Ideal Drawings of Rooted Trees With Approximately Optimal Width

Therese Biedl

Abstract

For rooted trees, an ideal drawing is one that is planar, straight-line, strictly-upward, and order-preserving. This paper considers ideal drawings of rooted trees with the objective of keeping the width of such drawings small. It is not known whether finding the minimum width is NP-hard or polynomial. This paper gives a 2-approximation for this problem, and a 2Δ-approximation (for ∆-ary trees) where additionally the height is $O(n)$. For trees with $\Delta \leq 3$, the former algorithm finds ideal drawings with minimum width.
1 Introduction

Let $T$ be a rooted tree. An upward drawing of $T$ is one in which the curves from parents to children are $y$-monotone. It is called strictly-upward if the curves are strictly $y$-monotone. In this paper, all drawings must be planar (no edges cross), and order-preserving (the drawing respects a given order of children around a node). Usually drawings should be straight-line (edges are drawn as straight-line segments). A tree-drawing is called an ideal drawing [5] if it is planar, strictly-upward, straight-line, and order-preserving.

To keep drawings legible, nodes are required to be placed at grid-points (i.e., have integer coordinates), and the main objective is to minimize the width and height of the required grid. In a strictly-upward drawing of a rooted tree, the height can never be smaller than the (graph-theoretic) height of the tree, and so may well be required to be $\Omega(n)$. Hence for such drawings the main objective is to minimize the width.

Previous Results: Any $n$-node tree has a planar straight-line strictly-upward drawing of area $O(n \log n)$ [6], but these drawings are not order-preserving. The best bounds for ideal drawings are by Chan [5]: he gives ideal drawing of area $O(n^{4/\sqrt{2 \log n}})$ whose width is $O(2^{1+\sqrt{2 \log n}})$. He mentions that the width can be reduced to $O(\log n)$ at the cost of super-polynomial height. For binary trees, Garg and Rusu showed that $O(\log n)$ width and $O(n)$ height can be achieved [10]. This is optimal (within the class of binary trees with $n$ nodes) since there are binary trees that require width $\Omega(\log n)$ and height $\Omega(n)$ for any upward drawing [6]. See the recent overview paper by Frati and Di Battista [7] for many other related results.

It is not known whether $O(n \log n)$ area can be achieved for ideal drawings of rooted trees. If the condition on straight-line drawings is relaxed to allow poly-line drawings (i.e., edges may have bends, as long as the bends are on grid-points), then a minor modification of the construction of Chan achieves order-preserving planar strictly-upward drawings with $O(n \log n)$ area [5].

This paper focuses on algorithms to find ideal drawings with small width. It is not known whether finding ideal drawings that have the optimal width (for the given tree) is NP-hard or polynomial. In a recent paper [4], I showed that finding minimum-width drawings is feasible if either the “order-preserving” or the “straight-line” condition is dropped from the conditions on ideal drawings, but neither of these two algorithms seems to generalize to minimum-width ideal drawings. If “upward” is dropped, then one can minimize the smaller dimension (then usually chosen to be the height) for unordered drawings [1] and approximate it for order-preserving drawings [2].

Results of this paper: This paper gives two approximation-algorithms for the width of ideal tree-drawings. The first one is a 2-approximation, which is quite similar to Chan’s approach [5], but uses the so-called rooted pathwidth $rpw(T)$ (the width of a minimum-width unordered upward drawing [4]) to find a path along which to split the tree and recurse.

However, the method to construct these drawings relies on first construct-
ing an $x$-monotone poly-line drawing and then stretching it into a straight-line drawing. This generally results in extremely large height, and in fact, one can argue that for some trees exponential height is required for drawings of optimal width. It hence makes sense to allow more width if this reduces the height drastically. This motivates the second algorithm of this paper, which creates drawings whose width may be a factor of $2\Delta$ away from the optimum, but where the height is $O(n)$. Here, $\Delta$ is an upper bound on the number of children that a node has. In particular, this gives ideal drawings of area $O(\Delta n \log n)$; the existence of such drawings was previously shown only for binary trees by Garg and Rusu [10]. With a minor modification, the algorithm achieves width $2\text{rpw}(T) - 1 \leq 2\log(n+1) - 1$ for binary trees, while the one by Garg and Rusu used width up to $3\log n$.

