A Near-Linear Area Bound for Drawing Binary Trees*

Timothy M. Chan†

August 28, 2001

Abstract

We present several simple methods to construct planar, strictly upward, strongly order-preserving, straight-line drawings of any n-node binary tree. In particular, it is shown that $O(n^{1+\varepsilon})$ area is always sufficient for an arbitrary constant $\varepsilon > 0$.

Key Words. Graph drawing, Trees.

1 Introduction

What is a good way to draw a given binary tree? Several natural criteria come to mind. As one can observe from usual pictures in computer science textbooks, drawings should be:

1. *Planar.* Nodes are drawn as distinct points in the plane, and if node u is a child of node v, a polygonal curve ("polyline") should be drawn connecting u and v. We want to ensure that no two curves cross.

2. *Strictly upward.* To see which is the parent/child in a curve, we may require that the curve from the parent to the child is strictly decreasing in the y-direction.

3. *Strongly order-preserving.* To tell which is the left/right child of a given node, we may require that the curve from the parent to the left child is monotone decreasing in the x-direction, and the curve from the parent to the right child is monotone increasing in the x-direction.

4. *Straight-line.* To make visualization easier, we would like each curve to be a single line segment.

For lack of a better adjective, we call drawings that satisfy all of the above *ideal* drawings of binary trees in this paper. See Figure 1 for examples.

Finally, we want our drawing to take up as little space as possible. To measure space, we require the vertices of each line segment to have integer coordinates and define the *height*, *width*, and *area* of the drawing to be the height, width, and area of its bounding box (i.e., smallest axes-parallel rectangle). Our objective is to investigate worst-case asymptotic upper bounds on the area as a

*A preliminary version of this paper appeared in *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 161–168, 1999. This work was done while the author was at the Department of Mathematics and Computer Science, University of Miami.

†Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, **tmchan@math.uwaterloo.ca**

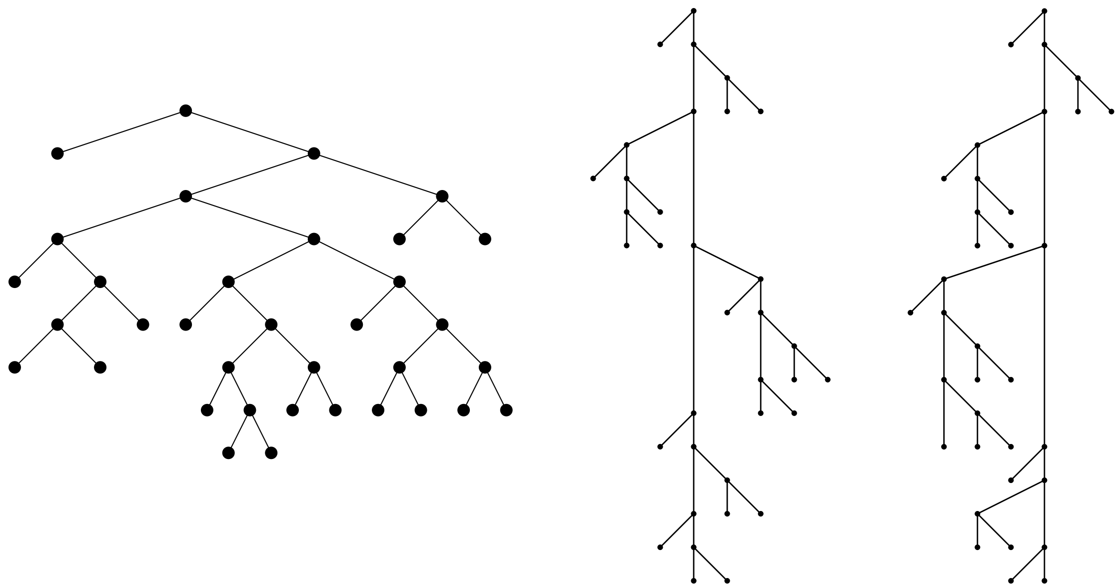

Figure 1: A binary tree and two ideal drawings (produced by the first and second method of this paper). The first has area 7×17, the second 6×17.

function of the size of the tree (i.e., the number of nodes). Specifically, we prove that any binary tree of size n has an ideal drawing of $O(n^{1+\varepsilon})$ area for any fixed constant $\varepsilon > 0$.

Although the literature in graph drawing is vast [5, 12], surprisingly no nontrivial upper bounds on the area were previously reported for ideal drawings of arbitrary binary trees, even if criteria 2 and 3 are weakened to:

2′. *Upward.* The curve from the parent to the child is monotone decreasing in the y-direction.

3′. *Order-preserving.* The curve from the parent to the left child is to the left of the curve from the parent to the right child.

Table 1 summarizes the known results for various combinations of the four criteria. The two $O(n \log n)$ area bounds in the table, due to Crescenzi, Di Battista, and Piperno [2] and Garg, Goodrich, and Tamassia [6], were obtained by simple recursive algorithms and proved to be tight in the worst case for their corresponding types of drawings. The type of drawings considered by Crescenzi *et al.* is not order-preserving, and thus, one cannot reconstruct the binary tree uniquely from the drawing (without the help of labels) if the order of the children is important. On the other hand, the type of drawings considered by Garg *et al.* is not straight-line, hence, may not be as aesthetically pleasing. All known drawing methods that simultaneously fulfill the four criteria (such as the early work by Reingold and Tilford [9]) require $O(n^2)$ area in the worst case [5].

