is the set of vertices in T. It is easy to see that the line-labeling of an unrooted tree T represents the minimum number of layers required for an upward drawing of T. Let r be a vertex of T such that $L_r(r) = L_T$. Then a minimum-layer upward drawing of T_r obtained by Algorithm **Draw-Rooted** is a minimum-layer upward drawing of T. In the remainder of this section, we thus give an algorithm for computing the line-labeling L_T of T and finding a vertex r of T such that $L_r(r) = L_T$.

The idea of our algorithm is as follows. Let u be a vertex of T and let the degree of u in T be $deg(u) = d$. Let u_1, u_2, ..., u_d be the neighbors of u in T. Given that the line-labelings $L_u(u_1)$, $L_u(u_2)$, ..., $L_u(u_d)$ of the vertices u_1, u_2, ..., u_d in T_u are known, one can compute all the line-labelings $L_{u_1}(u)$, $L_{u_2}(u)$, ..., $L_{u_d}(u)$ of u in T_{u_1}, T_{u_2}, ..., T_{u_d} respectively, in $O(d)$ time. We prove this claim in Lemma 6 but before that we first have the following lemma.

Lemma 5 *Let u be a vertex of a tree T such that u_1, u_2, ..., u_d are the neighbors of u in T and k is the maximum value among the line-labelings $L_u(u_1)$, $L_u(u_2)$, ..., $L_u(u_d)$ of the vertices u_1, u_2, ..., u_d in T_u. Then for each of the neighbors u_m of u in T, the line-labeling of u in T_{u_m} is $L_{u_m}(u) = k + 1$ if one of the following conditions holds:*

(i) u has at least two saturated children with line-labeling k in T_u;

(ii) u has at least four children with line-labeling k in T_u.

Proof: We only prove for the case (i), since the proof in the other case is similar. Let u_i and u_j be the two saturated children of u with line-labeling k in T_u as illustrated in Figure 9(a). Note that all the neighbors of u in T are children of u in T_u. Also note that for any neighbor u_m of u in T, u_m becomes the parent of u in T_{u_m} and the rest of the neighbors of u in T remain the children of u in T_{u_m}. Since the line-labeling of a vertex in a rooted tree is computed in a bottom-up fashion, the line-labelings of all the neighbors of u remain the same in T_u and T_{u_m} except for the vertex u_m. Therefore, u has at least two saturated children u_i and u_j with line-labeling k in T_{u_m} for any neighbors of u in T other than u_i and u_j as illustrated in Figure 9(b). Again, u has at least one saturated child u_j with line-labeling k in T_{u_i} as illustrated in Figure 9(c). Similarly, u has at least one saturated child u_i with line-labeling k in T_{u_j}. Thus in all these scenarios, the line-labeling of u in the corresponding rooted tree is $k + 1$. □

Lemma 6 *Let u be a vertex of a tree T with degree $deg(u) = d$ in T and let u_1, u_2, ..., u_d be the neighbors of u in T. Given that the line-labelings $L_u(u_1)$, $L_u(u_2)$, ..., $L_u(u_d)$ of the vertices u_1, u_2, ..., u_d in T_u are known, one can compute all the line-labelings $L_{u_1}(u)$, $L_{u_2}(u)$, ..., $L_{u_d}(u)$ of u in T_{u_1}, T_{u_2}, ..., T_{u_d} respectively, in $O(d)$ time.*

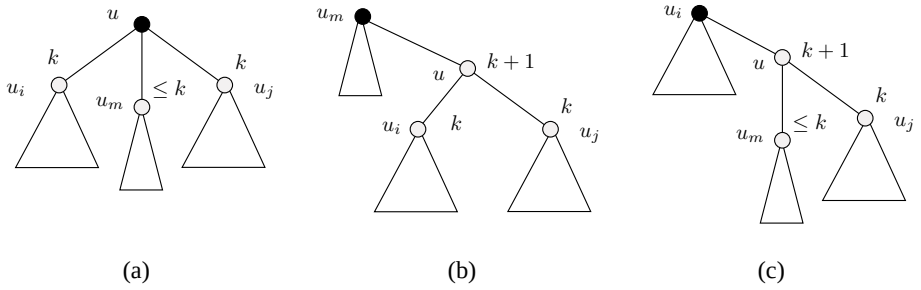

Figure 9: (a) u has at least two saturated children with line-labeling k in T_u, and (b)–(c) u has at least one saturated child with line-labeling k in T_{u_m} for each of its neighbor u_m in T.