2 Preliminaries

A rooted tree $T$ consists of $n$ nodes $V$, of which one has been selected to be the root, and all non-root nodes have a unique parent in such a way that the root is the ancestor of all other nodes. We say that $T$ has arity $\Delta$ if all nodes have at most $\Delta$ children. A binary (ternary) tree is a tree with arity 2 (3). A node without children is called a leaf. A root-to-leaf path is a path from the root to some leaf.

For any node $v$, we use $T_v$ to denote the subtree of $T$ consisting of all descendants of $v$ (including $v$ itself). We assume that for each node a specific order of the children has been fixed. We usually use $c_1, \ldots, c_d$ for the children of the root, enumerated from left to right.

A drawing of $T$ maps each node $v$ to a grid-point with integer coordinates. The width (height) of such a drawing is the smallest integer $W$ ($H$) such that (after possible translation) all used grid-points have $x$-coordinate ($y$-coordinate) in $\{1, \ldots, W\}$ ($\{1, \ldots, H\}$). The grid-line with $x$-coordinate ($y$-coordinate) $i$ is called column $i$ (row $i$). All drawings are required to be planar (i.e., no two edges cross), upward (i.e., parents have no smaller $y$-coordinate than their children, and edges are drawn $y$-monotonically) and order-preserving (i.e., at every node the edges to the children appear in the prescribed left-to-right order at the node). The drawings we construct are in fact strictly upward, i.e., parents have strictly larger $y$-coordinate than their children and edges are drawn strictly $y$-monotone. We usually consider straight-line drawings where edges are represented by straight-line segments between their endpoints, but occasionally relax this to poly-line drawings, where edges may have bends, as long as these bends are also at grid-points and the curve of the edge remains $y$-monotone. An ideal drawing is planar, strictly-upward, order-preserving and straight-line. We often identify the graph-theoretic concept (node, edge, subtree) with the geometric feature (point, poly-line, drawing) that represents it.

Crucial for our construction is the so-called rooted pathwidth $\text{rpw}(T)$ of a tree $T$ [1]. We set $\text{rpw}(T) := 1$ if $T$ is a path from the root to a (unique) leaf.
Else, we set

$$rpw(T) := \min_{P \subseteq T} \max_{T' \subseteq T - P} \{1 + rpw(T')\},$$

where the minimum is taken over all root-to-leaf paths $P$ in $T$ and the maximum is taken over all subtrees that remain after removing the nodes of $P$ from $T$. Figure 1 illustrates this concept.

![Figure 1: A rooted tree with $rpw(T_v)$ indicated for all subtrees $T_v$. The rpw-main path is bold. Example taken from [4].](image)

A root-to-leaf path $P$ is called an rpw-main-path if the minimum in (1) is achieved when using $P$. If the root has a child $c$ with $rpw(T_c) = rpw(T)$, then any rpw-main-path must contain child $c$ for (1) to hold. Therefore there can exist at most one child $c$ of the root with $rpw(T_c) = rpw(T)$. If such a child exists, then we call it the rpw-heavy child of the root.

It follows from the lower-bound argument in [6] (and was shown explicitly in [4]) that any planar upward drawing of a tree $T$ has width at least $rpw(T)$, even if the drawing is neither straight-line nor order-preserving.

### 3 A 2-approximation

This section details an algorithm to create ideal drawings of width $2\cdot rpw(T) - 1$, hence a 2-approximation for the width. This algorithm is very similar to the one hinted at by Chan in his remarks [5]: the only difference is that we choose the “heavy” child to be the rpw-heavy-child, rather than the one whose subtree is biggest.

We first construct a poly-line drawing with the additional requirements that edges are drawn $x$-monotonically. Then we “straighten out” such a drawing to become a straight-line drawing, at the cost of increasing the height.

**Theorem 1** Any tree $T$ has an order-preserving planar strictly-upward drawing of width at most $2\cdot rpw(T) - 1$. Furthermore, every edge is drawn $x$-monotonically, and the height is at most $2n - \ell(T)$, where $\ell(T)$ denotes the number of leaves of $T$. 