Not in the table are results dealing with special types of trees. For most balanced trees (including the complete binary tree, the Fibonacci tree, AVL trees, and red-black trees), ideal drawings satisfying all criteria 1–4 can be constructed using only $O(n)$ area; see the references [3, 4, 7, 11, 13]. Also not in the table are results regarding *orthogonal* drawings, i.e., drawings in which all line segments are either horizontal or vertical; see the references [1, 2, 6, 8, 11, 14].

planar?	*upward?*	*order-preserv.?*	*straight-line?*	area	references
yes	yes	no	no	$O(n)$	[6]
yes	yes	no	yes	$O(n \log\log n)$	[11]
yes	yes (strictly)	no	yes	$O(n \log n)$	[2, 10]
yes	yes (strictly)	yes	no	$O(n \log n)$	[6]
yes	yes (strictly)	yes (strongly)	yes	$O(n^{1+\varepsilon})$	this paper

Table 1: Area bounds for drawing arbitrary binary trees.

Note that despite its naturalness, our strong definition of order-preserving drawings (criterion 3) is not as well studied. One may insist on an even stronger condition where the curves are not only monotone increasing/decreasing in the x-direction, but strictly increasing/decreasing. But with this restriction, $\Omega(n^2)$ area might be necessary while maintaining the upward property (for example, consider a tree that is a path with only right children).

The result of the present paper is derived here incrementally through a series of improvements. It is easy to construct ideal drawings of $O(n^2)$ area. As a first step, we examine a simple recursive algorithm (Section 2) and show remarkably that it always produces ideal drawings of subquadratic area, specifically, $O(n^{1.695})$ area (Section 3). This algorithm is based on the applications of two natural rules. We then refine the algorithm using a more judicious choice of these two rules to obtain an area bound of $O(n^{1.5})$ (Section 4) or even $O(n^{1.48})$ (Appendix A). Next, we extend the two rules slightly and show how to achieve $O(n^{1+\varepsilon})$ area for an arbitrarily small constant $\varepsilon > 0$; more precisely, the area is $O(n2^{\sqrt{2\log n}}\sqrt{\log n})$ (Section 5). An $O(n4^{2\sqrt{\log n}})$ bound can also be obtained by another simple method that can be generalized to handle ordered trees of any degree (Section 6). In this paper, all logarithms are in base 2.

2 First Method

It is not possible to produce strictly upward drawings with $o(n)$ height in general (for exmaple, consider a tree that is a path), so we focus our effort in minimizing the width while maintaining $O(n)$ height. One way to get such "narrow" drawings is as follows.

Let T be the given binary tree with root v. Suppose we have already constructed ideal drawings of the left subtree L and the right subtree R. We can combine the two drawings into an ideal drawing of T by applying one of the two following rules at the root v. In the *left rule*, we vertically align v with the root of R, place the bounding box of L one unit below and one unit to the left of v, and place the bounding box of R immediately below that of L. The *right rule* is defined symmetrically. See Figure 2. The special case where one of the subtrees is empty is left to the reader.

(To readers familiar with the literature, we remark that these two rules are quite different from the two rules of "h-v drawings" [5], for example, in that the root does not have to placed at the upper-left corner of the bounding box.)

Our first two drawing methods are obtained by recursive applications of the left and right rules. The only remaining question is which rule to choose at a given node. Our initial response is based on comparisons of subtree sizes and leads to this simple algorithm to draw T:

If $|T| \le 1$, return the trivial drawing. Otherwise, draw L and R recursively. Combine the

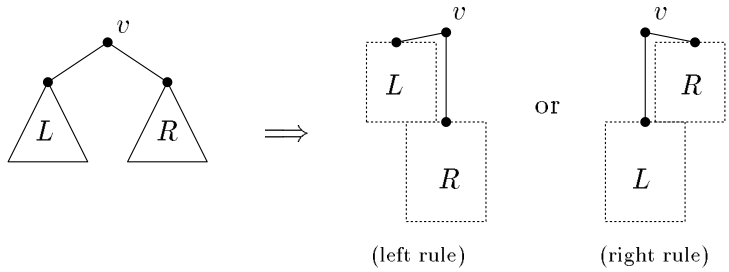

Figure 2: Two rules to draw a binary tree.

drawings using the left rule if $|L| < |R|$, and the right rule otherwise.

Here and throughout the paper, $|T|$ denotes the the number of nodes in T.

3 Analysis

Let $n = |T|$. It is straightforward to see that the preceding algorithm returns an ideal drawing of height at most $n - 1$. Despite the simplicity of the algorithm, the analysis of the width though is not obvious and makes for an interesting exercise. The following alternative view of the algorithm turns out to be helpful, not only to this analysis but also to the development of the subsequent improved methods.

Let $\pi = \langle v_0, v_1, \ldots, v_k \rangle$ be any downward path in the tree T. A *subtree of the path* π refers to the subtree rooted at the sibling of a node from $\{v_1, \ldots, v_k\}$. It is called a *left* or *right subtree* depending on whether its root is a left or right child. (By default, the empty tree is both a left and a right subtree.) Consider the following generic algorithm to draw T:

> Pick a path π from the root down to some leaf. Draw the subtrees of π recursively. Combine the drawings by applying the left and right rules at nodes on the path π in such a way that all nodes on π are vertically aligned.

