Planar Upward Tree Drawings with Optimal Area[*]

Ashim Garg
Dept. of Computer Science
Brown University
Providence, RI 02912–1910
ag@cs.brown.edu

Michael T. Goodrich
Dept. of Computer Science
The Johns Hopkins University
Baltimore, MD 21218–2694
goodrich@cs.jhu.edu

Roberto Tamassia
Dept. of Computer Science
Brown University
Providence, RI 02912–1910
rt@cs.brown.edu

September 23, 1993

Abstract

Rooted trees are usually drawn planar and upward, i.e., without crossings and without any parent placed below its child. In this paper we investigate the area requirement of planar upward drawings of rooted trees. We give tight upper and lower bounds on the area of various types of drawings, and provide linear-time algorithms for constructing optimal area drawings. Let T be a bounded-degree rooted tree with N nodes. Our results are summarized as follows:

- We show that T admits a planar polyline upward grid drawing with area $O(N)$, and with width $O(N^\alpha)$ for any prespecified constant α such that $0 < \alpha < 1$.
- If T is a binary tree, we show that T admits a planar orthogonal upward grid drawing with area $O(N \log\log N)$.
- We show that if T is ordered, it admits an $O(N \log N)$-area planar upward grid drawing that preserves the left-to-right ordering of the children of each node.
- We show that all of the above area bounds are asymptotically optimal in the worst case.
- We present $O(N)$-time algorithms for constructing each of the above types of drawings of T with asymptotically optimal area.
- We report on the experimentation of our algorithm for constructing planar polyline upward grid drawings, performed on trees with up to 24 million nodes.

Key Words Graph drawing, layout, upward drawing, tree, area, grid.

[*]Research at Brown University supported in part by the National Science Foundation under grant CCR-9007851, by the U.S. Army Research Office under grant DAAL03-91-G-0035, and by the Office of Naval Research and the Defense Advanced Research Projects Agency under contract N00014-91-J-4052, ARPA order 8225. Research at The Johns Hopkins University supported in part by the National Science Foundation under Grants CCR-9300079, CCR-9003299 and IRI-9116843, and by the NSF and DARPA under Grant CCR-8908092. An extended abstract of this paper was presented at the *Ninth Annual ACM Symposium on Computational Geometry*, San Diego, CA, May 19–21, 1993.

1 Introduction and Overview

The research area of *graph drawing* presents an exciting connection between computational geometry and graph theory. The general setting is that we are given a graph G, and we wish to produce a geometric representation of G that visualizes G's "important" properties. For example, we may wish to display all the symmetries in G, or, if G contains a Hamiltonian cycle, we may wish to draw G as a regular polygon with chords. The interest in this area has been growing significantly of late (see, e.g., [7, 11, 16, 17, 20]). For example, the annotated bibliography maintained by Di Battista, Eades, and Tamassia [10] mentions more than 250 papers in graph drawing. Important domains of application for graph drawing algorithms include software engineering, project management, visual languages, and VLSI layout.

Perhaps the most studied graph drawing problem is that of producing a planar drawing of a planar graph (e.g., see the classic work of Tutte on planar convex drawings [27] and the recent results on planar straight-line drawings [11, 12, 17, 20, 25]). But there are a variety of other interesting graph drawing problems that are also being investigated of late, such as representing G by means of visibility between geometric figures in the plane (e.g., see [24, 26] and O'Rourke's computational geometry column [21]), or dynamically maintaining a drawing under a sequence of insertions and deletions of vertices and edges, as studied by Cohen *et al.* [7].

1.1 The Problem

An important criterion for a drawing of a graph is that it take up as little area as possible. This is motivated by the finite resolution of all of our current technologies for rendering a drawing, and also by circuit-area optimization criteria in VLSI layout [2, 19, 28]. In the following, we assume the existence of a *resolution rule* that implies a finite minimum area for the drawing of any graph. A typical resolution rule is to require *grid* drawings, where the vertices and bends of the edges have integer coordinates. Indeed, this consideration recently motivated the re-examination of straight-line drawings of planar directed graphs, because they require exponentially-large area [9], whereas several researchers have recently shown that planar graph drawings require only quadratic area, and that such drawings can be produced in linear time [12, 17, 25]. Moreover, some very nice recent work by Kant [17] shows that a number of other aesthetic criteria (such as convex faces) can be satisfied for a planar drawing while still keeping the area quadratic.

In this paper we study area-efficient drawings of rooted trees. The goal of this research is to draw an N-node tree T in as little area as possible while still maintaining certain aesthetic qualities of the drawing. The aesthetic qualities we are particularly interested in are that the drawing be planar and *upward*, i.e., that every edge of T be a vertically monotone chain from the child to the parent, so that the parent u of a node v has y-coordinate greater than or equal to the one of v. This is the natural way in which rooted trees are usually drawn to display their hierarchic structure (e.g., see any undergraduate text in data structures). The difficulty is that most of the known techniques for constructing planar upward drawings of trees require $\Omega(N^2)$ area in the worst case [22, 23].

1.2 Previous Work

If we relax the upward requirement, however, then, as independently shown by Leiserson [19] and Valiant [28], one can construct an $O(N)$-area planar *orthogonal* grid drawing of an N-node tree T, where the nodes are placed at integer grid points and the edges follow paths of the grid. However, Brent and Kung [5] show that if the leaves of an N-node complete binary tree are constrained to be on the convex hull of the drawing, then the drawing needs $\Omega(N \log N)$ area.

Thus, a natural question is whether $O(N)$ area is still achievable for planar upward drawings.

Crescenzi, Di Battista, and Piperno [8] have recently provided a negative answer to this question for the case of *strictly upward* grid drawings, where the nodes have integer coordinates, and the parent of a node has y-coordinate strictly greater than the ones of its children. Namely, they exhibit a family of binary trees that require $\Omega(N \log N)$ area in any strictly upward planar grid drawing. They also show that this lower bound is tight within a constant factor: they give an algorithm that constructs a strictly upward planar straight-line grid drawing of an N-node tree with $O(N \log N)$ area, $O(N)$ height, and $O(\log N)$ width.

Their result doesn't settle the question for the standard notion of upward drawing, however, which allows a child node to be on the same horizontal line as its parent, so long as it is not above its parent. In addition, the issue is clouded somewhat by the fact that producing the exact minimization of the area of the drawing of a tree is NP-hard under several drawing conventions [1, 4, 13]. Nevertheless, Crescenzi *et al.* give $O(N)$ area planar straight-line upward grid drawings of complete binary trees and Fibonacci trees. They do not, however, give a general construction for other types of trees.

Related results on the area requirement of visibility representations of trees are given in [18].

1.3 Our Results

In this paper we show that, for any rooted bounded-degree tree T with N nodes, one can construct a planar upward grid drawing of T with $O(N)$ area in $O(N)$ time, and that such drawing can have width $O(N^\alpha)$, for any prespecified constant α such that $0 < \alpha < 1$. The latter feature provides great flexibility to applications that need to fit the drawing in a prescribed region of the plane. We also extend our approach to trees of arbitrary maximum degree d, at a small additional cost when d exceeds $N^{1-\epsilon}$ for any $\epsilon > 0$. Our drawings do not preserve the left-to-right ordering of the children in T, however. But this should not be surprising, for we show that if one requires a planar upward tree drawing to preserve the left-to-right ordering, then the drawing requires at least $\Omega(N \log N)$ area in the worst case, and we show that this is tight to within a constant factor. Our $O(N)$-area drawing is for the *polyline grid* model, where the nodes of T are mapped to integer grid points, and the edges of T are mapped to polygonal chains with bends at grid points. These polygonal chains need not follow along grid edges, however.

If one desires such a drawing, then in Section 4, we show that one can construct a planar upward orthogonal grid drawing of an N-node binary tree T with $O(N \log\log N)$ area in $O(N)$ time. This $\log\log N$ factor in the area may, at first, seem unnatural, but we show that it is not, for we give an N-node binary tree that requires $\Omega(N \log\log N)$ area for any upward orthogonal grid drawing. Thus, we show that there is an intermediate case between the $\Theta(N)$ area achievable for non-upward planar orthogonal grid tree drawings and the $\Theta(N \log N)$ area achievable for strictly-upward planar grid drawings, or for upward planar grid drawings that preserve the left-to-right order. It is also interesting to observe that the upward requirement penalizes the area less than the requirement of placing the leaves on the same horizontal line, for which the $\Omega(N \log N)$ area bound also applies [5]. We summarize the previous and current bounds on planar grid tree drawing in Table 1.