Proof: We prove a slightly stronger statement, namely, we can create such a drawing such that the root is in the top-left corner. (A symmetric construction gives a drawing where the root is in the top-right corner.) Clearly the claim holds for a single node, so assume that the root has children. We know that there can be at most one rpw-heavy child $c_h$ with $\text{rpw}(T_{c_h}) = \text{rpw}(T)$. Set $W := 2\text{rpw}(T) - 1$; we aim to create a drawing within columns $1, \ldots, W$.

Case 1: There is no rpw-heavy child, or $c_h = c_1$: Recursively draw the subtree at each child with the root at the top-left corner. Recall that $\text{rpw}(T_{c_i}) \leq \text{rpw}(T) - 1$ for $i > 1$ and hence the drawings have width at most $2\text{rpw}(T_{c_i}) - 1 \leq 2\text{rpw}(T) - 3 = W - 2$. We may assume, after padding with empty columns on the right, that the drawing of $T_{c_i}$ has width exactly $W - 2$ for $i > 1$. The drawing of $T_{c_1}$ has width at most $W$; for ease of description we pad it with empty columns on the right so that it has width exactly $W$.

Combine these drawings with the “standard” construction of drawing trees already used in [5, 6]. Thus, place the root in the top-left corner. Place the drawings of $T_{c_1}, \ldots, T_{c_d}$, in this order from top to bottom, within columns $2, \ldots, W - 1$. Since the root is in column 1 and each $c_i$ (for $i > 1$) is in column 2, the edge to $c_i$ can be drawn straight-line. Place the drawing of $T_{c_1}$ below all the other drawings, in columns 1, \ldots, $W$. We can connect the edge from the root to $c_1$ going vertically down. See Figure 2(a).

![Figure 2](a) (b)

Figure 2: A 2-approximation algorithm for the width.
Case 2: $c_h \neq c_1$: Draw $T_{c_h}$ recursively with the root in the top-right corner, and draw $T_{c_i}$ for $i \neq h$ recursively with the root in the top-left corner. Pad the drawings of $T_{c_h}$ with columns on the left so that it has width exactly $W$, and pad the drawing of $T_{c_i}$ for $i \neq h$ with columns on the right so that it has width $W - 2$.

Place the root in the top-left corner. Place the drawings of $T_{c_d}, \ldots, T_{c_1}$, in this order from top to bottom, in columns 2, \ldots, $W - 1$, except omit the drawing of $T_{c_h}$ and leave one row empty in its place. As before one argues that these drawings fit and that we can connect the root to each $c_i$ for $i \neq h$. Place the drawing of $T_{c_h}$ below all the other drawings. We can connect the edge from the root to $c_h$ (while maintaining the order of the children) with two bends, by using the empty row between $T_{c_{h-1}}$ and $T_{c_{h+1}}$ and column $W$. See Figure 2(b).

In both cases the height of the drawing is the sum of the heights of the subtrees, plus one row for the root and (possibly) one row for the first bend. Hence it is at most $1 + \sum_{i=1}^{d} (2(n(T_{c_i}) - \ell(T_{c_i}))) + 1 = 2(n-1) - \ell(T) + 2 = 2n - \ell(T)$ as desired. 

**Corollary 1** Every tree $T$ has an ideal drawing of width at most $2\text{rpw}(T) - 1$.

**Proof:** By the previous theorem $T$ has a planar strictly-upward order-preserving poly-line drawing of this width such that edges are drawn $x$-monotonically. It is known \[8, 9, 11\] that such a drawing can be turned into a straight-line drawing without increasing the width. Neither of these references discusses whether strictly-upward drawings remain strictly-upward, but it is not hard to show that this can be done for the construction of Theorem 1: essentially each subtree needs to “slide down” far enough to allow bends to be straightened out.

Since $T$ requires width at least $\text{rpw}(T)$ in any upward planar drawing \[4\], this gives the desired 2-approximation algorithm. Since $\text{rpw}(T) \leq \log(n + 1)$ \[4\], this also re-proves the remark by Chan \[5\] that trees have order-preserving strictly-upward planar drawings of area $O(n \log n)$ and ideal drawings of width $O(\log n)$. Unfortunately the height of these ideal drawings may be very large, and so the area is no improvement on the area of $O(4\sqrt{\log n})$ achieved by Chan \[5\] for straight-line order-preserving upward drawings. It remains open to find ideal drawings of area $O(n \log n)$ for trees with arbitrary arity. (For bounded arity, such drawings will be constructed below.)