See Figure 3 for an example. If we denote the width of the drawing of T by $W(T)$, then

$$W(T) \;=\; W(\alpha) + W(\beta) + 2 \tag{1}$$

for some left subtree α and some right subtree β of the path π.

The algorithm in Section 2 is identical to an instance of the generic algorithm where the root-to-leaf path $\pi = \langle v_0, v_1, \ldots \rangle$ is chosen in the following "greedy" fashion, with $T_0 = T$:

> Let v_i, L_i and R_i be the root, left subtree and right subtree of T_i respectively. If $|R_i| \leq |L_i|$, then set $T_{i+1} = L_i$, else set $T_{i+1} = R_i$.

We call this particular path the *greedy path*.

Lemma 3.1 *For any two different subtrees α and β of the greedy path, one of the following is true:* (i) $|\alpha| \leq n/2$ *and* $|\beta| \leq (n - |\alpha|)/2$, *or* (ii) $|\beta| \leq n/2$ *and* $|\alpha| \leq (n - |\beta|)/2$.

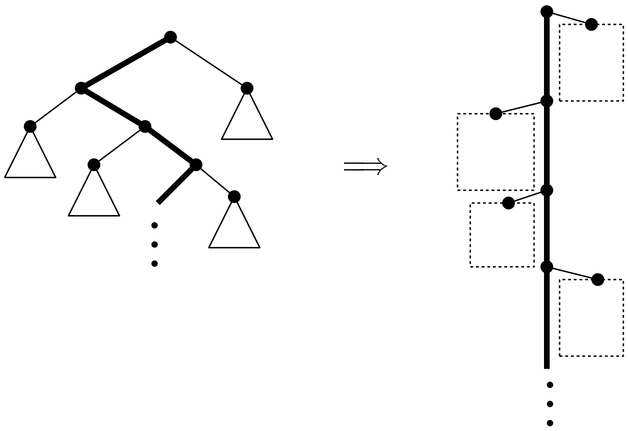

Figure 3: The generic algorithm. Bold line segments indicate the chosen path π. Subtrees of the path are drawn recursively.

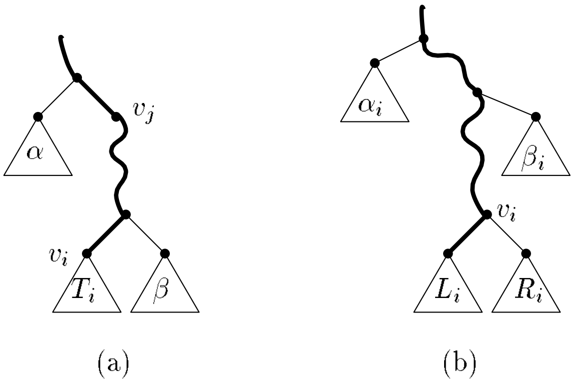

Figure 4: (a) Proof of Lemma 3.1. (b) Proof of Lemma 4.1.

Proof: Say the siblings of α's root and β's root are v_j and v_i respectively. Without loss of generality, suppose that $j < i$. By definition, $|\alpha| \le |T_j|$ and $|\beta| \le |T_i|$. In addition, $|\alpha| + |T_j| \le n$ and $|\alpha| + |T_i| + |\beta| \le n$. See Figure 4(a). Combining these inequalities, we get (i). $\square$

With equation (1), the above lemma immediately implies the following recurrence on the maximum width $W(n)$ for trees of size n:

$$W(n) \;\le\; \max_{n_1 \le n/2,\; n_2 \le (n-n_1)/2} (W(n_1) + W(n_2) + 2).$$

By considering the "critical" case $n_1 = n/2$ and $n_2 = n/4$ and solving the familiar recurrence

$$W(n) \;\le\; W(n/2) + W(n/4) + O(1), \tag{2}$$

one would suspect that $W(n) = O(n^{\log\phi})$ with the golden ratio $\phi = (1+\sqrt{5})/2$. This intuition can be confirmed formally by observing that $n_1 \le n/2$ and $n_2 \le (n - n_1)/2$ imply the following, by Hölder's inequality:

$$n_1^{\log\phi} + n_2^{\log\phi} \;\;=\;\; (1 - 1/2^{\log\phi})n_1^{\log\phi} \;+\; (n_1/2)^{\log\phi} \;+\; n_2^{\log\phi}$$

$$\begin{aligned} &\le (1 - 1/2^{\log\phi}) n_1^{\log\phi} \;+\; 2^{1-\log\phi}(n_1/2 + n_2)^{\log\phi} \\ &\le (1 - 1/\phi)(n/2)^{\log\phi} \;+\; (2/\phi)(n/2)^{\log\phi} \\ &\le (1 + 1/\phi)(1/\phi) n^{\log\phi} \;=\; n^{\log\phi}. \end{aligned}$$

By induction, we can then prove that $W(n) \le cn^{\log\phi} - 2$ for all $n \ge 1$ for a suitable constant c, and therefore conclude that the algorithm in Section 2 produces ideal drawings of $O(n)$ height and $O(n^{0.695})$ width.