2 Preliminaries

In this section we give definitions that will be used throughout the paper.

A *drawing* Γ of a graph G maps each vertex of G to a distinct point of the plane and each edge (u, v) of G to a simple Jordan curve with endpoints u and v. We say that Γ is a *straight-line* drawing (see Fig. 1.a) if each edge is a straight-line segment. Γ is a *polyline* drawing (see Fig. 1.b) if each edge is a polygonal chain, and we call *bends* the intermediate vertices of the chain that are

Tree Type	Previous Bounds	Our Bounds
Upward Polyline	$O(N \log N)$ [8]	$\Theta(N)$
Strictly-Upward Straight-Line	$\Theta(N \log N)$ [8]	
Upward Ordered Polyline	–	$\Theta(N \log N)$
Upward Straight-Line (complete binary trees and Fibonacci trees)	$O(N)$ [8]	
Non-Upward Orthogonal	$\Theta(N)$ [19, 28]	
Upward Orthogonal	–	$\Theta(N \log \log N)$
Leaves-on-Hull Orthogonal	$\Theta(N \log N)$ [5]	

Table 1 Area-requirements for planar grid tree drawings.

not vertices of G. Γ is an *orthogonal* drawing (see Fig. 1.c) if each edge is a chain of alternating horizontal and vertical segments. A *grid* drawing is such that the vertices and bends along the edges have integer coordinates. *Planar* drawings, where edges do not intersect, are especially important because they improve the readability of the drawing, and, in the context of VLSI layouts, they simplify the design process [2, 19, 28]. An *upward* drawing of a directed graph is such that every edge is a curve monotonically nondecreasing in the vertical direction (when traversed along the direction of the edge).

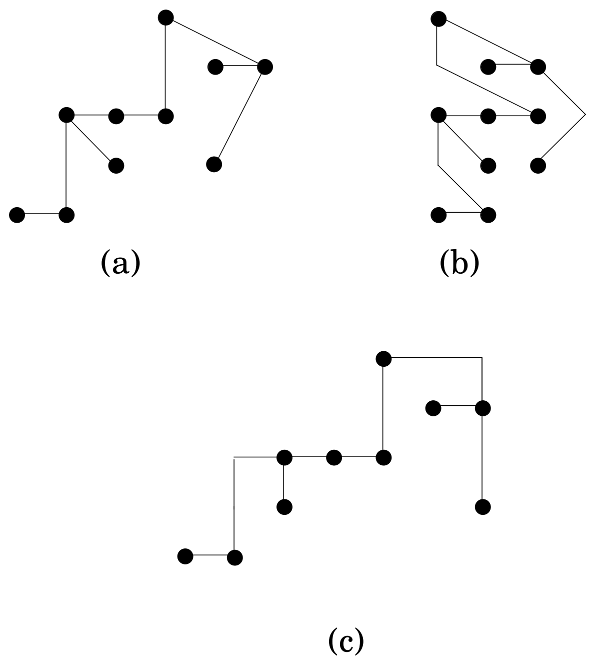


Figure 1: Example of planar upward drawings of the same binary tree: (a) straight-line; (b) polyline; (c) orthogonal.

The *area* of a drawing Γ is the area of the smallest rectangle R with sides parallel to the axes covering the drawing. The *width* and *height* of Γ are the *width* and *height* of R, respectively. We assume the existence of a *resolution rule* that implies a finite minimum area for the drawing of any graph. A typical resolution rule is to require grid drawings. When a resolution rule is given, it is meaningful to consider the problem of finding drawings with minimum area.

Let T be a rooted ordered tree. We assume that each edge of T is directed from the child to the parent. The ordering of the children of a node v will be referred to as their left-to-right order. Hence, the first and last children of v will be referred to as the leftmost child and rightmost child of v, respectively. The *degree* of a node of T is the number of its children. Tree T is said to be *left-heavy* (see Fig. 2.a) if, for every node v of T, the children of v are ordered by nonincreasing size of their subtrees. A *leftmost path* of T is a maximal path consisting of nodes that are leftmost children, except the last node.

A *binary tree* is defined as a rooted tree such that each node has at most two children. Examples of planar upward drawings of a binary tree are given in Fig. 1.

3 Polyline Drawings

In this section we investigate polyline drawings. First, we describe a layering technique that will be used to construct the drawings.

3.1 Upward Layerings

We define the *inorder visit* of a rooted ordered T as follows:

1. recursively visit the first subtree of T;
2. visit the root of T;
3. recursively visit the other subtrees of T, in left-to-right order.

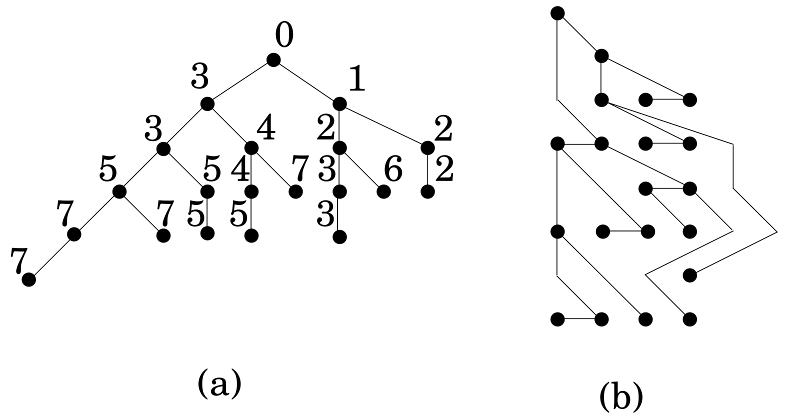


Figure 2: (a) Example of upward layering λ of a left-heavy tree. (b) Drawing associated with λ.

An *upward layering* of T is a mapping λ of the nodes of T to nonnegative integers that satisfies the following properties (see Fig. 2.a):

1. If w is the leftmost child of v, then $\lambda(v) \leq \lambda(w)$;
2. If w is a child of v but not the leftmost child, then $\lambda(v) < \lambda(w)$.
3. If u is the root of T, then $\lambda(u) = 0$.

We say that a node v is assigned to layer i if $\lambda(v) = i$. An edge (u, v) is said to *traverse* layer i if $\lambda(v) < i < \lambda(u)$. The *height* of upward layering λ is defined as $\max_{v \in T} \lambda(v)$. The *width* of a layer i is the number of nodes assigned to layer i plus the number of edges that traverse layer i. The *width* of λ is the maximum width of a layer.

The following theorem shows that an upward layering can be extended to a planar polyline upward grid drawing where the nodes are placed along horizontal lines associated with the layers (see Fig. 2.b).

Theorem 1 *Given an upward layering λ with height H and width W of an N-node ordered tree T, a planar polyline upward grid drawing of T with height W and width H can be constructed in $O(H \cdot W)$ time.*

Proof: First, we insert dummy nodes along the edges that traverse layers. Namely, if edge (u, v) traverses layers i through j, we insert $j - i + 1$ dummy nodes along (u, v), and assign them to layers i through j, respectively. Let T' be the resulting tree. For each node v of T', we set $y(v) = -\lambda(v)$, and $x(v)$ equal to the number of nodes of layer $\lambda(v)$ preceding v in the inorder visit. The edges of T' are then drawn as straight-line segments. Clearly, this yields a straight-line upward grid drawing of T' with height H and width W, such that every edge either joins nodes of consecutive layers, or joins a leftmost child to its parent on the same level. We claim that the drawing is also planar. To prove the claim, we observe: (a) a horizontal edge (u, v) on layer i cannot be crossed because all the nodes between u and v in the inorder sequence are assigned to layers below i; (b) if there were a crossing between edges (v', w') and (v'', w''), where w' precedes w'' in layer i and v'' precedes v' in layer $i + 1$, then we would have that w' precedes w'' in the inorder sequence but v' follows v'' in the inorder sequence, a contradiction. Finally, we obtain a planar polyline upward grid drawing of T by replacing the dummy nodes of T' with bends. The height and width are not affected. The above construction can be easily carried out in time $O(H \cdot W)$. □

Therefore it is sufficient for us to describe how to construct an upward layering of a tree.