Proof: If u has at least two saturated children with line-labeling k in T_u or has at least four children with line-labeling k in T_u, then for any neighbors u_m of u in T, the line-labeling of u in T_{u_m} can be computed in constant time according to the Lemma 5. We thus assume that u has at most three children with line-labeling k in T_u and at most one of them is a saturated child. Note that all the neighbors of u in T are children of u in T_u. Also note that for any neighbor u_m of u in T, u_m becomes the parent of u in T_{u_m} and the rest of the neighbors of u in T remain the children of u in T_{u_m}. Since the line-labeling of a vertex in a rooted tree is computed in a bottom-up fashion, the line-labelings of all the neighbors of u remain the same in T_u and T_{u_m} except for the vertex u_m.

We first consider the situation when none of the children of u in T_u is a saturated child. In this situation, we have the following three cases to consider.
Case 1. *u has three children with line-labeling k in T_u.*

Let u_i, u_j and u_k be the three children of u with line-labeling k in T_u as illustrated in Figure 10(a). In this case, for each of the neighbors u_m of u in T

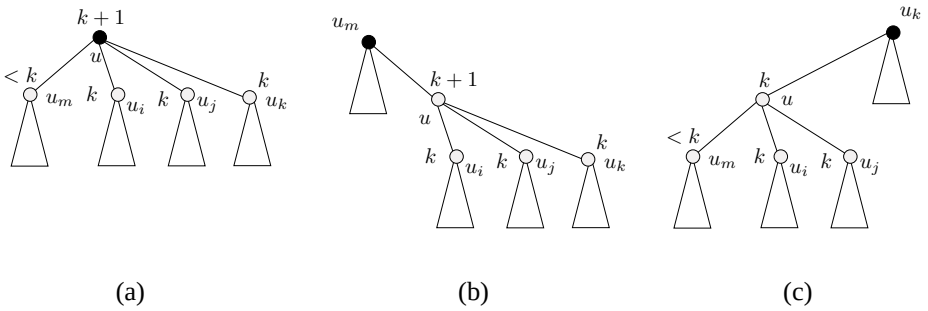

Figure 10: (a) u has three children u_i, u_j and u_k with line-labeling k in T_u, (b) the line-labeling of u is $k+1$ in T_{u_m} for each neighbor u_m of u in T other than u_i, u_j and u_k, and (c) u is a saturated child of u_k with line-labeling k in T_{u_k}.

other than u_i, u_j and u_k, u has three children with line-labeling k in T_{u_m} and

as such, $L_{u_m}(u) = k + 1$ as illustrated in Figure 10(b). Again as illustrated in Figure 10(c), u has two children u_i and u_j having line-labeling k in T_{u_k} and hence u is a saturated child of u_k with line-labeling k in T_{u_k}. Similarly, u is a saturated child of u_i and u_j with line-labeling k in T_{u_i} and T_{u_j} respectively.

Case 2. *u has two children with line-labeling k in T_u.*

Let u_i and u_j be the two children of u with line-labeling k in T_u. In this case, from the same line of reasoning, $L_{u_m}(u) = k$ for each neighbor u_m of u in T (See Figure 11(a)). However as illustrated in Figure 11(b), for each neighbor

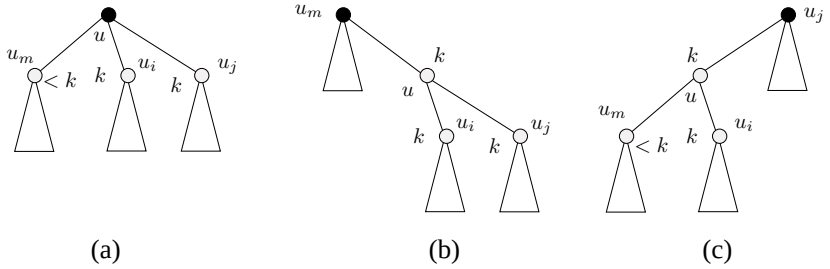

Figure 11: (a) u has two children u_i and u_j with line-labeling k in T_u, (b) u is a saturated child of u_m with line-labeling k in T_{u_m} for each neighbor u_m of u in T other than u_i and u_j, and (c) the line-labeling of u in T_{u_j} is $L_{u_j}(u) = k$.

u_m of u other than u_i and u_j, u has two children with line-labeling k in T_{u_m} and hence u is a saturated child of u_m in T_{u_m} for these vertices.