One can devise an example where this particular algorithm requires exactly $\Theta(n^{\log\phi})$ width, by a recursive construction to match a recurrence like (2).

4 Second Method

We now point out that a different choice of the path π in the generic algorithm of Section 3 leads to a better recurrence for the width. Although this improvement will be overshadowed by our third method, it is nevertheless interesting to see how much we can gain by utilizing only the left and right rules.

The new root-to-leaf path $\pi = \langle v_0, v_1, \ldots\rangle$ is defined as follows, with $T_0 = T$:

> Let v_i, L_i and R_i be the root, left subtree, and right subtree of T_i respectively. Furthermore, let α_i be the left subtree of the subpath $\langle v_0, \ldots, v_i\rangle$ of the largest size, and let β_i be the right subtree of $\langle v_0, \ldots, v_i\rangle$ of the largest size. If $|\alpha_i| + |R_i| \le |L_i| + |\beta_i|$, then set $T_{i+1} = L_i$, else set $T_{i+1} = R_i$.

We call this particular path the *halving path*, as explained by the lemma below:

Lemma 4.1 *For any left subtree α and right subtree β of the halving path, $|\alpha| + |\beta| \le n/2$.*

Proof: Say the parents of α's root and β's root are v_j and v_i respectively. By symmetry, we may assume that $j < i$. By construction, $|\alpha_i| + |R_i| \le |L_i| + |\beta_i|$. In addition, $|\alpha_i| + |L_i| + |R_i| + |\beta_i| \le n$. See Figure 4(b). Therefore, $|\alpha_i| + |R_i| \le n/2$, implying the lemma as $|\alpha| \le |\alpha_i|$ and $\beta = R_i$. □

Using the halving path π in the generic algorithm, we get a new recurrence from equation (1):

$$W(n) \;\le\; \max_{n_1 + n_2 \le n/2} (W(n_1) + W(n_2) + 2).$$

By Hölder's inequality, $n_1 + n_2 \le n/2$ implies $\sqrt{n_1} + \sqrt{n_2} \le \sqrt{n}$. By induction, $W(n) \le c\sqrt{n} - 2$ for all $n \ge 1$ for some constant c. We have therefore demonstrated the existence of an ideal drawing of $O(n)$ height and $O(\sqrt{n})$ width. Still, slight improvements on the width bound are possible by a similar strategy with a more complicated analysis; see Appendix A.

5 Third Method

To obtain a near-linear area bound, we find it necessary to extend the left and right rules of Section 2, as illustrated in Figure 5. In applying the *extended left rule* at the root v of T, we are allowed to translate the bounding box of the right subtree R horizontally by an arbitrary amount, as long as

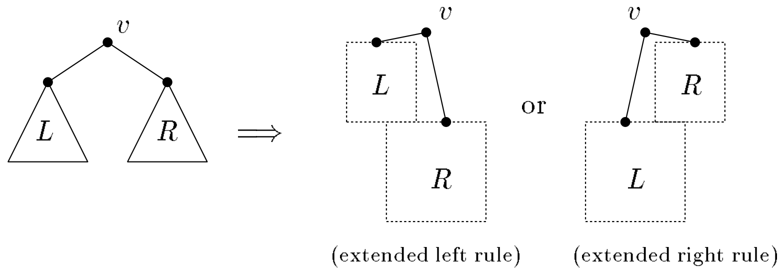

Figure 5: Two extended rules to draw a binary tree T.

the x-coordinate of the root of R is greater than or equal to the x-coordinate of v. The *extended right rule* is defined in a symmetric manner. It is easy to see that all criteria 1–4 are preserved under these two rules.

Choose a parameter A. To obtain our third drawing algorithm, we return to the greedy path $\pi = \langle v_0, v_1, \ldots \rangle$ as defined in Section 3. As before, the subtree rooted at v_i is denoted by T_i. Let k be the largest index such that $|T_k| \geq n - A$.

(For $A \approx 2n/3$, the vertex v_k thus defined is related to "centroids" or "separators" of trees, which have previously found several applications in tree drawings; for example, see the references [1, 6, 8, 11, 14].)

Our first case is when v_k is a left child. Let π' be the subpath $\langle v_0, \ldots, v_k \rangle$ and let π'' be the leftmost path from v_k down to a leaf. Consider applying the generic algorithm of Section 3 on the concatenation of these two paths π' and π'', but with a "bend" (literally):

> Draw the subtrees of π' and π'' recursively. To combine the drawings, apply the left and right rules at $v_0, \ldots, v_{k-2}$ so that the $v_0, \ldots, v_{k-1}$ are vertically aligned. Apply the left rule at nodes on π'' so that these nodes are vertically aligned as well. Finally, apply the extended left rule at v_{k-1} so that v_k is aligned with the left side of the bounding box of the entire drawing of T.

See Figure 6 for a better pictoral description. Notice that the width $W(T)$ of the drawing of T is given by

$$W(T) \;=\; \max\{\, W(\alpha) + W(\beta) + 2,\; W(\gamma) + 1 \,\} \tag{3}$$

for some left subtree α and some right subtree β of π', and some right subtree γ of π''.

The second case where v_k is a right child can be handled in a symmetric way (by choosing instead the rightmost path from v_k down to a leaf). We omit the easy treatment of the special case where v_k is the root of T.