3.2 Drawing Algorithm

In this section we describe an algorithm for constructing an upward layering of an N-node rooted tree. If the tree has bounded degree, an upward layering with width $O(N^{\alpha})$ and height $O(N^{1-\alpha})$ can be constructed for any constant α such that $0 < \alpha < 1$.

So, let T be a left-heavy ordered rooted tree with N nodes (we will show how to remove this left-heavy restriction later). The following algorithm constructs an upward layering λ of T with width $O(N^{\alpha} + d \log N)$ and height $O(N^{1-\alpha})$. The algorithm incrementally assembles an ordered sequence Λ of nodes of T, and marks some nodes of T, such that the following invariants are maintained:

1. If u precedes v in Λ, then $\lambda(u) \geq \lambda(v)$; and
2. a node is marked if and only if it is the the first node of its layer contained in Λ.

The assembly of Λ is performed by repetitive insertions of leftmost paths. At the end of the computation, the sequence Λ contains all the nodes of T, so that Λ and the marking of the nodes uniquely identify the upward layering λ. Note that in the sequence Λ a child precedes its parent, while in the layering λ, a child is assigned to a layer number greater than or equal to the one of its parent. The algorithm consists of three steps:

1. *Preprocessing:* Initialize Λ as the leftmost path containing the root of T.
2. *Main Loop:*
   for $k = 1, \cdots, \log N$ **do** (round k)
   (a) Select the nodes v of T such that
      - v is a child of a node of Λ;
      - v is not already in Λ; and

```
        • the subtree rooted at $v$ has size at least $N/2^k$.
    (b) Sort the selected nodes according to the order of their parents in $\Lambda$.
    (c) Partition the sorted sequence of selected nodes into blocks of $2^{\alpha k}$ nodes (with the last
        block possibly having fewer nodes).
    (d) For each block $(v_1, \cdots, v_m)$, let $u$ be the parent of $v_1$. Mark $u$ and successively insert
        into $\Lambda$ the leftmost paths of $v_1, \cdots, v_m$, right before $u$.
  endfor
3. Postprocessing: Scan sequence $\Lambda$, and mark a node if it has $N^\alpha - 1$ unmarked predecessors.
```

Layering λ is finally constructed by assigning node v to layer i if there are i marked nodes following v in Λ.

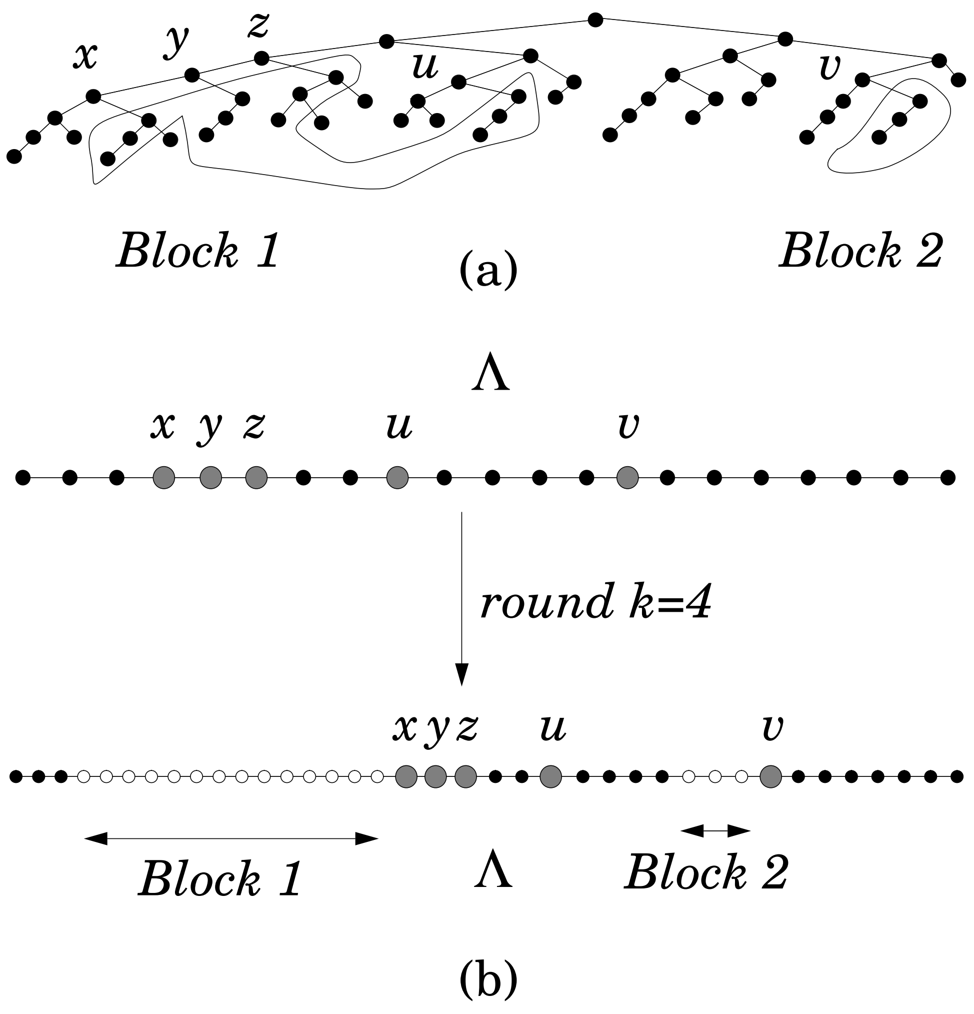


Figure 3: Illustration of the upward layering algorithm of Theorem 2.

The algorithm is illustrated in Fig. 3. An example of the layering λ produced (for a different tree) is shown in Fig. 2.

Theorem 2 *Let α be a constant such that $0 < \alpha < 1$. Given an N-node left-heavy rooted ordered tree T with maximum degree d, an upward layering λ of T with height $H = O(N^{1-\alpha})$ and width $O(N/H + d \log N)$ can be constructed in $O(N)$ time.*

Proof: Every node is eventually inserted into Λ. Namely, after round k all the nodes in the leftmost path of a subtree of size $N/2^k$ are inserted into Λ. Hence, every node is assigned to a layer. All the children of a node v precede v in Λ. Also, if u is a child of v but is not the leftmost child of v, then $\lambda(u) > \lambda(v)$ because either v is marked, or there is a marked node between u and v in Λ. Hence, λ is an upward layering.

We say an edge (u, w) of T is *across* a marked node v if u precedes v, and v precedes w in Λ. After round k, the number of edges across a node is increased by at most $2^{\alpha k} + d - 1$. Hence, at the end of the main loop the number of edges across a marked node is at most

$$\sum_{k=1}^{\log N} \left(2^{\alpha k} + d - 1\right) = \frac{2^\alpha}{2^\alpha - 1}(N^\alpha - 1) + (d-1)\log N.$$

The postprocessing step does not increase the number of edges across a marked node. After the postprocessing step, at most N^α nodes are assigned to a layer. Also, the number of edges that traverse a layer is less than or equal to the number of edges across the marked node of that layer. We conclude that the width of λ is at most

$$N^\alpha + \frac{2^\alpha}{2^\alpha - 1}(N^\alpha - 1) + (d-1)\log N.$$

After round k, the number of marked nodes is increased by at most $2^{(1-\alpha)k} + 1$. Hence the total number of nodes marked in the main loop is at most

$$\sum_{k=1}^{\log N} \left(2^{(1-\alpha)k} + 1)\right) = \frac{2^{1-\alpha}}{2^{1-\alpha} - 1}(N^{1-\alpha} - 1) + \log N.$$

The postprocessing step marks at most $N^{1-\alpha}$ nodes. The height of λ is equal to the number of marked nodes, so that it is at most

$$N^{1-\alpha} + \frac{2^{1-\alpha}}{2^{1-\alpha} - 1}(N^{1-\alpha} - 1) + \log N.$$

To achieve linear time complexity, we set up a data structure that allows us to efficiently perform Steps 2a–2b. We say that a node is *active* for round k if it is in Λ and has a child not in Λ whose subtree has size at least $N/2^k$. A node is called *active* if it is active for any round k. We maintain $\log N$ lists such that, before round k, the k-th list contains the nodes active for round k. Within each list, the active nodes are in the same relative order as in Λ. An active node can appear in more than one list, and has pointers to its representatives in the lists.

