Straight-line Drawings of General Trees with Linear Area and Arbitrary Aspect Ratio*
(Revised Version)

Ashim Garg Adrian Rusu

Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY 14260
{agarg, adirusu}@cse.buffalo.edu

Abstract

Trees are usually drawn planar, i.e. without any crossings. In this paper, we investigate the area requirement of (non-upward) planar straight-line grid drawings of general trees. A *degree-d tree* is one in which each node has at most d edges incident on it. Let T be a degree-d tree with n nodes, such that $d = O(n^\delta)$, where $0 \leq \delta < 1/2$ is a constant. We show that T admits a planar straight-line grid drawing with area $O(n)$ and with any pre-specified aspect ratio in the range $[n^{-\alpha}, n^\alpha]$, where α is a constant such that $0 \leq \alpha < 1$. We also show that such a drawing can be constructed in $O(n \log n)$ time. In particular, our result shows that optimal area (equal to $O(n)$) and optimal aspect ratio (equal to 1) is simultaneously achievable for such drawings.

1 Introduction

Trees are very common data-structures, which are used to model information in a variety of applications. such as Software Engineering (hierarchies of object-oriented programs), Business Administration (organization charts), and Web-site Design (structure of a Web-site). A *drawing* Γ of a tree T maps each node of T to a distinct point in the plane, and each edge (u, v) of T to a simple Jordan curve with endpoints u and v. Γ is a *straight-line* drawing (see Figure 1(a)), if each edge is drawn as a single line-segment. Γ is a *polyline* drawing (see Figure 1(b)), if each edge is drawn as a connected sequence of one or more line-segments, where the meeting point of consecutive line-segments is called a *bend*. Γ is an *orthogonal* drawing (see Figure 1(c)), if each edge is drawn as a chain of alternating horizontal and vertical segments. Γ is a *grid* drawing if all the nodes and edge-bends have integer coordinates. Γ is a *planar* drawing if edges do not intersect each other in the drawing (for example, all the drawings in Figure 1 are planar drawings). Γ is an *upward* drawing (see Figure 1(a,b)), if the parent is always assigned either the same or higher y-coordinate than its children. In this paper, we concentrate on grid drawings. So, we will assume that the plane is covered by a rectangular grid. Let R be a rectangle with sides parallel to the X- and Y-axes. The *width* (*height*) of R is equal to the number of grid points with the same y (x) coordinate contained within R. The *area* of R is equal to the number of grid points contained within R. The *aspect ratio* of R is the ratio of its width and height. R is the *enclosing rectangle* of Γ, if it is the smallest rectangle that covers the entire drawing. The *width*, *height*, *area*, and *aspect ratio* of Γ is equal to the width, height, area, and aspect ratio, respectively, of its enclosing rectangle. T is a binary tree if each node has at most two children. We denote by $T[v]$, the *subtree* of T rooted at a node v of T. $T[v]$ consists of v and all the descendents of v. Γ has the *subtree separation* property [1]

*Research supported by NSF CAREER Award No. IIS-9985136 and NSF CISE Research Infrastructure Award No. 0101244.

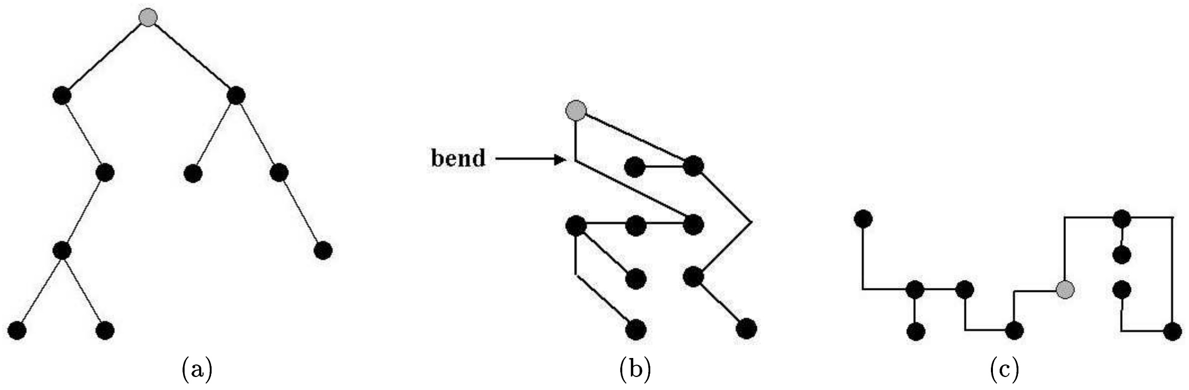

Figure 1: Various kinds of drawings of the same tree: (a) straight-line, (b) polyline, and (c) orthogonal. Also note that the drawings shown in Figures (a) and (b) are upward drawings, whereas the drawing shown in Figure (c) is not. The root of the tree is shown as a shaded circle, whereas other nodes are shown as black circles.

if, for any two node-disjoint subtrees $T[u]$ and $T[v]$ of T, the enclosing rectangles of the drawings of $T[u]$ and $T[v]$ do not overlap with each other. Drawings with subtree separation property are more aesthetically pleasing than those without subtree separation property. The subtree separation property also allows for a focus+context style [8] rendering of the drawing, so that if the tree has too many nodes to fit in the given drawing area, then the subtrees closer to focus can be shown in detail, whereas those further away from the focus can be contracted and simply shown as filled-in rectangles.

2 Our Result

Planar straight-line drawings are more aesthetically pleasing than non-planar polyline drawings. Grid drawings guarantee at least unit distance separation between the nodes of the tree, and the integer coordinates of the nodes and edge-bends allow the drawings to be displayed in a display surface, such as a computer screen, without any distortions due to truncation and rounding-off errors. Giving users control over the aspect ratio of a drawing allows them to display the drawing in different kinds of display surfaces with different aspect ratios. The subtree separation property makes it easier for the user to detect the subtrees in the drawing, and also allows for a focus+context style [8] rendering of the drawing. Finally, it is important to minimize the area of a drawing, so that the users can display a tree in as small drawing area as possible.

We, therefore, investigate the problem of constructing (non-upward) planar straight-line grid drawings of general trees with small area. Clearly, any planar grid drawing of a tree with n nodes requires $\Omega(n)$ area. A long-standing fundamental question, therefore, has been that whether this is a tight bound also, i.e., given a tree T with n nodes, can we construct a planar straight-line grid drawing of T with area $O(n)$?

In this paper, we partially answer this question in affirmative, by giving an algorithm that constructs a planar straight-line grid drawing of a degree-d tree with n nodes, where $d = O(n^\delta)$ is a positive integer and $0 \leq \delta < 1/2$ is a constant, with $O(n)$ area in $O(n \log n)$ time. Moreover, the drawing can be parameterized for its aspect ratio, i.e., for any constant α, where $0 \leq \alpha < 1$, the algorithm can construct a drawing with any user-specified aspect ratio in the range $[n^{-\alpha}, n^\alpha]$. Theorem 2 summarizes our overall result. In particular, our result shows that optimal area (equal to $O(n)$) and optimal aspect ratio (equal to 1) is simultaneously achievable (see Corollary 1). It is also interesting to note that the drawings constructed by our algorithm also exhibit the subtree separation property.

3 Previous Results

Most of the research on drawing trees has dealt with binary trees [4]. Previously, the best-known upper bound on the area of a planar straight-line grid drawing of an n-node binary tree was $O(n \log \log n)$, which was shown in [1] and [9]. This bound is very close to $O(n)$, but still it does not settle the question whether an n-

node tree can be drawn in this fashion in *optimal* $O(n)$ area. Thus, our result is significant from a theoretical view-point. In fact, we already know of one category of drawings, namely, planar upward orthogonal polyline grid drawings, for which $n \log \log n$ is a tight bound [5], i.e., any binary tree can be drawn in this fashion in $O(n \log \log n)$ area, and there exists a family of binary trees that requires $\Omega(n \log \log n)$ area in any such drawing. So, a natural question arises, if $n \log \log n$ is a tight bound for planar straight-line grid drawings also. Of course, our result implies that this is not the case. Besides, our drawing technique and proofs are significantly different from those of [1] and [9]. Moreover, the drawing constructed by the algorithms of [1] and [9] has a fixed aspect ratio, equal to $\theta(\log^2 n/(n \log \log n))$, whereas the aspect ratio of the drawing constructed by our algorithm can be specified by the user.

We now summarize some other known results on planar grid drawings of binary trees (for more results, see [4]). Let T be an n-node binary tree. [5] presents an algorithm for constructing an upward polyline drawing of T with $O(n)$ area, and any user-specified aspect ratio in the range $[n^{-\alpha}, n^{\alpha}]$, where α is any constant, such that $0 \leq \alpha < 1$. [7] and [11] present algorithms for constructing a (non-upward) orthogonal polyline drawing of T with $O(n)$ area. [1] gives an algorithm for constructing an upward orthogonal straight-line drawing of T with $O(n \log n)$ area, and any user-specified aspect ratio in the range $[\log n/n, n/\log n]$. It also shows that $n \log n$ is also a tight bound for such drawings. [9] gives an algorithm for constructing an upward straight-line drawing of T with $O(n \log \log n)$ area. If T is a Fibonacci tree, (AVL tree, complete binary tree), then [2, 10] ([3], [2], respectively) give algorithms for constructing an upward straight-line drawing of T with $O(n)$ area.