Lemma 5.1 *Let the paths π' and π'' be defined as above.* (i) *For any subtree α of π', $|\alpha| \leq A$.* (ii) *For any subtree γ of π'', $|\gamma| \leq n - A$.*

Proof: (i) follows from the inequalities $|T_k| \geq n - A$ and $|\alpha| + |T_k| \leq n$, while (ii) follows from the fact that γ is contained in either the left or the right subtree of T_k, each of which has size at most $|T_{k+1}| < n - A$. □

Combining equation (3) with the above lemma, we get the following inequality on the maximum width $W(n)$ for trees of size n, given any choice of A:

$$W(n) \;\leq\; \max\{\, 2\,W(A) + 2,\; W(n-A) + 1 \,\}.$$

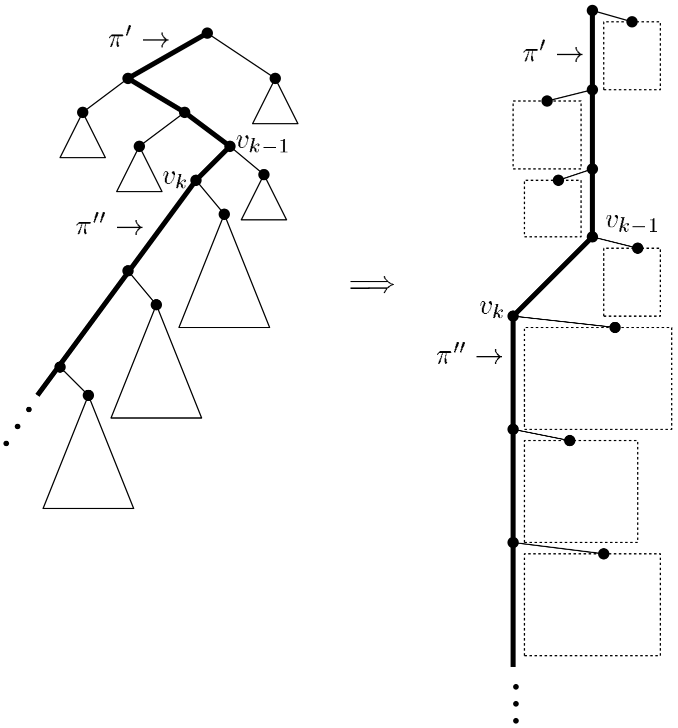

Figure 6: The modified generic algorithm on the concatenated path of π' and π''. Subtrees of the path are drawn recursively.

Iterating on the second term with a common value of A, we obtain

$$W(n) \;\leq\; 2\,W(A) + O(n/A).$$

If we set $A = n/2^{1/\varepsilon}$ for a fixed $\varepsilon > 0$, then the recurrence

$$W(n) \;\leq\; 2\,W(n/2^{1/\varepsilon}) + O(2^{1/\varepsilon})$$

solves to $W(n) = O(2^{1/\varepsilon} n^{\varepsilon})$, yielding the near-linear area result.

Actually, an even better effect is achieved by substituting $A = n/2^{\sqrt{2\log n}}$ instead. Then the recurrence

$$W(n) \;\leq\; 2\,W(n/2^{\sqrt{2\log n}}) + O(2^{\sqrt{2\log n}})$$

transforms to the following by a change of variables, $m = 2^{\sqrt{2\log n}}$, $n = 2^{(\log m)^2/2}$, and $f(m) := W(n)$:

$$\begin{aligned} f(m) \;&\leq\; 2\,W(2^{(\log m)^2/2 - \log m}) + O(m) \\ &\leq\; 2\,W(2^{(\log m - 1)^2/2}) + O(m) \;=\; 2f(m/2) + O(m). \end{aligned}$$

Thus, $f(m) = O(m \log m)$ and $W(n) = O(2^{\sqrt{2\log n}}\sqrt{\log n})$, giving our final bound for binary trees:

Theorem 5.2 *Any binary tree of size n admits an ideal drawing of height at most $n-1$ and width $O(2^{\sqrt{2\log n}}\sqrt{\log n})$.*

6 Fourth Method

The method from Section 5 does not extend easily to deal with ordered trees of degree $d > 2$. Nevertheless, we exhibit a fourth drawing method that yields a near-linear area bound for any d. Although some ideas are borrowed from the previous method, the new method actually has a simpler recursive structure. Unlike all our previous methods though, the height is not linear. Moreover, in the binary case ($d = 2$), the area bound is not as sharp as the one given by Theorem 5.2, and the drawing produced is also not as aesthetically nice (for reasons explicated in Section 7).

First, we need to define what we mean by an ideal drawing of an ordered tree. Criteria 1, 2, and 4 pose no problem, but criterion 3 does not generalize naturally. We take the following as the definition of the strongly order-preserving property: the line segment from a node to its leftmost/rightmost child is monotone decreasing/increasing in the x-direction; furthermore, the line segments from a node to all of its children are sorted by angle from left to right.

Given an ordered tree T of size n, we now describe a recursive algorithm to construct an ideal drawing of width $W(T)$ and height $H(T)$, where the root v is placed at the upper-left corner of the bounding box. The notation T^r stands for the tree obtained by reversing the order of the children at each node of T.