The nodes selected in round k are children of the nodes in list k, so that they can be accessed and sorted in Steps 2a–2b in $O(1)$ time per node. Every node v that has more than one child and gets inserted into Λ at round k becomes a new active node, and its representatives are inserted into the appropriate lists. The insertion in each such list is carried out in a manner similar to insertion in Λ. This can be done in $O(1)$ time per representative in Step 2d. Also, whenever a node becomes inactive for round k, we remove its representative from the k-th list. Again, this can be done in $O(1)$ time per representative. Therefore, the total time for maintaining the lists is proportional to the maximum total size of the lists. The k-th list can have at most 2^k nodes. Since a node in list k is also in list k' for $k' > k$, we have that the maximum total size of the lists is

$$\sum_{k=1}^{\log N} 2^k \cdot (\log N - k + 1) = 4N - \log N - 1.$$

The remaining computations can be performed in $O(N)$ time, and we conclude that the time complexity of the algorithm is $O(N)$. □

Theorem 3 *Given a rooted tree T with N nodes and maximum degree d, and a constant α such that $0 < \alpha < 1$, a planar polyline upward grid drawing of T with height $H = O(N^{1-\alpha})$ and width $O(N/H + d \log N)$ can be constructed in $O(N)$ time.*

Proof: First, permute the children of a node so that they are ordered by nonincreasing size. The tree so obtained is left-heavy, so that the result follows from Theorems 1–2. □

Theorem 3 implies the existence of optimal area planar upward polyline grid drawings for trees of maximum degree $O(N^{\delta})$, for any constant δ such that $0 < \delta < 1$.

Corollary 1 *Given a bounded-degree rooted tree T with N nodes, and a constant α such that $0 < \alpha < 1$, a planar polyline upward grid drawing of T with area $O(N)$, height $H = O(N^{1-\alpha})$ and width $O(N/H)$ can be constructed in $O(N)$ time.*

Corollary 2 *Let α and δ be constants such that $0 < \delta < \alpha < 1$. Given a rooted tree T with N nodes and maximum degree $O(N^{\delta})$, a planar polyline upward grid drawing of T with area $O(N)$, height $H = O(N^{1-\alpha})$ and width $O(N/H)$ can be constructed in $O(N)$ time.*

The algorithm described in this section has been implemented for binary trees. The drawing produced for a complete binary tree with 63 nodes and $\alpha = 1/2$ is shown in Fig. 4.

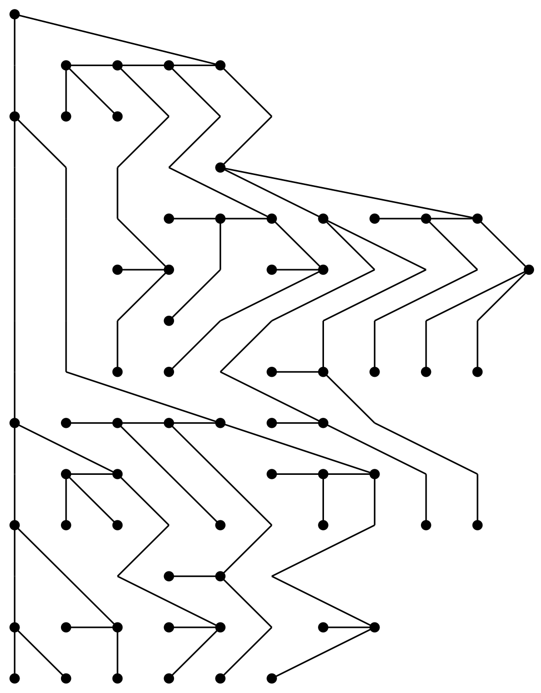


Figure 4: Drawing of the complete binary tree with 63 nodes produced by the implementation of the algorithm of Corollary 1 with $\alpha = 1/2$.

3.3 Order Preserving Drawings

The drawings obtained with the above algorithm do not preserve the left-to-right order of the children. This is justified, however, by the following lower bound:

Theorem 4 *The N-node ordered binary tree B_N requires $\Omega(N \log N)$ area in any planar upward polyline grid drawing that preserves the order of the children.*

Our proof of this theorem is based upon the following lemma:

Lemma 1 *Any planar upward polyline grid drawing of the complete binary tree with N nodes has $\Omega(\log N)$ width and $\Omega(\log N)$ height.*

Proof: Let us denote by $H(N)$ and $W(N)$, the minimum height and width, respectively, of an upward polyline grid drawing of an N node complete binary tree T.

In any upward polyline grid drawing of T, a node, its children and its grand-children can not all be placed at the same height. Hence we get the recurrence $H(N) \geq H((N-3)/4) + 1$; $H(1) = H(3) = 0$. Therefore $H(N) = \Omega(\log N)$.

The width of any upward polyline grid drawing of T is at least one plus the minimum of the widths (of the drawing) of the subtrees of T in the drawing. Therefore $W(N) \geq W((N-1)/2) + 1$ with $W(1) = 0$. Hence $W(N) = \Omega(\log N)$. □

Proof: (For Theorem 4) Let B_N be an ordered binary tree comprising (see Fig. 5.a):

- a chain with $N/3$ nodes, alternating between left and right children;
- $N/3$ leaves attached to each node of the chain, alternating as left and right children; and
- a complete subtree with $N/3$ nodes attached to the bottommost node of the chain.

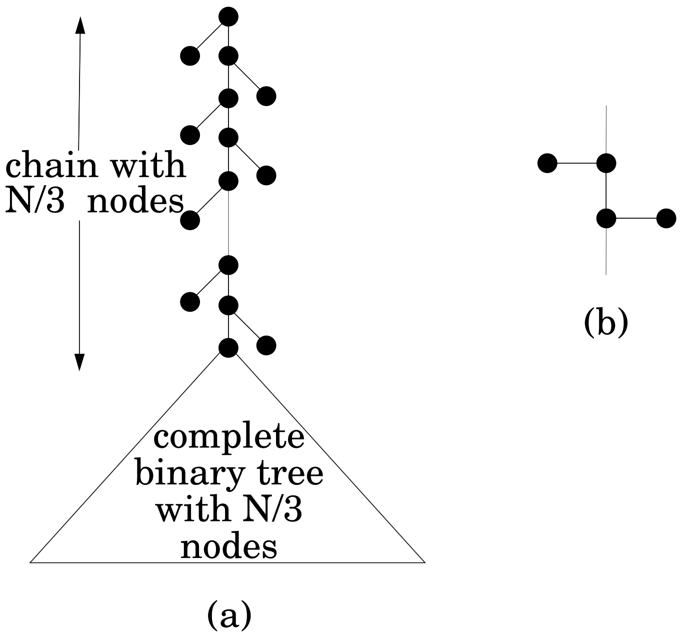
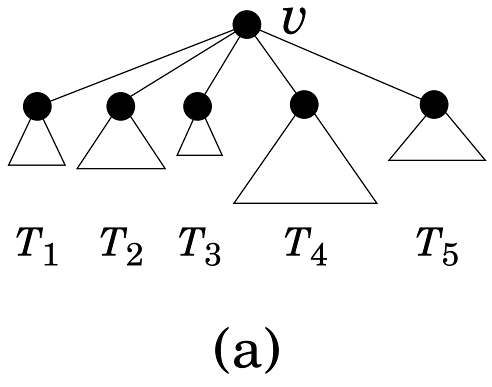


Figure 5: Order-preserving drawings: (a) Tree for the lower bound of Theorem 4. (b) Each edge of chain increases height by at least one unit.