Table 1 summarizes these results.

Tree Type	Drawing Type	Area	Aspect Ratio	Reference
Fibonacci	Upward Straight-line	$O(n)$	$\theta(1)$	[2, 10]
AVL	Upward Straight-line	$O(n)$	$\theta(1)$	[3]
Complete Binary	Upward Straight-line	$O(n)$	$\theta(1)$	[2]
General Binary	Upward Orthogonal Polyline	$O(n \log \log n)$	$\theta(\log^2 n/(n \log \log n))$	[5, 9]
	(Non-upward) Orthogonal Polyline	$O(n)$	$\theta(1)$	[7, 11]
	Upward Orthogonal Straight-line	$O(n \log n)$	$[\log n/n, n/\log n]$	[1]
	Upward Polyline	$O(n)$	$[n^{-\alpha}, n^{\alpha}]$	[5]
	Upward Straight-line	$O(n \log \log n)$	$\theta(\log^2 n/(n \log \log n))$	[9]
	(Non-upward) Straight-line	$O(n \log \log n)$	$\theta(\log^2 n/(n \log \log n))$	[1]
		$O(n)$	$[n^{-\alpha}, n^{\alpha}]$	[6]
Degree-$O(n^{\delta})$, where $0 \leq \delta < 1/2$	(Non-upward) Straight-line	$O(n)$	$[n^{-\alpha}, n^{\alpha}]$	*this paper*

Table 1: Bounds on the areas and aspect ratios of various kinds of planar grid drawings of an n-node tree. Here, α is a constant, such that $0 \leq \alpha < 1$.

4 Preliminaries

Throughout this paper, by the term *drawing*, we will mean a planar straight-line grid drawing. We will assume that the plane is covered by an infinite rectangular grid. A *horizontal channel* (*vertical channel*) is an infinite line parallel to X- (Y-) axis, passing through the grid-points.

Let T be a degree-d tree, with one distinguished node v, which has at most $d-1$ children. v is called the *link* node of T. Let n be the number of nodes in T. T is an *ordered* tree if the children of each node are assigned a left-to-right order. A *partial tree* of T is a connected subgraph of T. If T is an ordered tree, then the *leftmost path* p of T is the maximal path consisting of nodes that are leftmost children, except the first one, which is the root of T. The last node of p is called the *leftmost* node of T. Two nodes of T are *siblings* if they have the same parent in T. T is an *empty tree*, i.e., $T = \phi$, if it has zero nodes in it.

Let Γ be a drawing of T. By *bottom* (*top*, *left*, and *right*, respectively) boundary of Γ, we will mean the *bottom* (*top*, *left*, and *right*, respectively) boundary of the enclosing rectangle $R(\Gamma)$ of Γ. Similarly, by *top-left* (*top-right*, *bottom-left*, and *bottom-right*, respectively) corner of Γ, we mean the *top-left* (*top-right*, *bottom-left*, and *bottom-right*, respectively) corner of $R(\Gamma)$.

Let R be a rectangle, such that Γ is entirely contained within R. R has a *good* aspect ratio, if its aspect ratio is in the range $[n^{-\alpha}, n^{\alpha}]$, where $0 \le \alpha < 1$ is a constant.

Let r be the root of T. Let u^* be the link node of T. Γ is a *feasible* drawing of T, if it has the following three properties:

- **Property 1**: The root r is placed at the top-left corner of Γ.
- **Property 2**: If $u^* \neq r$, then u^* is placed at the bottom boundary of Γ. Moreover, we can move u^* downwards in its vertical channel by any distance without causing any edge-crossings in Γ.
- **Property 3**: If $u^* = r$, then no other node or edge of T is placed on, or crosses the vertical and horizontal channels occupied by r.

Theorem 1 *In any degree-d tree T, there is a node u, such that removing u and its incident edges splits T into at most d trees, where each tree has at most $n/2$ nodes in it, and $n \ge 2$ is the number of nodes in T. Node u is called a* separator node *of T. Moreover, u can be found in $O(n)$ time.*

Proof: To obtain u, we invoke Algorithm *FindSeparator* with the root of T as its input. Also, as a pre-process step, before invoking Algorithm *FindSeparator*, we first compute the number of nodes $n(v)$ in the subtree rooted at each node v of T. We store the value of $n(v)$ in each node v. We now describe Algorithm *FindSeparator*.

```
Algorithm FindSeparator($u$): {$u$ is a node of $T$}
1. Let $s_1, \ldots, s_j$ be the children of $u$.
2. Let $s_i$ be the child of $u$ such that $n(s_i) = \max\{n(s_1), \ldots, n(s_j)\}$.
3. If $n(s_i) > n/2$, then Return FindSeparator($s_i$).
4. Else Return $u$.
```

The algorithm clearly runs in $O(n)$ time.

Let u be the node of T returned on calling Algorithm *FindSeparator*(r), where r is the root of T. We will prove that u is a separator node of T. It is easy to see from the description of the algorithm that $n(u) > n/2$ and $n(s) \le n/2$, for each child s of u. Let T' be the partial tree of T obtained by removing the subtree rooted at u from T. Since $n(u) > n/2$, the number of nodes in T' is less than $n - n/2 = n/2$. Also recall that $n(s) \le n/2$, for each child s of u. Hence, removing u and its incident edges will split T into at most d trees, each containing at most $n/2$ nodes. Hence, u is indeed a separator node of T. $\square$

Let v be a node of tree T located at grid point (i, j) in Γ. Let Γ be a drawing of T. Assume that the root r of T is located at the grid point $(0, 0)$ in Γ. We define the following operations on Γ (see Figure 2):

- *rotate operation*: rotate Γ counterclockwise by δ degrees around the z-axis passing through r. After a rotation by δ degrees of Γ, node v will get relocated to the point $(i \cos\delta - j \sin\delta, i \sin\delta + j \cos\delta)$. In particular, after rotating Γ by $90°$, node v will get relocated to the grid point $(-j, i)$.
- *flip operation*: flip Γ vertically or horizontally. After a horizontal flip of Γ, node v will be located at grid point $(-i, j)$. After a vertical flip of Γ, node v will be located at grid point $(i, -j)$.

5 Our Tree Drawing Algorithm

Let T be a degree-d tree with a link node u^*, where $d = O(n^{\delta})$ is a positive integer, $0 \le \delta < 1/2$ is a constant, and n is the number of nodes in T. Let A and ϵ be two numbers such that $\delta/(1-\delta) < \epsilon < 1$, and A is in the range $[n^{-\epsilon}, n^{\epsilon}]$. A is called the *desirable aspect ratio* for T.

Our tree drawing algorithm, called *DrawTree*, takes ϵ, A, and T as input, and uses a simple divide-and-conquer strategy to recursively construct a feasible drawing Γ of T, by performing the following actions at each recursive step (as we will prove later, Γ will fit inside a rectangle with area $O(n)$ and aspect ratio A):

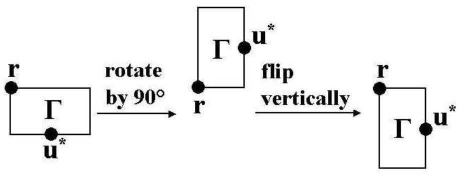

Figure 2: Rotating a drawing Γ by 90°, followed by flipping it vertically. Note that initially node u^* was located at the bottom boundary of Γ, but after the rotate operation, u^* is on the right boundary of Γ.

- *Split Tree*: Split T into at most $2d-1$ partial trees by removing at most two nodes and their incident edges from it. Each partial tree has at most $n/2$ nodes. Based on the arrangement of these partial trees within T, we get two cases, which are shown in Figures 3 and 4, and described later in Section 5.1.
- *Assign Aspect Ratios*: Correspondingly, assign a desirable aspect ratio A_k to each partial tree T_k. The value of A_k is based on the value of A, and the number of nodes in T_k.
- *Draw Partial Trees*: Recursively construct a feasible drawing of each partial tree T_k with A_k as its desirable aspect ratio.
- *Compose Drawings*: Arrange the drawings of the partial trees, and draw the nodes and edges, that were removed from T to split it, such that the drawing Γ of T thus obtained is a feasible drawing. Note that the arrangement of these drawings is done based on the cases shown in Figures 3 and 4. In each case, if $A < 1$, then the drawings of the partial trees are stacked one above the other, and if $A \geq 1$, then they are placed side-by-side.

Figure 5 shows a drawing of a complete binary tree with 63 nodes constructed by Algorithm *DrawTree*, with $A = 1$ and $\epsilon = 0.2$.