### 3.1 Ternary trees

For ternary trees, a minor change to the construction yields optimum width.

**Theorem 2** Every ternary tree $T$ has an order-preserving strictly-upward drawing of optimal width $\text{rpw}(T)$ and height $\frac{3}{4}n - \frac{1}{4}$ such that every edge is drawn $x$-monotonically.
We show something slightly stronger: \( T \) has such a drawing, and the root is either placed at the top-left or at the top-right corner. The choice between these two corners depends on the structure of the tree (i.e., it can not be chosen by the user). Clearly the claim holds for a single-node tree \( T \), so assume that \( T \) consists of a root \( v \), with children \( c_1, \ldots, c_d \), in order from left to right. Set \( W := rpw(T) \).

Recursively draw each sub-tree \( T_{c_i} \) with width \( rpw(T_{c_i}) \). We know that there can be at most one rpw-heavy child \( c_h \) with \( rpw(T_{c_h}) = rpw(T) = W \); for all other children \( c_i \) we have \( rpw(T_{c_i}) \leq rpw(T) - 1 = W - 1 \). If there is no rpw-heavy child, then define \( h := 1 \). Pad all the drawings with empty columns so that the drawing of \( T_{c_i} \) has width \( W - 1 \) if \( i \neq h \) and width \( W \) if \( i = h \). The empty columns are added on the opposite side (left or right) from the root, so that the root continues to be in the top-left or top-right corner.

As before we distinguish by \( h \), but in contrast to before we use the location of \( c_h \) in the drawing of \( T_{c_h} \) to determine where to put the root of \( T \).

**Case 1:** \( c_h = c_1 \): In this case the construction is almost exactly as for Theorem 1 (Case 1): the root is in the top-left corner and the subtrees are placed in columns \( 2, \ldots, W \), except for subtree \( T_{c_1} \), which occupies all columns. However, it may now be that for \( i = 1, 2, 3 \) tree \( T_{c_i} \) has its root \( c_i \) in the top-right corner. If needed, we hence use one bend (and, for \( i = 2 \), an extra row) to connect from the root to \( c_i \); this gives an \( x \)-monotone drawing. See Figure 3(a).

**Case 2:** \( c_h = c_d \): In this case the construction is symmetric: the root is in the top-right corner. See Figure 3(b).

**Case 3:** All remaining cases: We know that \( d \leq 3 \) since the tree is ternary. If \( d \leq 2 \) then one of the previous two cases applies, so we have \( d = 3 \). Also, we have \( h = 2 \), else one of the previous cases would apply. We know that in the drawing of \( T_{c_2} \) node \( c_2 \) is placed in one of the top corners.

**Case 3a:** \( c_2 \) is in the top-right corner: In this case the construction is similar to the one for Case 2 of Theorem 1. Place the root \( v_r \) in the top-left corner, place \( T_{c_3} \) in columns \( 2, \ldots, W \), place bends for edge \((v_r, c_2)\), place \( T_{c_1} \) in columns \( 1, \ldots, W - 1 \), and finally place \( T_{c_2} \) and connect the edge \((v_r, c_2)\). Two bends suffice for \((v_r, c_2)\) since \( c_2 \) is in the top-right corner, and so the edge is \( x \)-monotone. See Figure 3(c).

**Case 3b:** \( c_2 \) is in the top-left corner: In this case the construction is symmetric: the root is in the top-right corner. See Figure 3(d).