In any planar upward polyline grid drawing of B_N, because of the order of the children, each edge of the chain contributes at least one unit to the height of the drawing (see Fig. 5.b) so that the chain requires $\Omega(N)$ height. By Lemma 1, the complete subtree requires $\Omega(\log N)$ width. □

Incidentally, the lower bound of Theorem 4 is tight. It can be achieved with the following simple recursive algorithm:

1. Let $T_1, T_2, \ldots, T_m$ be the subtrees of a bounded degree tree T whose root is v (see Fig. 6.a-here m is five). Let the root of the subtree T_i be denoted by v_i. Recursively construct the drawings of each T_i. Vertically stack their drawings (see Fig. 6.b) so that the subtree at the bottom has the maximum size among the subtrees (other subtrees can be placed in any order, e.g. in Fig. 6.b, they are in the order T_1, T_2, T_3, T_5 from bottom to top). Place the root v above the drawing of the topmost subtree.
2. Now for every T_i, draw edge (v_i, v) as a polyline that uses one vertical track (grid column) either on the left or on the right of each T_j drawn above T_i depending upon whether T_j is to the left or right of T_i in T, and correspondingly switches from left to right or vice-versa (see Fig. 6.b). This completes the construction of the drawing of the tree T.



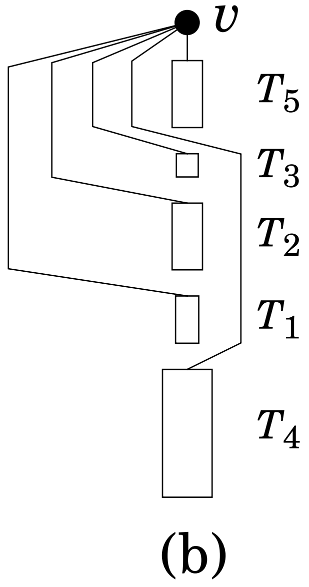


Figure 6: Example with $m = 5$ for constructing order-preserving drawings: (a) Tree T with subtrees $T_1, T_2, \ldots, T_5$. (b) Drawing of T, where the rectangles represent the drawings of the T_i's.

Theorem 5 *Given a bounded-degree tree T with N nodes, a planar polyline upward grid drawing of T with width $O(\log N)$, height $N-1$ and area $O(N \log N)$ that preserves the order of the children can be constructed in $O(N)$ time.*

Proof: Let $T_1, T_2, \ldots, T_m$ be the subtrees of T and v be the root of T. Let the root of the subtree T_i be denoted by v_i.

Since m is a constant, only a constant number (say c) of vertical tracks are needed to route all the edges of the type (v_i, v). Let T_k be the subtree that is drawn bottommost and hence has the maximum size among the subtrees. Therefore if we denote by $W(T)$ the width of the drawing of T, we have:

$$W(T) \leq \max(W(T_k), \max_{i \neq k}(W(T_i)) + c) \tag{1}$$

Let $width(N)$ be the maximum width of the drawing, constructed by the algorithm, of any tree with N nodes. Now we show inductively that $width(N) \leq \alpha \log_2 N$, for some constant $\alpha \geq c$. This is trivially true for $N = 1$ since $width(1) = 0$.

Now suppose this is true for any tree with less than N nodes. If N is the size of a tree T, then the size of T_k (as defined above) is at most $N-1$ and since there can be at most two subtrees of sizes both at least $(N-1)/2$, size of any subtree T_i of T, for $i \neq k$ is at most $(N-1)/2$. Therefore from equation 1 and our inductive hypothesis, $width(N) \leq \max(\alpha \log_2(N-1), \alpha \log_2((N-1)/2) + c)$. For $\alpha \geq c$, we have $width(N) \leq \alpha \log_2 N$.

It is easy to prove by a simple inductive argument that the height of the drawing of T is $N-1$. Thus the area of the drawing is $O(N \log N)$.

The algorithm can be trivially implemented in linear time. □

4 Orthogonal Drawings

In this section we consider planar orthogonal upward grid drawings of binary trees, and provide a tight $\Theta(N \log\log N)$ bound on the area.

4.1 Drawing Algorithm

First, we present a simple straight-line drawing algorithm that will be later used as a subroutine. We say that a node in a drawing is *obstructed* if the the vertical line through v intersects the drawing below v. The algorithm is based on the proof of the following lemma:

Lemma 2 *Let T be a binary tree with N nodes. A planar straight-line orthogonal upward grid drawing of T with width at most N and height at most* $\log N$*, such that every node of degree 1 or 2 is not obstructed, can be constructed in $O(N)$ time.*

Proof: The algorithm first transforms the tree into a left-heavy binary tree. Then for every node v, it sets $y(v)$ equal to the number of nodes on the path from v to the root that are right children. Finally traversing the left-heavy binary tree in the post order sequence, for every node v, it sets $x(v)$ equal to the x-coordinate of its right child, if it has one otherwise sets $x(v)$ to one plus the x-coordinate of the last vertex visited before it. □

An example of a drawing constructed by the algorithm of Lemma 2 is shown in Fig. 7.a. Now, we recall some definitions on separators of binary trees. Let T be a binary tree with N nodes. A *partial tree* of T is a tree which is a subgraph of T. (Note the difference between partial tree and subtree.) A *separator* of a binary tree T is an edge of T whose removal divides T into two partial trees, each with at least $N/3$ nodes and at most $2N/3$ nodes (e.g., see Chazelle [6]). A recursive decomposition of T by separators defines a binary tree S, called *separator tree*, where each leaf of S corresponds to a node of T, and each internal node μ of S corresponds to a partial tree T_μ of T and to the separator $s_\mu = (u, v)$ of T_μ, with the left child of μ being associated to the partial tree of T_μ rooted at u, and the right child associated with the rest of T_μ. Tree S has $2N-1$ nodes, height at most $\log_{3/2} N$, and can be constructed in $O(N)$ time (e.g., see Guibas *et al.* [15]).

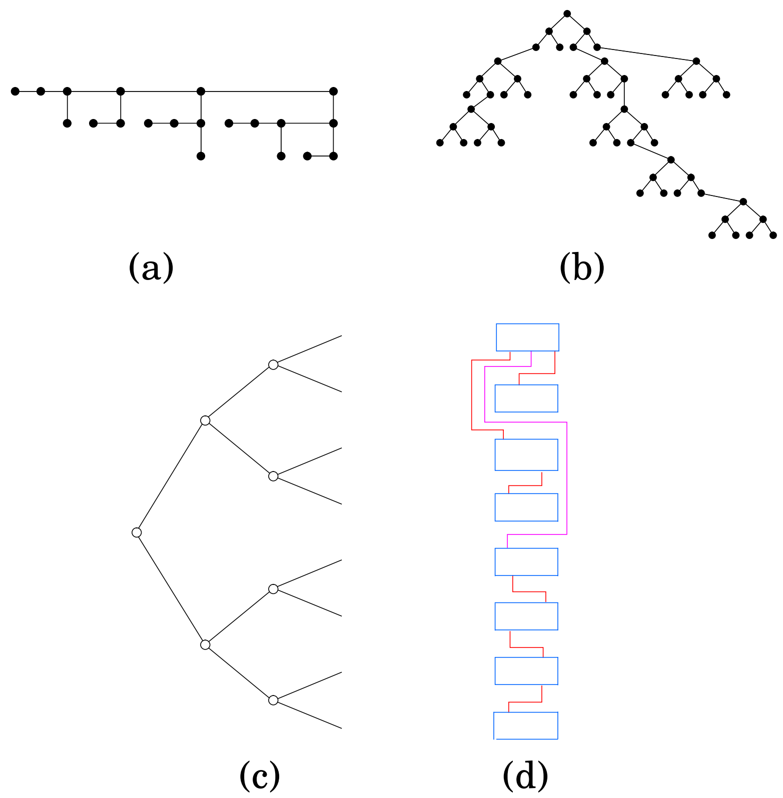


Figure 7: Illustration of the Algorithm of Theorem 6: (a) Example of a drawing produced by the algorithm of Lemma 2. (b) A tree T and the separators that join blocks. (c) Truncated separator tree of T. (d) Drawing of T, where the rectangles represent the drawings of the blocks.