Let $T_1, \ldots, T_d$ be the subtrees of v from left to right. The rules we use to combine drawings of subtrees are more complicated geometrically. Let T_k be the largest subtree other than T_1. Assuming a parameter A has been chosen, we consider two cases:

Case 1: $|T_k| < n - A$. Then all subtrees except T_1 have size at most $n - A$. An ideal drawing of T can be obtained by vertically stacking the bounding boxes of the recursively computed drawings of the T_i's ($i = 1, \ldots, d$), as shown in Figure 7. The width and height of the resulting drawing is clearly bounded by:

$$\begin{aligned} W(T) &\le \max\{\max_{i \ne 1}(W(T_i) + 1),\ W(T_1)\} \le \max\{W(n - A) + 1,\ W(T_1)\}; \\ H(T) &\le \sum_i (H(T_i) + 1). \end{aligned}$$

Case 2: $|T_k| \ge n - A$. Then all subtrees other than T_k have size at most A. We modify the ideal drawing of T from *Case 1* in two ways. First, the drawing of T_k is replaced by the reflection of the drawing of T_k^r, so that its root v_k is placed at the upper-right corner of its bounding box. Second, we "pull down" the bounding box of T_k^r so that it lies underneath the bounding boxes of all the other subtrees. See Figure 7.

To ensure planarity, we must make v_k visible from v. One way to accomplish this is to divide the bounding box of $\overline{vv_k}$ into four equal-sized quadrants, place the drawings of $T_1, \ldots, T_{k-1}$ into the SW quadrant, and place the drawings of $T_{k+1}, \ldots, T_d$ into the NE quadrant. Such a drawing is feasible if the width and the height of each quadrant exceed $\max_{i \ne k}(W(T_i)+1)$ and $\sum_{i \ne k}(H(T_i)+1)$ respectively. Thus, the following bounds are achievable:

$$\begin{aligned} W(T) &\le \max\{2\max_{i \ne k}(W(T_i) + 1),\ W(T_k^r)\} \le \max\{2(W(A) + 1),\ W(T_k^r)\}; \\ H(T) &\le 2\sum_{i \ne k}(H(T_i) + 1) + H(T_k^r). \end{aligned}$$

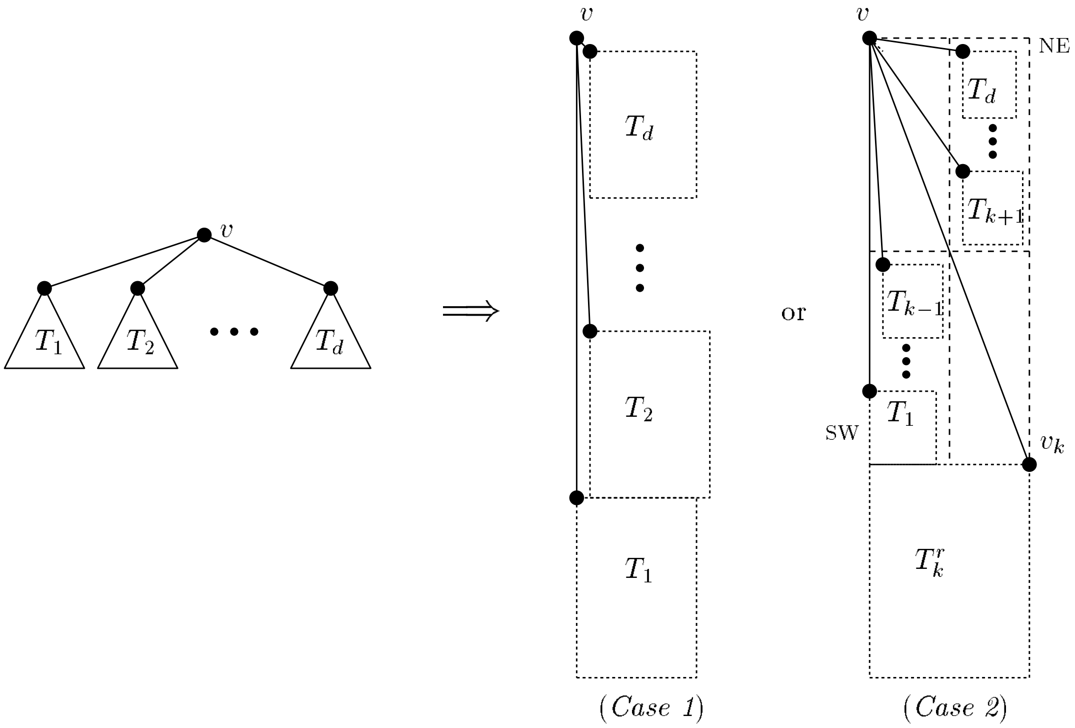

Figure 7: A recursive algorithm to draw an ordered tree.