Case 3. *u has only a single child with line-labeling k in T_u.*

Let u_i be the child of u with line-labeling k in T_u as illustrated in Figure 12(a). From the same line of reasoning, for each neighbor u_m of u in T

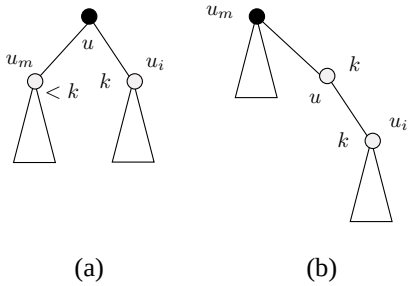

Figure 12: (a) u has only a single child u_i with line-labeling k in T_u, and (b) the line-labeling of u in T_{u_m} is k for each neighbor u_m of u in T other than u_i.

other than u_i, $L_{u_m}(u) = k$ as illustrated in Figure 12(b). However, the value of $L_{u_i}(u)$ cannot be deduced in constant time and must be computed from the values of the line-labelings of all the neighbors of u other than u_i in T_{u_i} or in T_u. (Note that the values are equal in both the rooted trees.) This computation takes $O(d)$ time.

We finally consider the situation when u has exactly one saturated child u_i with line-labeling k in T_u as illustrated in Figure 13(a). In this case, for every neighbor u_m of u in T other than u_i, u has one saturated child with line-labeling k in T_{u_m} and hence the line-labeling of u in T_{u_m} is $k+1$ as illustrated in Figure 13(b). However, the value of $L_{u_i}(u)$ cannot be deduced in constant time and must be computed from the values of the line-labelings of all the neighbors of u other than u_i in T_{u_i} or in T_u. (Again note that the values are equal in both the rooted trees.) This computation also takes $O(d)$ time.

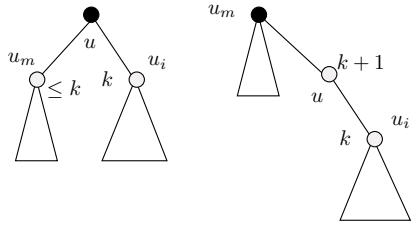

Figure 13: (a) u has a saturated child u_i with line-labeling k in T_u, and (b) the line-labeling of u in T_{u_m} is $k+1$ for each neighbor u_m of u in T other than u_i.

Thus in all the cases, we can compute in $O(d)$ time, the values of the line-labelings of u in all the rooted trees each with one of its neighbors in T as the root. □

We now have the following theorem.

Theorem 2 *Let T be an unrooted tree. One can obtain a minimum-layer upward drawing of T in linear time.*

Proof: From the definition, it is clear that the line-labeling of T represents the minimum number of layers required for an upward drawing of T. Let r be a vertex of T such that $L_r(r) = L_T$. Then a minimum-layer upward drawing of T_r is a minimum-layer upward drawing of T. Therefore, to construct an algorithm for minimum-layer upward drawing of T, we first compute the line-labeling of T. The main idea of this computation is to compute the line-labeling $L_u(u)$ of each vertex u of T in T_u by two traversals of T. For an arbitrarily chosen vertex r of T, we first consider the rooted tree T_r and compute the line-labeling $L_r(u)$ of each vertex u in T_r by a bottom-up computation in T_r according to the definition given in Section 3. Clearly, this traversal takes only linear time. In the next step, for each vertex u of T, we compute the line-labeling $L_u(u)$ of u in T_u by a top-down traversal of T_r as described below. During this top-down traversal of T_r, we maintain the invariance: while traversing a vertex u of T, the line-labeling of each of its neighbors in T_u is known. Note that this invariance is true at the start of the top-down traversal when we traverse the root r of T_r. We now consider the scenario when we traverse a vertex u of T. We can compute