The algorithm for constructing a planar orthogonal upward grid drawing of an N-node binary tree T is outlined below (see Fig. 7):

1. Construct the separator tree S of T.
2. Remove from S the nodes associated with partial trees with less than $\log N$ nodes, and let S' be the resulting truncated separator tree, which has $O(N/\log N)$ nodes and $O(\log(N/\log N)) = O(\log N)$ height. (See Fig. 7.b–c.)
3. For each leaf μ of S', construct a drawing of the associated partial tree T_μ, called a *block*, using the algorithm of Lemma 2. Since T_μ has $\Theta(\log N)$ nodes, its drawing has $O(\log N)$ width and $O(\log\log N)$ height.
4. Place the drawings of the blocks vertically one above the other, sorted from bottom to top according to the inorder sequence of the associated nodes of S'.
5. For each internal node ν of S', route separator $s_\nu = (u, v)$, on the current drawing, creating bends and adding extra tracks (grid rows or columns), whenever needed. (See Fig. 7.d.)

Theorem 6 *Given a binary tree T with N nodes, a planar orthogonal upward grid drawing of T with $O(N)$ bends, $O(N\log\log N)$ area, $O(\log N)$ width, and $O(N\log\log N/\log N)$ height can be constructed in $O(N)$ time.*

Proof: By Lemma 2, the union of the drawings of the blocks constructed in Step 4 has width $O(\log N)$ and height $O(N\log\log N/\log N)$. We show that all the separators can be routed in Step 5 by adding a total of $O(\log N)$ vertical tracks and $O(N/\log N)$ horizontal tracks. We say that a separator *spans* a block T_μ if its endpoints are one below and the other above the drawing of T_μ. A separator s_ν is routed using one vertical track either on the left or on the right side of the drawing of a block T_μ spanned by s_ν, depending on whether T_μ is to the left or the right of the path from s_ν to the root in modified T (modified because of the conversion of each T_μ into a left-heavy tree in lemma 2). A *switch* occurs when s_ν changes side between two blocks or enters a block. Each switch needs a distinct horizontal track and two bends. For each basic block T_μ, only the separators associated with ancestors of μ in S can span T_λ, so that $O(\log N)$ extra vertical tracks are sufficient to route all the separators in Step 5. The number of horizontal tracks added in Step 5 is equal to the total number of switches. The number of switches in the routing of separator s_ν is bounded by the height of the subtree rooted at ν in S'. If ν corresponds to a partial subtree of size k, the height of the subtree rooted at ν in S' is $O(\log(k/\log N))$. Thus the total number of switches $s(N)$ is the solution of the recurrence $s(k) = s(k/c)+s(k(1-1/c))+O(\log(k/\log N))$; $s(\log N) = 0$, where c lies between 2 and 3. It is easy to see that $s(k)$ is less than or equal to $c_1 k/\log N - c_2 \log(k/\log N) - c_3$ for appropriate constants c_1, c_2 and c_3. Therefore the total number of switches is $O(N/\log N)$. □

This $\log\log N$ factor in area may at first seem unnatural, but it is not.

4.2 A Superlinear Lower Bound on the Area

We show that the superlinear area-bound of Theorem 6 is tight. Let T_N be the N-node binary tree consisting of (see Fig. 8):

- a chain C with $N/3$ nodes, where every $\sqrt{\log N}$-th node of C is called a *joint* of C;
- $N/3\sqrt{\log N}$ complete subtrees, where each subtree has $\sqrt{\log N}$ nodes and is rooted at a child of a joint of C;
- a complete subtree with $N/3$ nodes, which is rooted at a child of the first node of C.

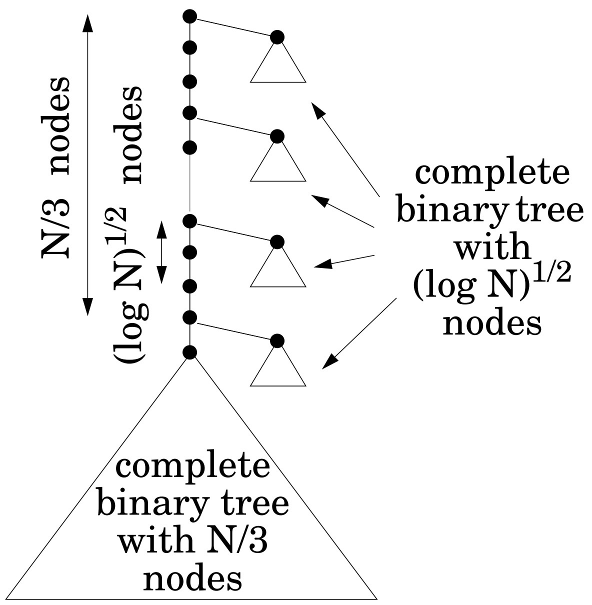

N/3 nodes
(log N)^1/2 nodes
complete binary tree with (log N)^1/2 nodes
complete binary tree with N/3 nodes

Figure 8: Tree for the lower bound of Theorem 7.

Theorem 7 *Any planar upward orthogonal grid drawing of the N-node tree T_N requires area $\Omega(N \log\log N)$.*

Proof: Consider any planar upward orthogonal grid drawing of T_N, and let W and H be the width and height of the drawing, respectively. If W is more than $N/6$, then the area of the drawing is $\Omega(N \log N)$ since, by Lemma 1, $H = \Omega(\log N)$. Now, suppose W is at most $N/6$. Since T_N contains a complete subtree with $N/3$ nodes, by Lemma 1 we have that W is $\Omega(\log N)$. Consider the subdrawing of any subchain S of C with $2W$ nodes. We claim that the height of the subdrawing of S is $\Omega(\log \log N)$. Since the drawing is upward, the subdrawings of any two consecutive subchains of C must be vertically stacked. Hence, the claim implies the statement of the theorem.

The proof of the claim is illustrated in Fig. 9. For the sake of contradiction, we need only

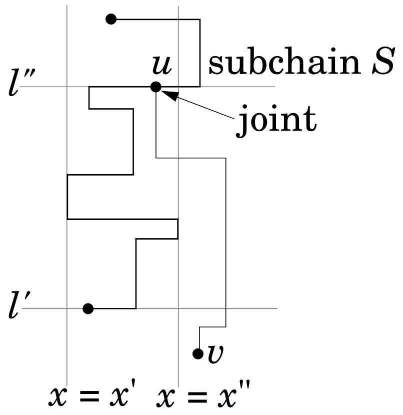


Figure 9: Drawing of subchain S in the proof of Theorem 6.

to consider the case when the height of the subdrawing of S is less than $\log \log N$. Let ℓ' be the horizontal line through the bottommost node of S in the drawing. Since S has $2W$ nodes and is drawn with width at most W, it contains at least W obstructed nodes (recall the definition of "obstructed" from Section 4.1). Also, by a simple pigeon-hole argument, there are at least $W/\log \log N$ obstructed nodes of S on the same horizontal line ℓ'', and such nodes form a subchain of S. Thus, there are at least $W/(\sqrt{\log N} \log \log N)$ obstructed joints along line ℓ''. Consider the subtrees connected to such obstructed joints. These subtrees are drawn below line ℓ''. If any such subtree is drawn entirely above ℓ', then by Lemma 1 the height of the subdrawing of S is $\Omega(\log \log N)$, and the claim is verified. Otherwise, every subtree has a leaf v below line ℓ', and we consider the path from v to its closest ancestor joint u. Let x' and x'' be the minimum and maximum x-coordinates of the subdrawing of S below line ℓ''. The path between u and v must intersect one of the vertical lines at $x = x' - 1$ or $x = x'' + 1$. Also, since the drawing is upward, orthogonal and grid, such intersections must occur at distinct grid points of these two vertical lines, and between ℓ' and ℓ''. Since there are at least $W/(\sqrt{\log N} \log \log N)$ such paths, we have that the height of the subdrawing of S is at least $W/(2\sqrt{\log N} \log \log N)$. Recalling that $W = \Omega(\log N)$, we conclude that the height of the subdrawing of S is $\Omega(\sqrt{\log N}/(2 \log \log N)) = \Omega(\log \log N)$, which contradicts our height assumption about S. This completes the proof of the claim. □

The drawings constructed by the algorithm of Theorem 6 do not preserve the left-to-right order of the children. Note that the lower bound of Theorem 4 applies also to orthogonal drawings.