We now analyze the worst-case width $W(n)$ and height $H(n)$ of the above algorithm. If we use the same value of A for the subtrees $T_1, \ldots, T_d$ in *Case 1*, and for the subtree T_k^r in *Case 2*, we can "unfold" the recurrences into:

$$\begin{aligned} W(n) &\leq 2\,W(A) + O(n/A); \\ H(n) &\leq 2\sum_i H(n_i) + O(n). \end{aligned}$$

Here, the n_i's form some sequence satisfying $n_i \leq A$ for each i and $\sum_i n_i \leq n$. Now, substitute $A = n/2^{1/\varepsilon}$ for a fixed $\varepsilon > 0$. As the depth of the recursion is near $\varepsilon \log n$, the recurrences solve to $W(n) = O(2^{1/\varepsilon} n^{\varepsilon})$ and $H(n) = O(n^{1+\varepsilon})$. The best result is obtained by setting $\varepsilon = 1/\sqrt{2\log n}$, yielding:

Theorem 6.1 *Any ordered tree of size n admits an ideal drawing of area $O(n4^{\sqrt{2\log n}})$.*

7 Remarks

The obvious open problem is either to improve the upper bounds of Theorems 5.2 and 6.1, or to derive an $\omega(n \log n)$ lower bound on the area of ideal drawings. We should remark that there is a simple recursive algorithm (related to our fourth method), as shown in Figure 8, that constructs ideal drawings of $O(\log n)$ width for any binary tree by using very small angles. Unfortunately, the height is superpolynomial ($\Omega(n^{\log\log n})$) in the worst case. Notice how easy it is to reduce the height of this drawing to linear if the straight-line condition is relaxed to allow for one bend per curve.

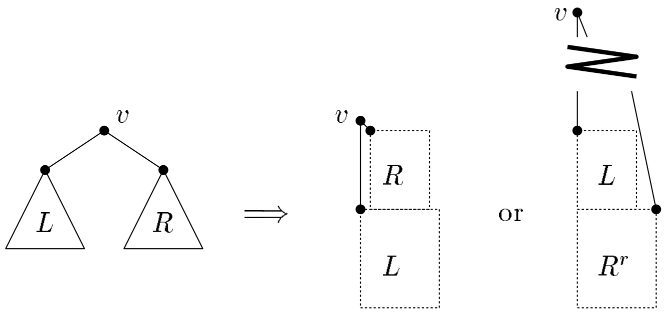

Figure 8: Another recursive algorithm to draw a binary tree, where the root is placed at the upper-left corner. The width satisfies the recurrence $W(n) \leq W(n/2) + 1$, implying $W(n) = O(\log n)$.

All our methods are practical to implement and take linear time. They all satisfy the following property in addition to the four "ideal" criteria:

- Disjoint subtrees are separable by horizontal lines in the drawings. (Thus, different parts of the tree can be quickly distinguished by the viewer.)

Drawings generated by the first three methods, but not the fourth, have in addition these nice properties:

- The bounding boxes of the line segments have disjoint interiors. (Thus, precision and aliasing problems are less of a concern to the viewer.)
- If v_1 and v_2 are children of v, then the angle $\angle v_1 v v_2$ cannot be too small (at least 45 degrees).

Moreover, drawings produced by the first two methods obey a curious property:

- Any horizontal line crosses at most one non-vertical line segment.

We can modify the first two methods to construct drawings in which the slope of each non-vertical line is either $+1$ or -1, without increasing the width; this feature enables the output to be displayed as an ASCII file (using the three characters `|`, `/`, and `\`).

By dynamic programming, one can compute in polynomial time the exact minimum area of drawings that utilize only the rules from Section 2 or 5 for a given binary tree. In practice, the aspect ratio (i.e., the height-to-width ratio) can be reduced by adding a rule that stacks drawings of subtrees horizontally rather than vertically.

References

[1] T. M. Chan, M. T. Goodrich, S. R. Kosaraju, and R. Tamassia. Optimizing area and aspect ratio in straight-line orthogonal tree drawings. In *Graph Drawing (Proc. GD '96)*, Lect. Notes in Comput. Sci., vol. 1190, Springer-Verlag, pages 63–75, 1997.

[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 and P. Penna. Upward drawings of ordered search trees. *Theoret. Comput. Sci.*, 203:51–67, 1998.

[4] P. Crescenzi, P. Penna, and A. Piperno. Optimal-area upward drawings of AVL trees. *Comput. Geom. Theory Appl.*, 9:25–42, 1998.

[5] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, 1998.

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

[7] S. K. Kim. Logarithmic width, linear area upward drawing of AVL trees. *Inform. Process. Lett.*, 63:303–307, 1997.

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

[9] E. Reingold and J. Tilford. Tidier drawing of trees. *IEEE Trans. Software Engineering*, SE-7(2):223–228, 1981.

[10] Y. Shiloach. Arrangements of planar graphs on the planar lattice. Ph.D. Thesis, Dept. of Applied Mathematics, Weizmann Institute of Science, Israel, 1976.

[11] C.-S. Shin, S. K. Kim, and K.-Y. Chwa. Area-efficient algorithms for upward straight-line tree drawings. In *Proc. 2nd Int. Computing and Combinatorics Conf. (COCOON'96)*, Lect. Notes in Comput. Sci., vol. 1090, Springer-Verlag, pages 106–116, 1996.

[12] R. Tamassia. Graph drawing. In *Handbook of Discrete and Computational Geometry* (J. E. Goodman and J. O'Rourke, ed.), pages 815–832, CRC Press, 1997.

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

[14] L. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. on Computers*, C-30:135–140, 1981.