the line-labeling of u in T_u in $O(deg(u))$ time by the definition of Section 3 since for each neighbor v of u in T, the line-labeling $L_u(v)$ of v in T_u is known. Again for all the neighbors v of u in T, we can compute the line-labeling $L_v(u)$ of u in T_v from the line-labeling of the neighbors of u in T_u in $O(deg(u))$ time by the algorithm described in Lemma 6. This fact maintains the invariance for each child v of u in T_r. We can thus compute the line-labeling $L_u(u)$ of each vertex u of T in T_u in the top-down traversal of T_r. This top-down traversal takes $\sum_{u \in V} deg(u) = O(n)$ time.

We then compute the line-labeling L_T of T by computing the minimum value among all the line-labelings $L_u(u)$ of u in T_u and find a vertex r of T such that $L_r(r) = L_T$. We finally obtain a minimum-layer upward drawing of T_r by Algorithm **Draw-Rooted** and this is also a minimum-layer upward drawing of T. Since each step of the algorithm takes linear time, the overall complexity of the algorithm is linear. □

5 Upward Drawings of Rooted Ordered Trees

The drawing we have proposed for a rooted tree in Section 3 does not preserve a given left-to-right ordering of the children of the vertices in the tree. However, if the given rooted tree T_r is ordered, then in an upward drawing of T_r, there is an additional constraint that a given left-to-right ordering of the children of each vertex of T_r is preserved in the drawing [10]. For example, Figure 14(a) illustrates a rooted tree T_a with the root vertex a and a minimum-layer upward drawing of T_a occupies only two layer as illustrated in Figure 14(b) where the

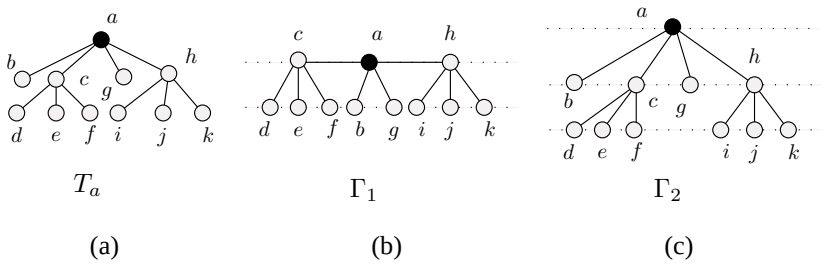

Figure 14: (a) A rooted tree T_a, (b) a minimum-layer upward drawing of T_a where the ordering is not considered, (c) a minimum-layer upward drawing of T_a preserving the order.

ordering of the children of the vertices are not considered. If we consider that T_a is a rooted ordered tree with the root vertex a where the ordering of the children of each vertex is given in the drawing of Figure 14(a), then there is no upward drawing of T_a on two layers. Figure 14(c) illustrates an upward drawing of the rooted tree T_a, which occupies three layers. In this section, we adapt Algorithm **Draw-Rooted** in Section 3 to obtain a minimum-layer upward drawing of a

rooted ordered tree.

Let T_r be a rooted ordered tree with the root vertex r. A path in T_r is called a *left-left path* in T_r if each vertex of the path is the leftmost child of its parent in T_r except for the ancestor of all the vertices. Similarly we define a *right-right path* in T_r. In order to compute a minimum-layer upward drawing of T_r, we introduce another term called the "Ordered line-labelings of the vertices in T_r". Let u be a vertex of T_r. Then the *ordered line-labeling of u in T_r*, which we have denoted by $L'_r(u)$ in the remainder of this paper, is defined as follows.