We now give the details of each action performed by Algorithm *DrawTree*:

5.1 Split Tree

The splitting of tree T into partial trees is done as follows:

- Order the children of each node such that u^* becomes the leftmost node of T.
- Using Theorem 1, find a separator node u of T.
- Based on whether, or not, u is in the leftmost path of T, we get two cases:
  - *Case 1: The separator node u is not in the leftmost path of T.* We get seven subcases:
    (a) In the general case, T has the form as shown in Figure 3(a). In this figure:
      * r is the root of T,
      * $c_1, \ldots, c_j$ are the children of u, $0 \leq j \leq d-1$,
      * $T_1, \ldots, T_j$ are the trees rooted at $c_1, \ldots, c_j$ respectively, $0 \leq j \leq d-1$,
      * T_α is the subtree rooted at u,
      * w is the parent of u,
      * a is the last common node of the path $r \rightsquigarrow v$ and the leftmost path of T,
      * f is the child of a that is contained in the path $r \rightsquigarrow v$,
      * T_β is the maximal tree rooted at f that contains w but not u,
      * T_B is the tree consisting of the trees T_α and T_β, and the edge (w, u),
      * e is the parent of a,
      * g is the leftmost child of a,

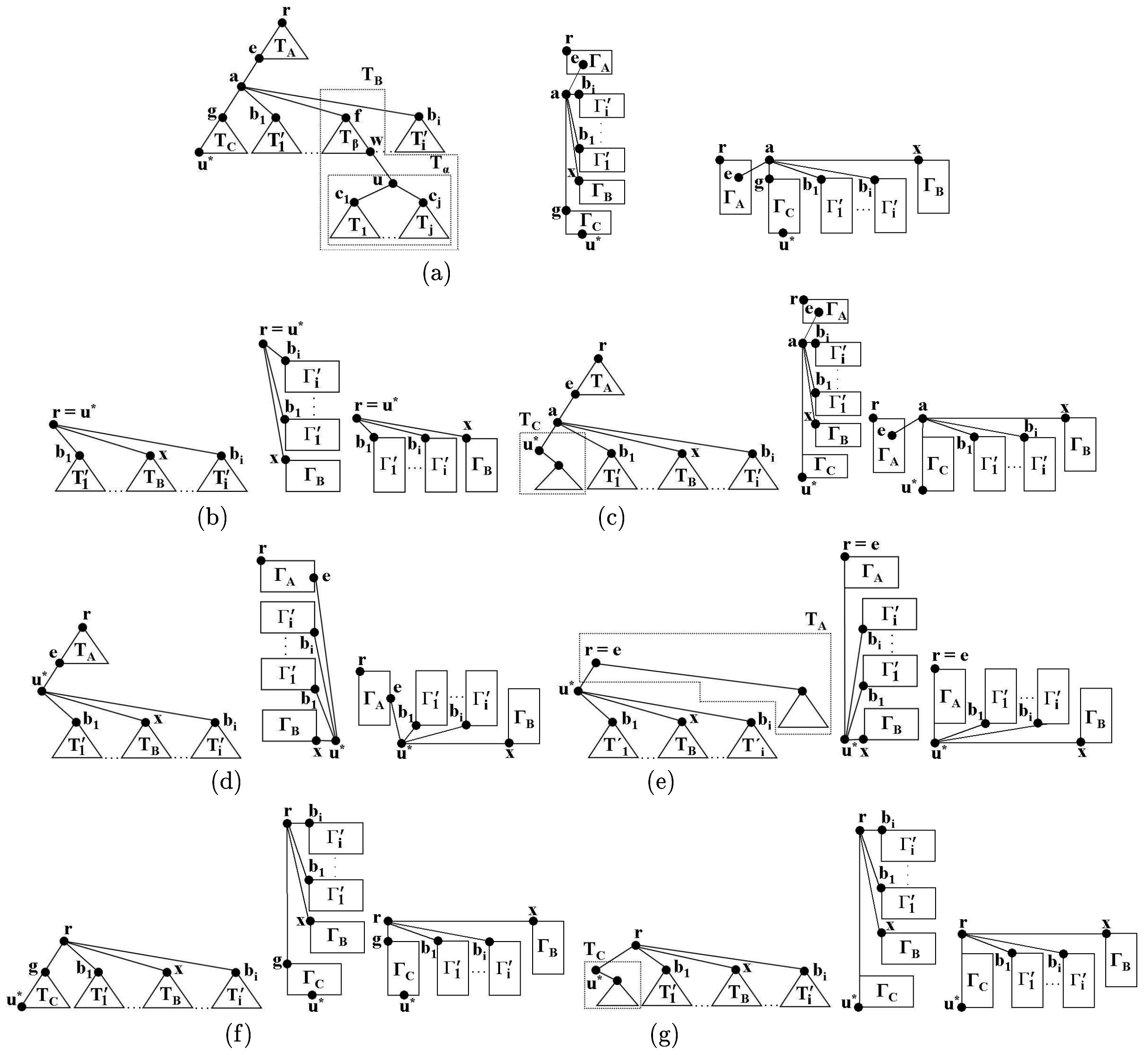
Figure 3: Drawing T in all the seven subcases of Case 1 (when the separator node u is not in the leftmost path of T): (a) $T_A \neq \emptyset$, $T_C \neq \emptyset$, $g \neq u^*$, $0 \leq i \leq d-3$, (b) $T_A = \emptyset$, $T_C = \emptyset$, $0 \leq i \leq d-3$, (c) $T_A \neq \emptyset$, $T_C \neq \emptyset$, $g = u^*$, $0 \leq i \leq d-3$, (d) $T_A \neq \emptyset$, $T_C = \emptyset$, $r \neq e$, $0 \leq i \leq d-3$, (e) $T_A \neq \emptyset$, $T_C = \emptyset$, $r = e$, $0 \leq i \leq d-3$, (f) $T_A = \emptyset$, $T_C \neq \emptyset$, $g \neq u^*$, $0 \leq i \leq d-3$, and (g) $T_A = \emptyset$, $T_C \neq \emptyset$, $g = u^*$, $0 \leq i \leq d-3$. For each subcase, we first show the structure of T for that subcase, then its drawing when $A < 1$, and then its drawing when $A \geq 1$. Here, x is the same as f if $T_\beta \neq \phi$, and is the same as the root of T_α if $T_\beta = \phi$. In Subcases (a) and (c), for simplicity, e is shown to be in the interior of Γ_A, but actually, either it is the same as r, or if $A < 1$ ($A \geq 1$), then it is placed on the bottom (right) boundary of Γ_A. For simplicity, we have shown Γ_A, Γ_B, and Γ_C as identically sized boxes, but in actuality, they may have different sizes.

* T_A is the maximal tree rooted at r that contains e but not a,
* T_C is the tree rooted at g,
* $b_1, \ldots, b_i$ are the siblings of f and g,
* $T'_1, \ldots, T'_i$ are the trees rooted at $b_1, \ldots, b_i$ respectively, $0 \leq i \leq d-3$, and
* $g \neq u^*$.

In addition to this general case, we get six special cases: (b) $T_A = \emptyset$, $T_C = \emptyset$, $0 \leq i \leq d-3$