A Appendix: Refining the Second Method

This appendix demonstrates that the $O(\sqrt{n})$ width bound from Section 4 is not optimal even without extending the left and right rules. Intuitively, the reason is this: the worst-case bound for the drawing method occurs when all four subtrees α_i, β_i, L_i, and R_i in Figure 4(b) have size near $n/4$, but in this "critical" case, there are alternative drawings that yield smaller bounds.

Formally, an $o(\sqrt{n})$ area bound can be shown by choosing a path that minimizes the expression $|\alpha|^p + |\beta|^p$ instead of the sum $|\alpha| + |\beta|$ in Lemma 4.1, for a suitable value of $p < 1/2$.

Lemma A.1 *Let $p = 0.48$. Given any binary tree T of size n, there exists a root-to-leaf path π such that for any left subtree α and right subtree β of π, $|\alpha|^p + |\beta|^p \leq (1-\delta)n^p$ for some constant $\delta > 0$.*

Proof: We modify the path construction of Section 4. As before, v_i, L_i, and R_i denote the root, left subtree, and right subtree of T_i, where initially $T_0 = T$. As before, α_i (resp. β_i) denotes the largest left (resp. right) subtree of $\langle v_0, \ldots, v_i\rangle$. We will maintain the invariant that $|\alpha_i|^p + |\beta_i|^p \leq (1-\delta)n^p$ for every i.

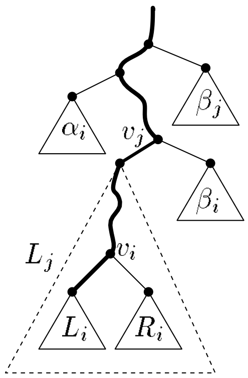

Figure 9: Proof of Lemma A.1 (*Case 4*).

Case 1: $|\alpha_i|^p + |R_i|^p \leq (1-\delta)n^p$ and $|L_i|^p + |\beta_i|^p > (1-\delta)n^p$. Set $T_{i+1} = L_i$. Clearly, the invariant remains true as we increment i.

Case 2: $|\alpha_i|^p + |R_i|^p > (1-\delta)n^p$ and $|L_i|^p + |\beta_i|^p \leq (1-\delta)n^p$. Set $T_{i+1} = R_i$. Again, the invariant remains true as we increment i.

Case 3: $|\alpha_i|^p + |R_i|^p \leq (1-\delta)n^p$ and $|L_i|^p + |\beta_i|^p \leq (1-\delta)n^p$. We terminate the construction as follows:

Consider the first subcase where $|L_i| \leq |R_i|$. Set the path π to be the concatenation of $\langle v_0, \ldots, v_i \rangle$ with the leftmost path from v_i down to a leaf. A left subtree of this path π has size at most $|\alpha_i|$, whereas a right subtree of π has size at most $\max\{|R_i|, |\beta_i|\}$. Thus, the lemma holds.

The other subcase, $|L_i| > |R_i|$, can be handled in a symmetric fashion (by choosing instead the rightmost path from v_i down to a leaf).

Case 4: $|\alpha_i|^p + |R_i|^p > (1-\delta)n^p$ and $|L_i|^p + |\beta_i|^p > (1-\delta)n^p$. We claim that this case is not possible.

Without loss of generality, suppose that the parent of β_i's root, which we will denote by v_j, is lower than the parent of α_i's root. By construction, we have a third inequality $|L_j|^p + |\beta_j|^p > (1-\delta)n^p$. In addition, $|L_i| + |R_i| \leq |L_j|$ and $|\alpha_i| + |\beta_i| + |L_j| + |\beta_j| \leq n$. See Figure 9. We can derive a contradiction for the value $p = 0.48$ (with a sufficiently small δ) by applying Hölder's inequality:

$$\begin{aligned}
2.5(1-\delta)n^p &< |\alpha_i|^p + |\beta_i|^p + |L_i|^p + |R_i|^p + 0.5|L_j|^p + 0.5|\beta_j|^p \\
&\leq |\alpha_i|^p + |\beta_i|^p + 2^{1-p}(|L_i| + |R_i|)^p + 0.5|L_j|^p + 0.5|\beta_j|^p \\
&\leq |\alpha_i|^p + |\beta_i|^p + (2^{1-p} + 0.5)|L_j|^p + 0.5|\beta_j|^p \\
&\leq (2 + (2^{1-p} + 0.5)^{1/(1-p)} + 0.5^{1/(1-p)})^{1-p}\,(|\alpha_i| + |\beta_i| + |L_j| + |\beta_j|)^p \\
&< 2.499n^p.
\end{aligned}$$

□

Combining the above lemma with equation (1) yields a recurrence of the form

$$W(n) \leq \max_{n_1^p + n_2^p \leq (1-\delta)n^p} (W(n_1) + W(n_2) + 2),$$

which solves to $W(n) = O(n^p)$ by induction. We conclude that it is possible to get $O(n)$ height and $O(n^{0.48})$ width using only the left and right rules of Section 2.

The exponent $p = 0.48$ is certainly not the best possible. By examining various ways to draw the "critical" case (*Case 4*), one can throw in additional inequalities (taking into account more variables) so as to lower the value of p necessary to derive a contradiction. However, the analysis becomes more complex and the improvement is at the moment rather small, so details of this line of attack will be omitted here. Still, it is intriguing to see what the optimal worst-case area bound is for this restricted type of drawings; perhaps, a different approach is called for.