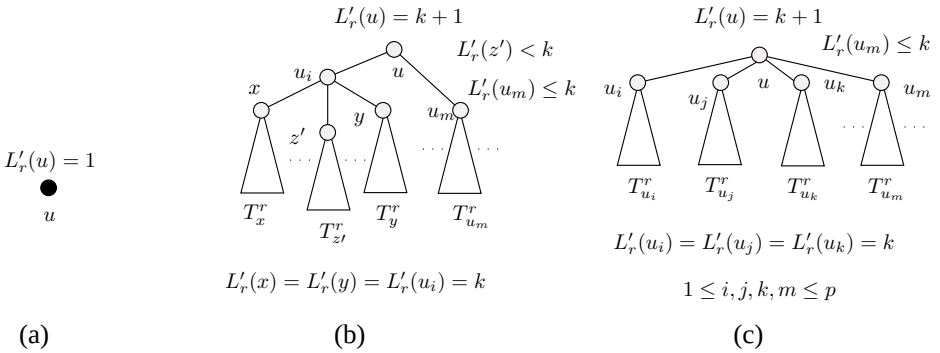

Figure 15: Definition of the ordered line-labeling $L_r(u)$.

(a) If u is a leaf of T_r, then $L'_r(u) = 1$.

(b) If u is a non-leaf vertex of T_r with children u_1, u_2, ..., u_p and k is the maximum among the values $L'_r(u_1)$, $L'_r(u_2)$, ..., $L'_r(u_p)$, then $L'_r(u)$ is defined according to (i)–(iv) as follows.

  (i) If u has at least one child u_i in T_r such that $L'_r(u_i) = k$ and u_i has at least two children, each of which has ordered line-labeling k in T_r, then $L'_r(u) = k+1$. We call a child u_i of u in T_r a *saturated child of* u if u_i has at least two children with the same ordered line-labeling as u_i in T_r.

  (ii) If u has no saturated child with ordered line-labeling k in T_r but has three or more children with ordered line-labeling k in T_r, then $L_r(u) = k+1$.

  (iii) If u has exactly one child u_i with ordered line-labeling k in T_r where u_i is not a saturated child of u in T_r and P is the path induced by the vertices with ordered line-labeling k in $T^r_{u_i}$, then $L'_r(u) = k$ if either u_i is the leftmost child of u and P is a left-left path or u_i is the rightmost child of u and P is a right-right path in T_r; otherwise $L'_r(u) = k+1$.

  (iv) If u has exactly two children u_i and u_j with ordered line-labeling k in T_r where neither of them is a saturated child of u in T_r and u_i comes before u_j in the left-to-right ordering of the children of u in T_r, then, $L'_r(u)$ is defined as follows.

A. If u_i and u_j are the leftmost and the rightmost children of u in T_u, the path induced by the vertices with ordered line-labeling k in $T^r_{u_i}$ is a left-left path in T_r and the path induced by the vertices with ordered line-labeling k in $T^r_{u_j}$ is a right-right path in T_r, then $L_r(u) = k$.

B. Otherwise $L'_r(u) = k + 1$.

Figure 15 illustrates different cases of this definition. The definition above implies that a saturated child of u with ordered line-labeling k in T_r has exactly two children with ordered line-labeling k in T_r. We call a vertex u of T_r with $L_r(u) = k$ in T_r a *left-sided vertex in* T_r if the path induced by all the vertices with ordered line-labeling k in T^r_u is a left-left path in T_r. Similarly, we define a *right-sided vertex in* T_r.

We now have the following lemma, which implies that for a rooted ordered tree T_r, $L'_r(r)$ gives a lower bound on the number of layers in any upward drawing of T_r. The proof of this lemma follows the same line of reasoning as in the proof of Lemma 1.

Lemma 7 *Let T_r be a rooted ordered tree with the vertex r as the root. Then any upward drawing of T_r requires at least $L'_r(r)$ layers.*

We also have the following lemma, which implies that there is also an upward drawing of a rooted ordered tree T_r where the number of layers attains this lower bound.

Lemma 8 *Let T_r be a rooted ordered tree with root vertex r. Then there is a k-layer upward drawing Γ of T_r where $L'_r(r) = k$ such that if r is a left-sided (right-sided) vertex in T_r, then there is no vertex or edge to right (left) of r on the topmost layer in Γ.*

Proof: Let n be the number of vertices in T_r. We prove this lemma by induction on n. The claim is obvious for $n = 1$ since a tree with a single vertex r can be drawn on a single layer with no other vertex or edge occupying the left or right side of r and by definition $L'_r(r) = 1$. We thus assume that $n > 1$ and for any rooted tree T' with less than n vertices, there is a k'-layer upward drawing of T' where k' is the line-labeling of the root in T'. We now obtain a k-layer upward drawing of T_r with n vertices.