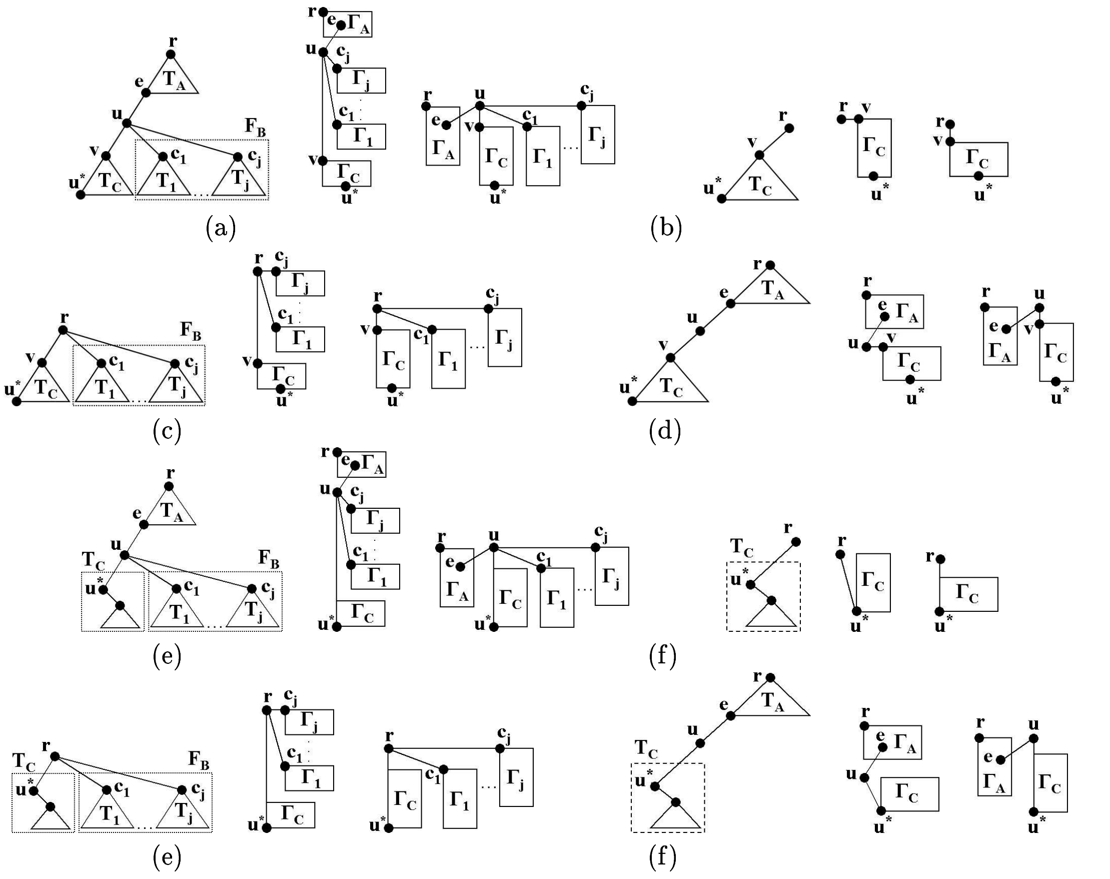

Figure 4: Drawing T in all the eight subcases of Case 2 (when the separator node u is in the leftmost path of T): (a) $T_A \neq \emptyset, T_C \neq \emptyset$, $v \neq u^*$, $1 \leq j \leq d-2$, (b) $T_A = \emptyset, T_C \neq \emptyset$, $v \neq u^*$, $j = 0$, (c) $T_A = \emptyset, T_C \neq \emptyset$, $v \neq u^*$, $1 \leq j \leq d-2$, (d) $T_A \neq \emptyset, T_C \neq \emptyset$, $v \neq u^*$, $j = 0$, (e) $T_A \neq \emptyset$, $T_B \neq \emptyset$, $v = u^*$, $1 \leq j \leq d-2$, (f) $T_A = \emptyset$, $T_B = \emptyset$, $v = u^*$, $j = 0$, (g) $T_A = \emptyset$, $T_B \neq \emptyset$, $v = u^*$, $1 \leq j \leq d-2$, and (h) $T_A \neq \emptyset$, $T_B = \emptyset$, $v = u^*$, $j = 0$. For each subcase, we first show the structure of T for that subcase, then its drawing when $A < 1$, and then its drawing when $A \geq 1$. In Subcases (a) and (d), for simplicity, e is shown to be in the interior of Γ_A, but actually, either it is same as r, or if $A < 1$ ($A \geq 1$), then it is placed on the bottom (right) boundary of Γ_A. For simplicity, we have shown Γ_A, Γ_B, and Γ_C as identically sized boxes, but in actuality, they may have different sizes.

(see Figure 3(b)), (c) $T_A \neq \emptyset$, $T_C \neq \emptyset$, $g = u^*$, $0 \leq i \leq d-3$ (see Figure 3(c)), (d) $T_A \neq \emptyset$, $T_C = \emptyset$, $r \neq e$, $0 \leq i \leq d-3$ (see Figure 3(d)), (e) $T_A \neq \emptyset$, $T_C = \emptyset$, $r = e$, $0 \leq i \leq d-3$ (see Figure 3(e)), (f) $T_A = \emptyset$, $T_C \neq \emptyset$, $g \neq u^*$, $0 \leq i \leq d-3$ (see Figure 3(f)), and (g) $T_A = \emptyset$, $T_C \neq \emptyset$, $g = u^*$, $0 \leq i \leq d-3$ (see Figure 3(g)). (The reason we get these seven subcases is as follows: T_α has at least $n/2$ nodes in it because of Theorem 1. Hence $T_\alpha \neq \phi$, and so, $T_B \neq \phi$. Based on whether $T_A = \phi$ or not, $T_C = \phi$ or not, $g = u^*$ or not, and $r = e$ or not, we get totally sixteen cases. From these sixteen cases, we obtain the above seven subcases, by grouping some of these cases together. For example, the cases $T_A = \phi$, $T_C = \phi$, $d \neq u^*$, $r = u^*$, and $T_A = \phi$, $T_C = \phi$, $d \neq u^*$, $r \neq u^*$ are grouped together to give Case (a), i.e., $T_A = \phi$, $T_C = \phi$, $d \neq u^*$. So, Case (a) corresponds to 2 cases. Similarly, Cases (c), (d), (e), (f), and (g) correspond to 2 cases each, and Case (b) corresponds to 4 cases.) In each case, we remove nodes a and u, and their incident edges, to split T into at most $2d-1$ partial trees T_A, T_C, T_β, $T'_1, \ldots, T'_i$, $0 \leq i \leq d-3$, and $T_1, \ldots, T_j$, $0 \leq j \leq d-1$. We also designate e as the link node of T_A, w as the link node of T_β, and u^* as the link node of T_C. We arbitrarily select a leaf e_i of T'_i, $0 \leq i \leq d-3$, and a leaf e_j of T_j, $0 \leq j \leq d-1$, and designate them as the link nodes of T'_i and T_j, respectively.

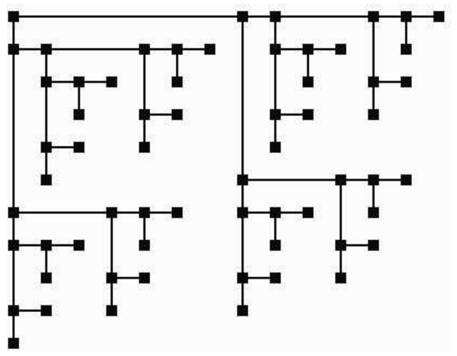

Figure 5: Drawing of the complete binary tree with 63 nodes constructed by Algorithm *DrawTree*, with $A = 1$ and $\epsilon = 0.2$.

- *Case 2: The separator node u is in the leftmost path of T.* We get eight subcases: (a) In the general case, T has the form as shown in Figure 4(a). In this figure,
  - $*$ r is the root of T,
  - $*$ v is the leftmost child of u,
  - $*$ $c_1, \ldots, c_j$ are the siblings of v, $1 \leq j \leq d-2$,
  - $*$ $T_1, \ldots, T_j$ are the trees rooted at $c_1, \ldots, c_j$ respectively, $1 \leq j \leq d-2$,
  - $*$ e is the parent of u,
  - $*$ T_A is the maximal tree rooted at r that contains e but not u,
  - $*$ T_C is the subtree of T rooted at v,
  - $*$ F_B is the forest composed by trees $T_1, \ldots, T_j$, $1 \leq j \leq d-2$, and
  - $*$ $v \neq u^*$.

  In addition to the general case, we get the following seven special cases: (b) $T_A = \emptyset$, $j = 0$, $v \neq u^*$ (see Figure 4(b)), (c) $T_A = \emptyset$, $1 \leq j \leq d-2$, $v \neq u^*$ (see Figure 4(c)), (d) $T_A \neq \emptyset$, $j = 0$, $v \neq u^*$ (see Figure 4(d)), (e) $T_A \neq \emptyset$, $1 \leq j \leq d-2$, $v = u^*$ (see Figure 4(e)), (f) $T_A = \emptyset$, $j = 0$, $v = u^*$ (see Figure 4(f)), (g) $T_A = \emptyset$, $1 \leq j \leq d-2$, $v = u^*$ (see Figure 4(g)), and (h) $T_A \neq \emptyset$, $j = 0$, $v = u^*$ (see Figure 4(h)). (The reason we get these eight subcases is as follows: T_C has at least $n/2$ nodes in it because of Theorem 1. Hence, $T_C \neq \phi$. Based on whether $T_A = \phi$ or not, $F_B = \phi$ or not, and $v = u^*$ or not, we get the eight subcases given above.) In each case, we remove node u, and its incident edges, to split T into at most d partial trees T_A, T_C, and $T_1, \ldots, T_j$, $0 \leq j \leq d-2$. We also designate e as the link node of T_A, and u^* as the link node of T_C. We randomly select a leaf e_j of T_j and designate it as the link node of T_j, $0 \leq j \leq d-2$.

5.2 Assign Aspect Ratios

Let T_k be a partial tree of T, where for Case 1, T_k is either T_A, T_C, T_β, $T'_1, \ldots, T'_i$, $0 \leq i \leq d-3$, or $T_1, \ldots, T_j$, $0 \leq j \leq d-1$, and for Case 2, T_k is either T_A, T_C, or $T_1, \ldots, T_j$, $0 \leq j \leq d-2$. Let n_k be number of nodes in T_k.

Definition: T_k is a *large* partial tree of T if:

- $A \geq 1$ and $n_k \geq (n/A)^{1/(1+\epsilon)}$, or
- $A < 1$ and $n_k \geq (An)^{1/(1+\epsilon)}$,

and is a *small* partial tree of T otherwise.