5 Experimental Results

We have implemented the algorithm of Corollary 1 given in section 3 for constructing planar upward polyline drawings of trees, on a Sun Sparcstation 10 in language "C". Our implementation takes a binary tree as its input. The implementation places $\lceil 2^{\alpha^k} \rceil$ nodes in a block in step 2c of the algorithm. The theoretical upper bound for width and height of the drawing with $\alpha = 1/2$, of a N-node binary tree, produced by our implementation, computed as in the proof of theorem 2 is

$$\begin{aligned} Width &= \lceil (3+\sqrt{2})\sqrt{N} + \log_2 N \rceil \\ Height &= \lceil (3+\sqrt{2})\sqrt{N} + \log_2 N \rceil \end{aligned}$$

The additive term of $\log_2 N$ appears in width because the implementation places $\lceil 2^{\alpha k} \rceil$ nodes in a block in the step 2c as compared to $2^{\alpha k}$ nodes as described in the algorithm. The ratio of theoretical area bound of drawing and number of nodes in input tree therefore lies between 19.48 and 33.34.

The experimental results obtained for complete binary trees and Fibonacci trees, with $\alpha = 1/2$ are presented in Table 2. In the table, we have denoted the theoretical area bound by A_{th} and the experimental area by A_{ex}. Fig. 10 and Fig. 11 give a comparison of the the ratios $r_{th} = A_{th}/N$ and $r_{ex} = A_{ex}/N$ for complete binary and Fibonacci trees respectively. The value of r_{ex} for both complete binary and Fibonacci trees is less than 5 even up to about 24 million nodes and hence the algorithm is quite area-efficient in practice.

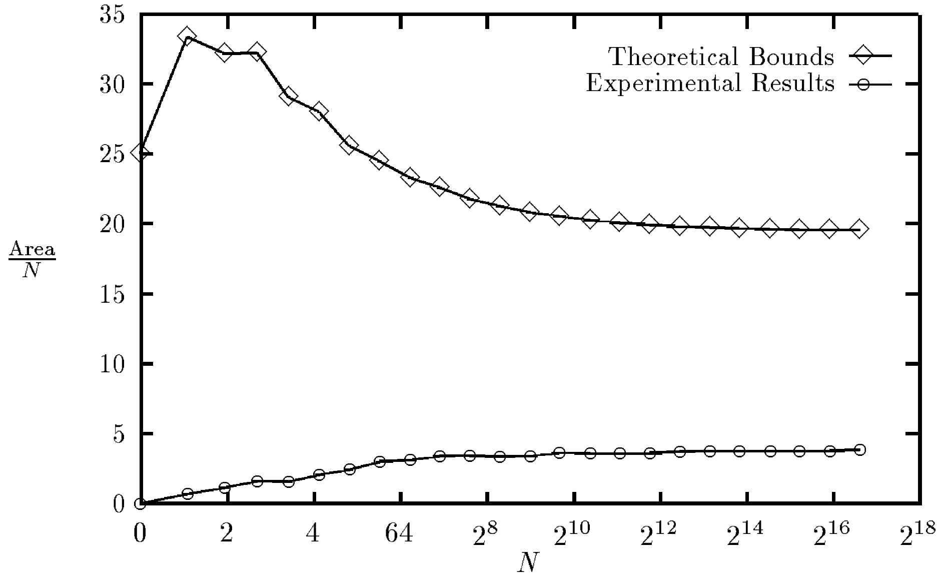


Figure 10: Experimental results on the area of the drawings of complete binary trees produced by the algorithm of Corollary 1 with $\alpha = 1/2$, and comparison with the theoretical upper bounds

6 Conclusion

In this paper we have investigated the area requirement of different types of planar upward drawings of a rooted bounded-degree tree T with N nodes. Our results are summarized as follows: T admits a planar upward polyline grid drawing with area $O(N)$, and with width $O(N^{\alpha})$ for any prespecified constant α such that $0 < \alpha < 1$. If T is a binary tree, then T admits a planar orthogonal upward grid drawing with area $O(N \log\log N)$. If T is ordered, then T admits an $O(N \log N)$-area planar upward grid drawing that preserves the left-to-right ordering of the children of each node. All of the above area bounds are asymptotically optimal in the worst case. Also, there are $O(N)$-time algorithms for constructing each of the above types of drawings of T with asymptotically optimal area.

	# Nodes N	Theoretical Bounds				Experimental Results			
		Width	Height	Area A_{th}	$r_{th} = \frac{A_{th}}{N}$	Width	Height	Area A_{ex}	$r_{ex} = \frac{A_{ex}}{N}$
Complete	15	22	22	484	32.27	5	6	30	2.00
Binary	31	30	30	900	29.03	8	7	56	1.81
Tree	63	42	42	1764	28.00	11	13	143	2.27
	127	57	57	3249	25.58	18	18	324	2.55
	255	79	79	6241	24.47	27	29	783	3.07
	511	109	109	11881	23.25	40	41	1640	3.21
	1023	152	152	23104	22.58	62	57	3534	3.45
	2047	211	211	44521	21.75	90	79	7110	3.47
	4095	295	295	87025	21.25	121	114	13794	3.37
	8191	413	413	170569	20.82	178	158	28124	3.43
	16383	580	580	336400	20.53	261	228	59508	3.63
	32767	815	815	664225	20.27	374	314	117436	3.58
	65535	1147	1147	1315609	20.07	521	451	234971	3.59
	131071	1616	1616	2611456	19.92	746	632	471472	3.60
	262143	2279	2279	5193841	19.81	1071	905	969255	3.70
	524287	3216	3216	10342656	19.73	1542	1284	1979928	3.78
	1048575	4541	4541	20620681	19.67	2182	1802	3931964	3.75
	2097151	6414	6414	41139396	19.62	3074	2553	7847922	3.74
	4194303	9063	9063	82137969	19.58	4344	3616	15707904	3.75
	8388607	12808	12808	164044864	19.56	6192	5108	31628736	3.77
	16777215	18105	18105	327791025	19.54	8902	7229	64352558	3.84
Fibonacci	20	25	25	625	31.25	6	7	42	2.10
Tree	33	31	31	961	29.12	10	9	90	2.73
	54	39	39	1521	28.17	10	11	110	2.04
	88	48	48	2304	26.18	15	14	210	2.39
	143	60	60	3600	25.17	24	19	456	3.19
	232	76	76	5776	24.90	26	23	598	2.58
	376	95	95	9025	24.00	42	30	1260	3.35
	609	119	119	14161	23.25	56	40	2240	3.68
	986	149	149	22201	22.52	64	48	3072	3.12
	1596	187	187	34969	21.91	86	62	5332	3.34
	2583	236	236	55696	21.56	130	77	10010	3.88
	4180	298	298	88804	21.24	194	95	18430	4.41
	6764	376	376	141376	20.90	193	122	23546	3.48
	10945	476	476	226576	20.70	276	156	43056	3.93
	17710	602	602	362404	20.46	384	194	74496	4.21
	28656	763	763	582169	20.32	402	252	101304	3.54
	46367	967	967	935089	20.17	577	320	184640	3.98
	75024	1226	1226	1503076	20.03	825	404	333300	4.44
	121392	1555	1555	2418025	19.92	831	521	432951	3.57
	196417	1974	1974	3896676	19.84	1282	659	844838	4.30
	317810	2507	2507	6285049	19.78	1705	832	1418560	4.46
	514228	3185	3185	10144225	19.73	1717	1070	1837190	3.57
	1346268	5143	5143	26450449	19.65	3463	1705	5904415	4.39
	2178308	6537	6537	42732369	19.62	4824	2141	10328184	4.74
	3524577	8309	8309	69039481	19.59	4890	2733	13364370	3.79
	5702886	10564	10564	111598096	19.57	6968	3494	24346192	4.27
	9227464	13433	13433	180445489	19.56	9530	4425	42170250	4.57
	14930351	17081	17081	291760561	19.54	9980	5664	56526720	3.79
	24157816	21721	21721	471801841	19.53	14633	7202	105386866	4.36

Table 2 Experimental results on the drawings of complete binary trees and Fibonacci trees produced by the implementation of the algorithm of Corollary 1 with $\alpha = 1/2$, and comparison with the theoretical upper bounds.