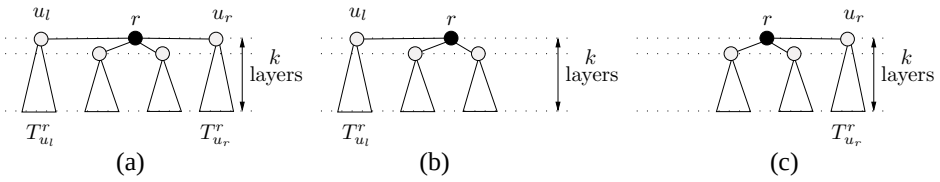

Figure 16: Constructing a k-layer upward drawing of T_r.

Since $L'_r(r) = k$, according to definition, r has at most two children u_l and u_r with ordered line-labeling k in T_r and for the rest of the children u of r, $L'_r(u) < k$. Furthermore, if u_l comes before u_r in the left-to-right ordering of the children of r, then u_l must be a left-sided vertex and the leftmost child of r and u_r must be a right-sided vertex and the rightmost child of u in T_r. Since ordered line-labeling is computed in a bottom-up manner, ordered line-labeling of u_l (u_r) in $T^r_{u_l}$ ($T^r_{u_r}$) is also k. According to the induction hypothesis, there are k-layer upward drawings Γ_l and Γ_r of $T^r_{u_l}$ and $T^r_{u_r}$ respectively such that the right of u_l and the left of u_r are not occupied in these drawings. To obtain a k-layer upward drawing of T_r, we first place the drawing Γ_l to the left of the drawing Γ_r on these k layers. We then place the vertex r on the topmost layer between these two drawings and join the two edges (r, u_l) and (r, u_r) as illustrated in Figure 16(a). For each of the remaining children u of r in T_r, ordered line-labeling of u in $T^r_u < k$ and according to the induction hypothesis, there is a k'-layer upward drawing Γ_u of T^r_u where $k' < k$. We now place these drawings Γ_u of T^r_u for each of the remaining children u of r in T_r on the bottommost $k-1$ layers in the order of the given left-to-right ordering of the children of r in T_r and join all of them with r by an edge without disturbing planarity. We thus obtain a k-layer upward drawing of T_r.

Note that if r does not have two children with ordered line-labeling k in T_r, then Γ_l or Γ_r or both may be empty. More precisely, if r is a left-sided vertex, then the drawing Γ_r is empty and as such the right side of r on the topmost layer is not occupied by any vertex or edge in the drawing as illustrated in Figure 16(b). Similar arguments can also be given if r is a right sided vertex. (See Figure 16(c).) □

The constructive proof of the above lemma gives an algorithm to obtain a k-layer upward drawing of a rooted ordered tree T_r where $L_r(r) = k$. According to Lemma 7, k also represents the minimum number of layers for any upward drawing of T_r. We thus have the following theorem, whose proof is straightforward from Lemma 7 and Lemma 8.

Theorem 3 *Let T_r be a rooted ordered tree with the root vertex r. Then one can obtain a minimum-layer upward drawing of T_r in linear time.*

In this section we have addressed the problem of minimum-layer upward drawings of a rooted ordered tree. We have also given linear-time algorithms to solve this problem for both a rooted tree and an unrooted tree in the previous two sections where the circular ordering of the neighbors of the vertices of the tree is not given as input. Thus it remains to address the problem for an unrooted tree where the circular ordering of the neighbors of each vertex of the tree is given as input. However, it is not very difficult to show that a minimum-layer upward drawing of an unrooted ordered tree can be obtained in linear time using an approach similar to the one presented in Section 4.