In Step *Assign Aspect Ratios*, we assign a desirable aspect ratio A_k to each nonempty T_k as follows: Let $x_k = n_k/n$.

- If $A \geq 1$: If T_k is a large partial tree of T, then $A_k = x_k A$, otherwise (i.e., if T_k is a small partial tree of T) $A_k = n_k^{-\epsilon}$.
- If $A < 1$: If T_k is a large partial tree of T, then $A_k = A/x_k$, otherwise (i.e., if T_k is a small partial tree of T) $A_k = n_k^{\epsilon}$.

Intuitively, this assignment strategy ensures that each partial tree gets a good desirable aspect ratio, and so, the drawing of each partial tree constructed recursively by Algorithm *DrawTree* will fit inside a rectangle with linear area and good aspect ratio.

5.3 Draw Partial Trees

First, we change the desirable aspect ratios assigned to T_A and T_β in some cases as follows: Suppose T_A and T_β get assigned desirable aspect ratios equal to m and p, respectively, where m and p are some positive numbers. In Subcase (d) of Case 1, and if $A \geq 1$, then in Subcases (a) and (c) of Case 1, and Subcases (a), (d), (e), and (h) of Case 2, we change the value of the desirable aspect ratio of T_A to $1/m$. In Case 1, if $A \geq 1$, we change the value of the desirable aspect ratio of T_β to $1/p$. We make these changes because, as explained later in Section 5.4, in these cases, we need to rotate the drawings of T_A and T_β by 90° during the *Compose Drawings* step. Drawing T_A and T_β with desirable aspect ratios $1/m$ and $1/p$, respectively, compensates for this rotation, and ensures that the drawings of T_A and T_β used to draw T have the desirable aspect ratios, m and p, respectively.

Next we draw recursively each nonempty partial tree T_k with A_k as its desirable aspect ratio, where the value of A_k is the one computed in the previous step. The base case for the recursion happens when T_k contains exactly one node, in which case, the drawing of T_k is simply the one consisting of exactly one node.

5.4 Compose Drawings

Let Γ_k denote the drawing of a partial tree T_k constructed in Step *Draw Partial Trees*. We now describe the construction of a feasible drawing Γ of T from the drawings of its partial trees in both Cases 1 and 2.

In Case 1, we first construct a feasible drawing Γ_α of the partial tree T_α by composing $\Gamma_1, \ldots, \Gamma_j$, $0 \leq j \leq d-1$, as shown in Figure 6, then construct a feasible drawing Γ_B of T_B by composing Γ_α and Γ_β as shown in Figure 7, and finally construct Γ by composing Γ_A, Γ_B, Γ_C, $\Gamma'_1, \ldots, \Gamma'_i$, $0 \leq i \leq d-3$, as shown in Figure 3.

Γ_α is constructed as follows (see Figure 6): If $A < 1$, place $\Gamma_j, \ldots, \Gamma_2, \Gamma_1$, $1 \leq j \leq d-1$, one above the other, in this order, separated by unit vertical distance, such that the left boundaries of $\Gamma_j, \ldots, \Gamma_2$ are aligned, and one unit to the right of the left boundary of Γ_1. Place u in the same vertical channel as c_1 and in the same horizontal channel as c_j. If $A \geq 1$, place $\Gamma_1, \Gamma_2, \ldots, \Gamma_j$, $1 \leq j \leq d-1$ in a left-to-right order, separated by unit horizontal distance, such that the top boundaries of $\Gamma_1, \Gamma_2, \ldots, \Gamma_{j-1}$ are aligned, and one unit below the top boundary of Γ_j. Place u in the same vertical channel as c_1 and in the same horizontal channel as c_j.

Γ_B is constructed as follows (see Figure 7):

- if $T_\beta \neq \emptyset$ (see Figure 7(a)) then, if $A < 1$, then place Γ_β one unit above Γ_α such that the left boundaries of Γ_β and Γ_α are aligned; otherwise (i.e., if $A \geq 1$), first rotate Γ_β by 90° and then flip it vertically, then place Γ_β one unit to the left of Γ_α such that the top boundaries of Γ_β and Γ_α are aligned. Draw edge (w, y).
- Otherwise (i.e., if $T_\beta = \emptyset$), Γ_B is same as Γ_α (see Figure 7(b)).

Γ is constructed from Γ_A, Γ_B, Γ_C, $\Gamma'_1, \ldots, \Gamma'_i$, $0 \leq i \leq d-3$, as follows (see Figure 3): Let x be the root of T_B. Note that $x = f$ if $T_\beta \neq \emptyset$, and $x = u$ otherwise.

- In Subcase (a), as shown in Figure 3(a), if $A < 1$, stack Γ_A, $\Gamma'_i, \ldots, \Gamma'_1$, Γ_B, Γ_C one above the other, in this order, such that they are separated by unit vertical distance from each other, and the left boundaries of $\Gamma'_{i-1}, \ldots, \Gamma'_1, \Gamma_B$ are aligned with each other and are placed at unit horizontal distance to the right of the left boundaries of Γ_A and Γ_C. Place node a in the same vertical channel as r and g and in the same horizontal channel as b_i. If $A \geq 1$, then first rotate Γ_A by 90°, and then flip it vertically. Then, place Γ_A, Γ_C, $\Gamma'_1, \ldots, \Gamma'_i$, Γ_B from left-to-right in this order, separated by unit horizontal distances, such that the top boundaries of Γ_C, $\Gamma'_1, \ldots, \Gamma'_i$, are aligned, and are at unit vertical distance below the top boundaries of Γ_A and Γ_B. Then, move Γ_C down until u^* becomes the lowest node of Γ. Place node a in the same vertical channel as g and in the same horizontal channel as r and x. Draw edges (a, e), (a, x), (a, g), $(a, b_1), \ldots, (a, b_i)$.

- In Subcase (b), as shown in Figure 3(b), if $A < 1$, stack $\Gamma'_i, \ldots, \Gamma'_1$, Γ_B, one above the other, such that they are separated by unit vertical distance from each other, and their left boundaries are aligned. Place node r one unit above and left of the top boundary of Γ'_i. If $A \geq 1$, place $\Gamma'_1, \ldots, \Gamma'_i$, Γ_B in a left-to-right order such that they are separated by unit horizontal distance from each other, and their top boundaries are aligned. Place node r one unit above and left of the top boundary of Γ'_1. Draw edges $(r, b_1), \ldots, (r, b_i)$, (r, x).
- The drawing procedure for Subcase (c) is similar to the one in Subcase (a), except that we also flip Γ_C vertically (see Figure 3(c)).
- In Subcase (d), as shown in Figure 3(d), if $A < 1$, flip $\Gamma'_i, \ldots, \Gamma'_1$, Γ_B first vertically, and then horizontally, so that their roots get placed at their lower-right corners. Then, first rotate Γ_A by 90°, and then flip it vertically. Next, place Γ_A, $\Gamma'_i, \ldots, \Gamma'_1$, Γ_B one above the other, in this order, with unit vertical separation, such that their left boundaries are aligned, next move node e (which is the link node of T_A) to the right until it is either to the right of, or aligned with the rightmost boundary among $\Gamma'_i, \ldots, \Gamma'_1$, Γ_B (since Γ_A is a feasible drawing, by Property 2, as given in Section 4, moving e will not create any edge-crossings), and then place u^* in the same horizontal channel as x and one unit to the right of e. If $A \geq 1$, first rotate Γ_A by 90°, and then flip it vertically. Then flip $\Gamma'_i, \ldots, \Gamma'_1, \Gamma_B$ vertically. Then, place Γ_A, u^*, $\Gamma'_1, \ldots, \Gamma'_i$, Γ_B left-to-right in this order, separated by unit horizontal distances, such that the bottom boundaries of $\Gamma'_1, \ldots, \Gamma'_i$, are aligned, and are at unit vertical distance above the bottom boundary of Γ_B. Move Γ_B down until its bottom boundary is either aligned with or below the bottom boundary of Γ_A. Also, u^* is in the same horizontal channel with x. Draw edges (u^*, e), $(u^*, b_1), \ldots, (u^*, b_i)$, (u^*, x).
- In Subcase (e), as shown in Figure 3(e), if $A < 1$, first flip $\Gamma'_i, \ldots, \Gamma'_1$, Γ_B, vertically, then place Γ_A, $\Gamma'_i, \ldots, \Gamma'_1$, Γ_B one above the other, in this order, with unit vertical separation, such that the left boundaries of $\Gamma'_i, \ldots, \Gamma'_1$, Γ_B are aligned, and the left boundary of Γ_A is at unit horizontal distance to the left of the left boundary of Γ_B. Place u^* in the same vertical channel with r and in the same horizontal channel with x . If $A \geq 1$, then first flip $\Gamma'_1, \ldots, \Gamma'_i$, Γ_B vertically, next place Γ_A, $\Gamma'_1, \ldots, \Gamma'_i$, Γ_B in a left-to-right order at unit horizontal distance, such that the top boundaries Γ_A, $\Gamma'_1, \ldots, \Gamma'_i$ are aligned, and the bottom boundary of Γ_B is one unit below the bottom boundary of the drawing among Γ_A, $\Gamma'_1, \ldots, \Gamma'_i$ with greater height. Then, place u^* in the same vertical channel as r and in the same horizontal channel as r. Draw edges (u^*, r), $(u^*, b_1), \ldots, (u^*, b_i)$, (u^*, x). Note that, since Γ_A is a feasible drawing, by Property 3 (see Section 4), drawing (u^*, r) will not create any edge-crossings.
- The drawing procedure in Subcase (f) is similar to the one in Subcase (a), except that we do not have Γ_A here (see Figure 3(f)).
- The drawing procedure in Subcase (g) is similar to the one in Subcase (f), except that we also flip Γ_C vertically (see Figure 3(g).