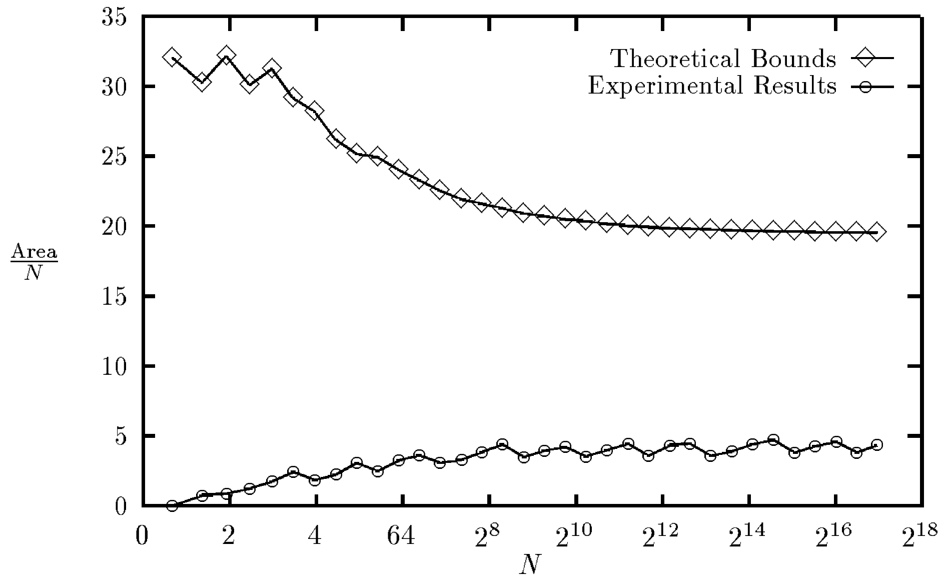


Figure 11: Experimental results on the area of the drawings of Fibonacci trees produced by the algorithm of Corollary 1 with $\alpha = 1/2$, and comparison with the theoretical upper bounds

In view of our results, the main open problem on this subject is determining the area requirement of a planar upward *straight-line* drawing of T. There is still a gap between the trivial $\Omega(N)$ lower bound and the $O(N \log N)$ upper bound given in [8]. It would also be interesting to extend our results to unbounded degree trees. A related open problem is to investigate the area requirement of planar upward straight-line drawings of rooted trees such that the *angular resolution* (i.e., the minimum angle between any two edges incident on the same node) is maximized. Previous results on the angular resolution of (non upward) drawings of graphs appear in [14, 20].

Acknowledgement

We would like to thank Günter Rote for useful comments.

References

[1] S. Bhatt and S. Cosmadakis, "The Complexity of Minimizing Wire Lengths in VLSI Layouts," *Information Processing Letters*, vol. 25, 1987, pp. 263–267.

[2] S.N. Bhatt and F.T. Leighton, "A Framework for Solving VLSI Graph Layout Problems," *J. Computer Systems Sciences*, vol. 28, 1984, pp. 300–343.

[3] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications*, North-Holland, 1976.

[4] F.J. Bradenburg, "Nice Drawings of Graphs and Trees are Computationally Hard," Technical Report MIP-8820, Fakultat fur Mathematik und Informatik, Univ. Passau, 1988.

[5] R.P. Brent and H.T. Kung, "On the Area of Binary Tree Layouts," *Information Processing Letters*, vol. 11, 1980, pp. 521–534.

[6] B. Chazelle, "A Theorem on Polygon Cutting with Applications," *Proc. IEEE Symp. on Foundations of Computer Science*, 1982, pp. 339–349.

[7] R.F. Cohen, G. Di Battista, R. Tamassia, I.G. Tollis, and P. Bertolazzi, "A Framework for Dynamic Graph Drawing," *Proc. ACM Symp. on Computational Geometry*, 1992, pp. 261–270.

[8] P. Crescenzi, G. Di Battista, and A. Piperno, "A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees," to appear in *Computational Geometry: Theory and Applications.*

[9] G. Di Battista, R. Tamassia, and I.G. Tollis, "Area Requirement and Symmetry Display of Planar Upward Drawings," *Discrete & Computational Geometry*, vol. 7, 1992, pp. 381–40.

[10] G. Di Battista, P. Eades and R. Tamassia, "Algorithms for Drawing Graphs: An Annotated Bibliography," revised version in preparation, 1993. See also Technical Report CS-89-09, Dept. of Computer Sci., Brown Univ., 1989.

[11] G. Di Battista and L. Vismara, "Angles of Planar Triangular Graphs," *Proc. ACM Symp. on Theory of Computing*, 1993.

[12] H. de Fraysseix, J. Pach, and R. Pollack, "Small Sets Supporting Fary Embeddings of Planar Graphs," Proc. 20th ACM Symp. on Theory of Computing, 1988, pp. 426-433.

[13] P. Eades, T. Lin, and X. Lin, "Two Tree Drawing Conventions," Technical Report 174, Key Centre for Software Technology, Dept. of Computer Science, The Univ. of Queensland, 1990.

[14] M. Formann, T. Hagerup, J. Haralambides, M. Kaufmann, F.T. Leighton, A. Simvonis, E. Welzl, and G. Woeginger, "Drawing Graphs in the Plane with High Resolution," *Proc. IEEE Symp. on Foundations of Computer Science*, 1990, pp. 86–95.

[15] L.J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R.E. Tarjan, "Linear-Time Algorithms for Visibility and Shortest Path Problems inside Triangulated Simple Polygons," *Algorithmica*, vol. 2, 1987, pp. 209–233.

[16] M. Furer, B. Raghavachari, X. He, and M.-Y. Kao, "$O(n \log n)$-Work Parallel Algorithms for Straight-Line Grid Embeddings of Planar Graphs," *Proc. ACM Symp. on Parallel Algorithms and Architectures*, 1992.

[17] G. Kant, "Drawing Planar Graphs Using the *lmc*-Ordering," *Proc. IEEE Symp. on Foundations of Computer Science*, 1992, pp. 101–110.

[18] G. Kant, G. Liotta, R. Tamassia, and I.G. Tollis, "Area Requirement of Visibility Representations of Trees," *Proc. 5th Canadian Conf. on Computational Geometry*, pp. 192-197 (1993).

[19] C.E. Leiserson, "Area-Efficient Graph Layouts (for VLSI)," *Proc. IEEE Symp. on Foundations of Computer Science*, 1980, pp. 270–281.

[20] S.M. Malitz and A. Papakostas, "On the Angular Resolution of Planar Graphs," *Proc. ACM Symp. on Theory of Computing*, 1992, pp. 527–538.

[21] J. O'Rourke, "Computational Geometry Column 18," to appear in *Int. J. on Computational Geometry and Applications*, 1992.

[22] E. Reingold and J. Tilford, "Tidier Drawing of Trees," *IEEE Trans. on Software Engineering*, vol. SE-7, 1981, pp. 223–228.

[23] K.J. Supowit and E.M. Reingold, "The Complexity of Drawing Trees Nicely," *Acta Informatica*, vol. 18, 1983, pp. 377–392.

[24] P. Rosenstiehl and R.E. Tarjan, "Rectilinear Planar Layouts of Planar Graphs and Bipolar Orientations," *Discrete & Computational Geometry*, vol. 1, 1986, pp. 343–353.

[25] W. Schnyder, "Embedding Planar Graphs on the Grid," *Proc. ACM-SIAM Symp. on Discrete Algorithms*, 1990, pp. 138–148.

[26] R. Tamassia and I.G. Tollis, "A Unified Approach to Visibility Representations of Planar Graphs," *Discrete & Computational Geometry*, vol. 1, 1986, pp. 321–341.

[27] W.T. Tutte, "How to Draw a Graph," *Proc. London Math Soc.*, vol. 3, 1963, pp. 743–768.

[28] L. Valiant, "Universality Considerations in VLSI Circuits," *IEEE Trans. on Computers*, vol. C-30, 1981, pp. 135-140.