6 Conclusion

In this paper, we gave a linear-time algorithm to obtain a minimum-layer upward drawing of a rooted tree T. If T is not a rooted tree, we also gave a linear-time algorithm to select a vertex r of T such that a minimum-layer upward drawing of T_r results in a minimum-layer upward drawing of T. Our algorithm achieves the best area-bound known so far for upward drawing of rooted trees. We also extended the result for rooted ordered trees. It remains our future work to obtain planar straight-line drawings of trees that are not necessarily upward but nevertheless require the minimum number of layers.

Almost all the previous results on layered drawings of trees are based on the "pathwidth" of trees. We studied layered drawings of trees through a new parameter called the "line-labeling" of trees. It remains an open problem to find a relationship between these two parameters of a tree. However, Felsner *et al.* [6] showed that if a planar graph G has a planar straight-line drawing on k layers, then the pathwidth of G is at most k. Thus the pathwidth of a tree T gives a lower bound on the value of the line-labeling of T.

Acknowledgements

We thank the referees for their useful comments that helped us to improve the presentation of the paper. This research work is done under the project "Layered Drawings of Planar Graphs", funded by CASR, BUET.

References

[1] M. J. Alam, M. A. H. Samee, M. M. Rabbi, and M. S. Rahman. Upward drawing of trees on the minimum number of layers (extended abstract). In *the Proceedings of the Workshop on Algortihms and Computation (WALCOM) 2008*, volume 4921 of *Lecture Notes in Computer Science*, pages 88–99. Springer-Verlag, 2008.

[2] S. Cornelsen, T. Schank, and D. Wagner. Drawing graphs on two and three lines. *Journal of Graph Algorithms and Applications*, 8(2):161–177, 2004.

[3] P. Crescenzi, G. Di Battista, and A. Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Computational Geometry: Theory and Application*, 2:187–200, 1992.

[4] P. Crescenzi and A. Piperno. Optimal-area upward drawings of avl trees. In *the Proceedings of Graph Drawing 1994*, volume 894 of *Lecture Notes in Computer Science*, pages 307–317. Springer, 1994.

[5] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, Upper Saddle River, New Jersey, 1999.

[6] S. Felsner, G. Liotta, and S. K. Wismath. Straight-line drawings on restricted integer grids in two and three dimensions. *Journal of Graph Algorithms and Applications*, 7(4):363–398, 2003.

[7] F. Frati. On minimum area planar upward drawings of directed trees and other families of directed acyclic graphs. *Internation Journal of Computational Geometry and Applications*, 18(3):251–271, 2008.

[8] A. Garg, M. T. Goodrich, and R. Tamassia. Area-efficient upward tree drawings. In *the Proceedings of the Symposium on Computational Geometry*, pages 359–368, 1993.

[9] A. Garg, M. T. Goodrich, and R. Tamassia. Planar upward tree drawings with optimal area. *International Journal of Computational Geometry and Applications*, 6(3):333–356, 1996.

[10] A. Garg and A. Rusu. Area-efficient order-preserving planar straight-line drawings of ordered trees. In *the Proceedings of the 9th Annual International Conference on Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 475–486. Springer-Verlag, 2003.

[11] M. Kaufmann and D. Wagner. *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*. Springer-Verlag, London, UK, 2001.

[12] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layouts*. Wiley, New York, NY, USA, 1990.

[13] Y. Shiloach. *Arrangements of Planar Graphs on the Planar Lattice*. PhD thesis, Weizmann Institute of Science, 1976.

[14] C.-S. Shin, S.-K. Kim, and K.-Y. Chwa. Area-efficient algorithms for upward straight-line tree drawings (extended abstract). In *the Proceedings of the 2nd Annual International Conference on Computing and Combinatorics*, volume 1090 of *Lecture Notes in Computer Science*, pages 106–116. Springer-Verlag, 1996.

[15] M. Suderman. Pathwidth and layered drawings of trees. *International Journal of Computational Geometry and Applications*, 14(3):203–225, 2004.

[16] L. Trevisan. A note on minimum-area upward drawing of complete and fibonacci trees. *Information Processing Letters*, 57(5):231–236, 1996.

[17] J. N. Warfield. Crossing theory and hierarchy mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, 7:502–523, 1977.

[18] M. S. Waterman and J. R. Griggs. Interval graphs and maps of DNA. *Bulletin of Mathematical Biology*, 48:189–195, 1986.