In Case 2, we construct Γ by composing Γ_A, $\Gamma_1, \ldots, \Gamma_j$, Γ_C as follows (see Figure 4):

- The drawing procedures in Subcases (a) and (c) are similar to those in Subcases (a) and (f), respectively, of Case 1 (see Figures 4(a,c)).
- In Subcase (b) as shown in Figure 6(b), if $A < 1$, place u in the same horizontal channel and at one unit to the left of v; otherwise (i.e. $A \geq 1$), place u in the same vertical channel and at one unit above v. Draw edge (r, v).
- In Subcase (d), as shown in Figure 4(d), if $A > 1$, we place Γ_A above Γ_C, separated by unit vertical distance such that the left boundary of Γ_C is one unit to the right of the left boundary of Γ_A. Place u in the same vertical channel as r and in the same horizontal channel as v. If $A \geq 1$, then first rotate Γ_A by 90°, and then flip it vertically. Then, place Γ_A to the left of Γ_C, separated by unit horizontal distance, such that the top boundary of Γ_C is one unit below the top boundary of Γ_A. Then, move

Γ_C down until u^* becomes the lowest node of Γ. Place u in the same vertical channel as v and in the same horizontal channel as r. Draw edges (u, v) and (u, e).

- The drawing procedures in Subcases (e), (f), (g), and (h) are similar to those in Subcases (a), (b), (c), and (d), respectively, (see Figures 4(e,f,g,h)), except that we also flip Γ_C vertically.

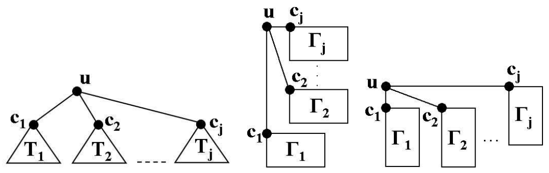

Figure 6: Drawing T_α. Here, we first show the structure of T_α, then its drawing when $A < 1$, and then its drawing when $A \geq 1$. For simplicity, we have shown $\Gamma_1, \dots, \Gamma_j$ as identically sized boxes, but in actuality, their sizes may be different.

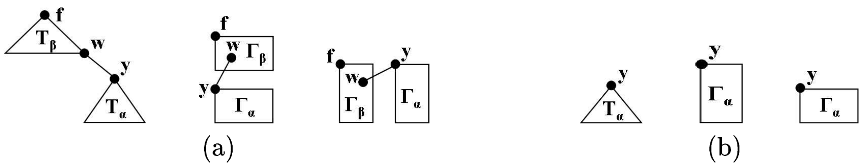

Figure 7: Drawing T_B when: (a) $T_\beta \neq \emptyset$, and (b) $T_\beta = \emptyset$. For each case, we first show the structure of T_B for that case, then its drawing when $A < 1$, and then its drawing when $A \geq 1$. In Case (a), for simplicity, w is shown to be in the interior of Γ_β, but actually, it is either same as f, or if $A < 1$ ($A \geq 1$), then is placed on the bottom (right) boundary of Γ_β. For simplicity, we have shown Γ_β and Γ_α as identically sized boxes, but in actuality, their sizes may be different.

5.5 Proof of Correctness

Lemma 1 (Planarity) *Given an n-node degree-d tree T, where $d = O(n^\delta)$ is a positive integer and $0 \leq \delta < 1$, with a link node u^*, Algorithm* DrawTree *will construct a feasible drawing Γ of T.*

Proof: We can easily prove using induction over the number of nodes n in T that Γ is a feasible drawing:
Base Case ($n = 1$): Γ consists of exactly one node and is trivially a feasible drawing.
Induction ($n > 1$): Consider Case 1. By the inductive hypothesis, the drawing constructed of each partial tree of T is a feasible drawing.

Hence, from Figure 6, it can be easily seen that the drawing Γ_α of T_α is also a feasible drawing.

From Figure 7, it can be easily seen that the drawing Γ_B of T_B is also a feasible drawing. Note that because Γ_β is a feasible drawing of T_β and w is its link node, w is either at the bottom of Γ_β (from Property 2, see Section 4), or at the top-left corner of Γ_β and no other edge or node of T_β is placed on, or crosses the vertical channel occupied by it (Properties 1 and 3, see Section 4). Hence, in Figure 7(a), in the case $A < 1$, drawing edge (w, x) will not cause any edge crossings. Also, in Figure 7(a), in the case $A \geq 1$, drawing edge (w, x) will not cause any edge crossings because after rotating Γ_β by 90° and flipping it vertically, w will either be at the right boundary of Γ_β (see Property 2), or at the top-left corner of Γ_β and no other edge or node of T_β will be placed on, or cross the horizontal channel occupied by it (see Properties 1 and 3).

Finally, by considering each of the seven subcases shown in Figure 3 one-by-one, we can show that Γ is also a feasible drawing of T:

- *Subcase (a):* See Figure 3(a). Γ_A is a feasible drawing of T_A and e is the link node of T_A. Hence, e is either at the bottom of Γ_A (from Property 2), or is at the top-left corner of Γ_A, and no other edge or node of T_A is placed on, or crosses the horizontal and vertical channels occupied by it (from Properties 1 and 3). Hence, in the case $A < 1$, drawing edge (e, a) will not create any edge-crossings, and Γ will also be a feasible drawing of T. In the case $A \geq 1$ also, drawing edge (e, a) will not create any edge-crossings because after rotating Γ_A by 90° and flipping it vertically, e will either be at the right boundary of Γ_A (see Property 2), or at the top-left corner of Γ_β and no other edge or node of T_A will be placed on, or cross the horizontal channel occupied by it (see Properties 1 and 3). Thus, for the case $A \geq 1$ also, Γ will also be a feasible drawing of T.

- *Subcase (b):* See Figure 3(b). Because $\Gamma'_1, \ldots, \Gamma'_i, \Gamma_B$ are feasible drawings of $T'_1, \ldots, T'_i, T_B$ respectively, it is straightforward to see that Γ is also a feasible drawing of T for both the cases when $A < 1$ and $A \geq 1$.
- *Subcase (c):* See Figure 3(c). The proof is similar to the one for Subcase (a).
- *Subcase (d):* See Figure 3(d). Γ_A is a feasible drawing of T_A, e is the link node of T_A, and $e \neq r$. Hence, from Property 2, e is located at the bottom of Γ_A. Rotating Γ_A by $90°$ and flipping it vertically will move e to the right boundary of Γ_A. Moving e to the right until it is either to the right of, or aligned with the right boundary of Γ_B will not cause any edge-crossings because of Property 2. It can be easily seen that in both the cases, $A < 1$ and $A \geq 1$, drawing edge (e, u^*) does not create any edge-crossings, and Γ is a feasible drawing of T.
- *Subcase (e):* See Figure 3(e). Γ_A is a feasible drawing of T_A, e is the link node of T_A, and $e = r$. Hence, from Properties 1 and 3, e is at the top-left corner of Γ_A, and no other edge or node of T_A is placed on, or crosses the horizontal and vertical channels occupied by it. Hence, in both the cases, $A < 1$ and $A \geq 1$, drawing edge (e, u^*) will not create any edge-crossings, and Γ is a feasible drawing of T.
- *Subcase (f):* See Figure 3(f). It is straightforward to see that Γ is a feasible drawing of T for both the cases when $A < 1$ and $A \geq 1$.
- *Subcase (g):* See Figure 3(g). Γ_C is a feasible drawing of T_C, u^* is the link node of T_C, and u^* is also the root of T_C. Hence, from Properties 1 and 3, u^* is at the top-left corner of Γ_C, and no other edge or node of T_C is placed on, or crosses the horizontal and vertical channels occupied by it. Flipping Γ_C vertically will move u^* to the bottom-left corner of Γ_C and no other edge or node of T_C will be on or crosses the vertical channel occupied by it. Hence, drawing edge (r, u^*) will not create any edge-crossings, and Γ will be a feasible drawing of T.

Using a similar reasoning, we can show that in Case 2 also, Γ is a feasible drawing of T. □

Lemma 2 (Time) *Given an n-node degree-d tree T, where $d = O(n^\delta)$ is a positive integer and $0 \leq \delta < 1$, with a link node u^*, Algorithm* DrawTree *will construct a drawing Γ of T in $O(n \log n)$ time.*

Proof: From Theorem 1, each partial tree into which Algorithm *DrawTree* would split T will have at most $n/2$ nodes in it. Hence, it follows that the depth of the recursion for Algorithm *DrawTree* is $O(\log n)$. At the first recursive level, the algorithm will split T into partial trees, assign aspect ratios to the partial trees and compose the drawings of the partial trees to construct a drawing of T. At the next recursive level, it will split all of these partial trees into smaller partial trees, assign aspect ratios to these smaller partial trees, and compose the drawings of these smaller partial trees to construct the drawings of all the partial trees. This process will continue until the bottommost recursive level is reached. At each recursive level, the algorithm takes $O(m)$ time to split a tree with m nodes into partial trees, assign aspect ratios to the partial trees, and compose the drawings of partial trees to construct a drawing of the tree. At each recursive level, the total number of nodes in all the trees that the algorithm considers for drawing is at most n. Hence, at each recursive level, the algorithm totally spends $O(n)$ time. Hence, the running time of the algorithm is $O(n) \cdot O(\log n) = O(n \log n)$.

□

In Lemma 4 given below, we prove that the algorithm will draw a degree-d tree, where $d = O(n^\delta)$ is a positive integer and $0 \leq \delta < 1$, in $O(n)$ area.

Lemma 3 *Let R be a rectangle with area D and aspect ratio A. Let W and H be the width and height, respectively, of R. Then, $W = \sqrt{AD}$ and $H = \sqrt{D/A}$.*

Proof: By the definition of aspect ratio, $A = W/H$. $D = WH = W(W/A) = W^2/A$. Hence, $W = \sqrt{AD}$. $H = W/A = \sqrt{AD}/A = \sqrt{D/A}$. □

Lemma 4 (Area) *Let T be an n-node degree-d tree, where $d = O(n^\delta)$ is a positive integer and $0 \leq \delta < 1$, with a link node u^*. Let ϵ and A be two numbers such that $\delta/(1-\delta) < \epsilon < 1$, and A is in the range*

$[n^{-\epsilon}, n^{\epsilon}]$. *Given T, ϵ, and A as input, Algorithm* DrawTree *will construct a drawing Γ of T that can fit inside a rectangle R with $O(n)$ area and aspect ratio A.*

Proof: Let $D(n)$ be the area of R. We will prove, using induction over n, that $D(n) = O(n)$. More specifically, we will prove that $D(n) \leq c_1 n - c_2 n^{\beta}$ for all $n \geq n_0$, where n_0, c_1, c_2, β are some positive constants and $\beta < 1$.

We now give the proof for the case when $A \geq 1$ (the proof for the case $A < 1$ is symmetrical). Algorithm *DrawTree* will split T into at most $2d-1$ partial trees. Let T_k be a non-empty partial tree of T, where T_k is one of $T_A, T_C, T_\beta, T'_1, \ldots, T'_i$, $0 \leq i \leq d-3$, $T_1, \ldots, T_j$, $0 \leq j \leq d-1$, in Case 1, and is one of $T_A, T_C, T_1, \ldots, T_j$, $0 \leq j \leq d-2$, in Case 2. Let n_k be the number of nodes in T_k, and let $x_k = n_k/n$. Let $P_k = c_1 n - c_2 n^{\beta}/x_k^{1-\beta}$. From Theorem 1, it follows that $n_k \leq n/2$, and hence, $x_k \leq 1/2$. Hence, $P_k \leq c_1 n - c_2 n^{\beta}/(1/2)^{1-\beta} = c_1 n - c_2 n^{\beta} 2^{1-\beta}$. Let $P' = c_1 n - c_2 n^{\beta} 2^{1-\beta}$. Thus, $P_k \leq P'$.

From the inductive hypothesis, Algorithm *DrawTree* will construct a drawing Γ_k of T_k that can fit inside a rectangle R_k with aspect ratio A_k and area $D(n_k)$, where A_k is as defined in Section 5.2, and $D(n_k) \leq c_1 n_k - c_2 n_k^{\beta}$. Since $x_k = n_k/n$, $D(n_k) \leq c_1 n_k - c_2 n_k^{\beta} = c_1 x_k n - c_2 (x_k n)^{\beta} = x_k (c_1 n - c_2 n^{\beta}/x_k^{1-\beta}) = x_k P_k \leq x_k P'$.

Let W_k and H_k be the width and height, respectively, of R_k. We now compute the values of W_k and H_k in terms of A, P', x_k, n, and ϵ. We have two cases:

- *T_k is a small partial tree of T:* Then, $n_k < (n/A)^{1/(1+\epsilon)}$, and also, as explained in Section 5.2, $A_k = 1/n_k^{\epsilon}$. From Lemma 3, $W_k = \sqrt{A_k D(n_k)} \leq \sqrt{(1/n_k^{\epsilon})(x_k P')} = \sqrt{(1/n_k^{\epsilon})(n_k/n)P'} = \sqrt{n_k^{1-\epsilon} P'/n}$. Since $n_k < (n/A)^{1/(1+\epsilon)}$, $W_k < \sqrt{(n/A)^{(1-\epsilon)/(1+\epsilon)} P'/n} = \sqrt{(1/A^{(1-\epsilon)/(1+\epsilon)}) P'/n^{2\epsilon/(1+\epsilon)}} \leq \sqrt{P'/n^{2\epsilon/(1+\epsilon)}}$ since $A \geq 1$.
  From Lemma 3, $H_k = \sqrt{D(n_k)/A_k} \leq \sqrt{x_k P'/(1/n_k^{\epsilon})} = \sqrt{(n_k/n) P' n_k^{\epsilon}} = \sqrt{n_k^{1+\epsilon} P'/n}$. Since $n_k < (n/A)^{1/(1+\epsilon)}$, $H_k < \sqrt{(n/A)^{(1+\epsilon)/(1+\epsilon)} P'/n} = \sqrt{(n/A) P'/n} = \sqrt{P'/A}$.
- *T_k is a <u>large</u> partial tree of T:* Then, as explained in Section 5.2, $A_k = x_k A$. From Lemma 3, $W_k = \sqrt{A_k D(n_k)} \leq \sqrt{x_k A x_k P'} = x_k \sqrt{AP'}$.
  From Lemma 3, $H_k = \sqrt{D(n_k)/A_k} \leq \sqrt{x_k P'/(x_k A)} = \sqrt{P'/A}$.

In Step *Compose Drawings*, we use at most two additional horizontal channels and at most one additional vertical channels while combining the drawings of the partial trees to construct a drawing Γ of T. Hence, Γ can fit inside a rectangle R' with width W' and height H', respectively, where,

$$H' \leq \max_{T_k \text{ is a partial tree of } T} \{H_k\} + 2 \leq \sqrt{P'/A} + 2,$$

and

$$\begin{aligned} W' &\leq \sum_{T_k \text{ is a large partial tree}} W_k + \sum_{T_k \text{ is a small partial tree}} W_k + 1 \\ &\leq \sum_{T_k \text{ is a large partial tree}} x_k \sqrt{AP'} + \sum_{T_k \text{ is a small partial tree}} \sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + 1 \\ &\leq \sqrt{AP'} + (2d-1)\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + 1 \end{aligned}$$

(because $\sum_{T_k \text{ is a large partial tree}} x_k \leq 1$, and T is split into at most $2d-1$ partial trees)

R' does not have aspect ratio equal to A, but it is contained within a rectangle R with aspect ratio A, area $D(n)$, width W, and height H, where

$$W = \sqrt{AP'} + (2d-1)\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + 1 + 2A,$$

and

$$H = \sqrt{P'/A} + 2 + ((2d-1)/A)\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + 1/A$$

Hence, $D(n) = WH = (\sqrt{AP'} + (2d-1)\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + 1 + 2A)(\sqrt{P'/A} + 2 + ((2d-1)/A)\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + 1/A) = P' + 2(2d-1)P'/\sqrt{An^{2\epsilon/(1+\epsilon)}} + 4\sqrt{AP'} + (2d-1)^2 P'/(An^{2\epsilon/(1+\epsilon)}) + 4(2d-1)\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + 4A +$

$4 + 1/A + 2\sqrt{P'/A} + 2(2d-1)\sqrt{P'/n^{2\epsilon/(1+\epsilon)}}/A$.

Since, $1 \le A \le n^\epsilon$, we have that

$$\begin{aligned} D(n) &\le P' + c_3 dP'/\sqrt{n^{2\epsilon/(1+\epsilon)}} + c_4\sqrt{n^\epsilon P'} + c_5 d^2 P'/n^{2\epsilon/(1+\epsilon)} + c_6 P'/n^{2\epsilon/(1+\epsilon)} \\ &\quad + c_7 d\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + c_8 n^\epsilon + c_9 + c_{10}\sqrt{P'} \end{aligned}$$

where $c_3, c_4, \ldots, c_{10}$ are some constants.

Since $P' < c_1 n$,

$$\begin{aligned} D(n) &< P' + c_3 dc_1 n/\sqrt{n^{2\epsilon/(1+\epsilon)}} + c_4\sqrt{n^\epsilon c_1 n} + c_5 d^2 c_1 n/n^{2\epsilon/(1+\epsilon)} + c_6 c_1 n/n^{2\epsilon/(1+\epsilon)} \\ &\quad + c_7 d\sqrt{c_1 n/n^{2\epsilon/(1+\epsilon)}} + c_8 n^\epsilon + c_9 + c_{10}\sqrt{c_1} n^{1/2} \\ &= P' + c_3 dc_1 n^{1/(1+\epsilon)} + c_4\sqrt{c_1} n^{(1+\epsilon)/2} + c_5 d^2 c_1 n^{(1-\epsilon)/(1+\epsilon)} + c_6 c_1 n^{(1-\epsilon)/(1+\epsilon)} \\ &\quad + c_7 d\sqrt{c_1} n^{(1-\epsilon)/(2(1+\epsilon))} + c_8 n^\epsilon + c_9 + c_{10}\sqrt{c_1} n^{1/2} \\ &\le P' + c_{11} n^{(1+\epsilon)/2} + c_{12} dn^{1/(1+\epsilon)} + c_{13} d^2 n^{(1-\epsilon)/(1+\epsilon)} \end{aligned}$$

where c_{11}, c_{12}, and c_{13} are large enough constants (because, since $0 \le \delta/(1-\delta) < \epsilon < 1$, $(1-\epsilon)/(2(1+\epsilon)) < (1-\epsilon)/(1+\epsilon) < 1/(1+\epsilon)$, $\epsilon < (1+\epsilon)/2$, and $1/2 < (1+\epsilon)/2$).

Because $d = O(n^\delta)$, for a large enough constant n_0, there exist constants c_{14} and c_{15} such that for all $n \ge n_0$, $D(n) \le P' + c_{11} n^{(1+\epsilon)/2} + c_{14} n^{\delta + 1/(1+\epsilon)} + c_{15} n^{2\delta + (1-\epsilon)/(1+\epsilon)}$.

$P' = c_1 n - c_2 n^\beta 2^{1-\beta} = c_1 n - c_2 n^\beta (1 + c_{16})$, where c_{16} is a constant such that $1 + c_{16} = 2^{1-\beta}$.

Hence, $D(n) \le c_1 n - c_2 n^\beta (1 + c_{16}) + c_{11} n^{(1+\epsilon)/2} + c_{14} n^{\delta+1/(1+\epsilon)} + c_{15} n^{2\delta+(1-\epsilon)/(1+\epsilon)} = c_1 n - c_2 n^\beta - (c_{16} n^\beta - c_{11} n^{(1+\epsilon)/2} - c_{14} n^{\delta+1/(1+\epsilon)} - c_{15} n^{2\delta+(1-\epsilon)/(1+\epsilon)})$. Thus, for a large enough constant n_0, and large enough β, where $1 > \beta > \max\{(1+\epsilon)/2, \delta + 1/(1+\epsilon), 2\delta + (1-\epsilon)/(1+\epsilon)\}$, for all $n \ge n_0$, $c_{16} n^\beta - c_{11} n^{(1+\epsilon)/2} - c_{14} n^{\delta+1/(1+\epsilon)} - c_{15} n^{2\delta+(1-\epsilon)/(1+\epsilon)} \ge 0$, and hence $D(n) \le c_1 n - c_2 n^\beta$. Note that because $\epsilon > \delta/(1-\delta)$, $\delta + 1/(1+\epsilon) < 1$ and $2\delta + (1-\epsilon)/(1+\epsilon) < 1$, and because $\epsilon < 1$, $(1+\epsilon)/2 < 1$.

The proof for the case $A < 1$ uses the same reasoning as for the case $A \ge 1$. With T_k, R_k, W_k, H_k, R', W', H', R, W, and H defined as above, and A_k as defined in Section 5.2, we get the following values for W_k, H_k, W', H', W, H, and $D(n)$:

$$\begin{aligned} W_k &\le \sqrt{AP'} \\ H_k &\le \sqrt{P'/n^{2\epsilon/(1+\epsilon)}} \text{ if } T_k \text{ is a small partial tree} \\ &\le x_k\sqrt{P'/A} \text{ if } T_k \text{ is a large partial tree} \\ W' &\le \sqrt{AP'} + 2 \\ H' &\le \sqrt{P'/A} + (2d-1)\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + 1 \\ W &\le \sqrt{AP'} + 2 + (2d-1)A\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + A \\ H &\le \sqrt{P'/A} + (2d-1)\sqrt{P'/n^{2\epsilon/(1+\epsilon)}} + 1 + 2/A \\ D(n) &\le P' + c_{11} n^{(1+\epsilon)/2} + c_{14} n^{\delta+1/(1+\epsilon)} + c_{15} n^{2\delta+(1-\epsilon)/(1+\epsilon)} \end{aligned}$$

where c_{11}, c_{14}, and c_{15} are the same constants as in the case $A \ge 1$. Therefore, $D(n) \le c_1 n - c_2 n^\beta$ for $A < 1$ too. (Notice that in the values that we get above for W_k, H_k, W', H', W, and H, if we replace A by $1/A$, exchange W_k with H_k, exchange W' with H', and exchange W with H, we will get the same values for W_k, H_k, W', H', W, and H as in the case $A \ge 1$. This basically reflects the fact that the cases $A \ge 1$ and $A < 1$ are symmetrical to each other.) □

Theorem 2 (Main Theorem) *Let T be an n-node degree-d tree, where $d = O(n^\delta)$ is a positive integer and $0 \le \delta < 1/2$ is a constant. Given any number A, where $n^{-\alpha} \le A \le n^\alpha$, for some constant α, where $0 \le \alpha < 1$, we can construct in $O(n \log n)$ time, a planar straight-line grid drawing of T with $O(n)$ area, and aspect ratio A.*

Proof: Let ϵ be a constant such that $n^{-\epsilon} \leq A \leq n^{\epsilon}$ and $\delta/(1-\delta) < \epsilon < 1$. Designate any leaf of T as its link node. Construct a drawing Γ of T in R by calling Algorithm *DrawTree* with T, A and ϵ as input. From Lemmas 1, 2, and 4, Γ will be a planar straight-line grid drawing of T contained entirely within a rectangle with $O(n)$ area, and aspect ratio A. □

Corollary 1 *Let T be an n-node degree-d tree, where $d = O(n^{\delta})$ is a positive integer and $0 \leq \delta < 1/2$ is a constant.. We can construct in $O(n \log n)$ time, a planar straight-line grid drawing of T with optimal (equal to $O(n)$) area, and optimal aspect ratio (equal to 1).*

Proof: Immediate from Theorem 2, with $A = 1$. □

References

[1] T. Chan, M. Goodrich, S. Rao Kosaraju, and R. Tamassia. Optimizing area and aspect ratio in straight-line orthogonal tree drawings. *Comput. Geom. Theory Appl.*, 23:153–162, 2002.

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

[3] P. Crescenzi, P. Penna, and A. Piperno. Linear-area upward drawings of AVL trees. *Comput. Geom. Theory Appl.*, 9:25–42, 1998. (special issue on Graph Drawing, edited by G. Di Battista and R. Tamassia).

[4] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing.* Prentice Hall, Upper Saddle River, NJ, 1999.

[5] A. Garg, M. T. Goodrich, and R. Tamassia. Planar upward tree drawings with optimal area. *Internat. J. Comput. Geom. Appl.*, 6:333–356, 1996.

[6] A. Garg and A. Rusu. Straight-line drawings of binary trees with linear area and arbitrary aspect ratio. In Michael T. Goodrich and Stephen G. Kobourov, editors, *Graph Drawing (Proceedings of 10^{th} International Sympsium on Graph Drawing, 2002)*, volume 2528 of *Lecture Notes in Computer Science*, pages 320–331. Springer-Verlag, 2002.

[7] C. E. Leiserson. Area-efficient graph layouts (for VLSI). In *Proc. 21st Annu. IEEE Sympos. Found. Comput. Sci.*, pages 270–281, 1980.

[8] M. Sarkar and M. H. Brown. Graphical fisheye views. *Commun. ACM*, 37(12):73–84, 1994.

[9] C.-S. Shin, S.K. Kim, S.-H. Kim, and K.-Y. Chwa. Area-efficient algorithms for straight-line tree drawings. *Comput. Geom. Theory Appl.*, 15:175–200, 2000.

[10] L. Trevisan. A note on minimum-area upward drawing of complete and Fibonacci trees. *Inform. Process. Lett.*, 57(5):231–236, 1996.

[11] L. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Comput.*, C-30(2):135–140, 1981.