Clearly the height is at most \( \frac{4n - 1}{3} = 1 \) if \( n = 1 \). If \( n > 1 \) and we needed no extra row for bends, then the height is at most \( 1 + \sum_{i=1}^{d} \left( \frac{2}{3} n(T_{c_i}) - \frac{1}{3} \right) \leq \frac{4}{3} n - \frac{1}{3} \) since \( \sum_{i=1}^{d} n(T_{c_i}) = n - 1 \). If \( n > 1 \) and we did need an extra row for bends, then \( d = 3 \) and therefore the height is at most

\[
2 + \sum_{i=1}^{3} \left( \frac{4}{3} n(T_{c_i}) - \frac{1}{3} \right) = 2 + \frac{4}{3} (n - 1) - 1 = \frac{4}{3} n - \frac{1}{3}
\]

as desired.
ideal tree-drawings

(a) Case 1.  (b) Case 2.  (c) Case 3a.  (d) Case 3b.

Figure 3: The construction of optimum-width drawings for ternary trees.

As before, bends in $x$-monotone curves can be “straightened out” by sliding down, and so we have:

**Corollary 2** Every ternary tree $T$ has an ideal drawing of optimum width $rpw(T)$.

It is worth mentioning that the corollary cannot be generalized to higher arity: There exists a 4-ary tree $T$ with $rpw(T) = 2$ that has no ideal drawing of width 2 [4].

### 3.2 Super-polynomial height

Corollary 2 makes no claim on the height. Indeed, the transformations to straight-line drawings might increase the height exponentially in general (see [3]), and as we show now, also for ideal drawings of trees.

These lower bounds actually hold in under slightly weaker assumptions; we can replace “strictly-upward” by “upward”, i.e., horizontal edges are allowed. For lack of a better name, we say that a tree drawing is *weakly-ideal* if it is order-preserving, planar, straight-line and upward (but not necessarily strictly-upward).

**Theorem 3** For any $i \geq 1$, there exists a ternary tree $T_i$ that has an ideal drawing of width $i$, any weakly-ideal drawing requires width at least $i$, and any weakly-ideal drawing of width $i$ has height at least $(i − 1)! \in n^{Ω(\log \log n)}$.

**Proof:** The proof is by induction on $i$. We define two such trees, $T_i^R$ and $T_i^L$, that in addition to the claim satisfy the following: In any weakly-ideal drawing of $T_i^L$ of width $i$, the root is in the top-left corner and some point in the rightmost column has vertical distance (i.e., difference in $y$-coordinate) at least $(i − 1)!$ from the root. $T_i^R$ is symmetric to $T_i^L$, and hence in any weakly-ideal drawing of width $i$ the root is at the top-right corner and some point in the leftmost column has vertical distance at least $(i − 1)!$ from the root.
For $i = 1$, let trees $T^R_i$ and $T^L_i$ consist of a root with one child, which obviously requires width at least 1. Fix some weakly-ideal drawing of width 1. Clearly the root is in the desired corner. The child cannot be in the same row as the root (even if horizontal edges are allowed) because the drawing has width 1. Therefore the vertical distance of the child from the root is at least 1 = 0!, and the child is in the leftmost and rightmost column as desired.

For $i \geq 2$, we only give the construction for $T^L_i$; the one for $T^R_i$ is symmetric. Tree $T^L_i$ consists of the root $v_r$ with three children $c_1, c_2, c_3$, see also Figure 4(a). Subtree $T_{c_1}$ is a complete binary tree of height $i$ with $2^i - 1$ nodes, while $T_{c_2}$ and $T_{c_3}$ are two copies of $T^R_{i-1}$. The idea of the proof is as follows. $T_{c_1}$ requires width $i$, and therefore “blocks” the other subtrees from using column 1. Therefore subtrees $T_{c_2}$ and $T_{c_3}$ are drawn with width $i - 1$, so the inductive hypothesis applies to them. This forces some of their vertices to be drawn in such a way that the height must increase by a factor of $i$ when combining them.

We first argue that $T^L_i$ requires width at least $i$. This holds because (as one shows easily by induction) the complete binary tree with $2^i - 1$ nodes has rooted pathwidth $i$. Therefore $rpw(T_{c_1}) = i$ and already $T_{c_1}$ requires width $i$.

Next, we argue that $T^L_i$ can be drawn with width $i$. To this end, find a straight-line drawing of $T_{c_1}$ with width $i$. This exists, because any tree $T$ has a (not necessarily order-preserving) planar strictly-upward drawing of width $rpw(T)$ [5]. Since $T_{c_1}$ is symmetric, this drawing is in fact order-preserving for $T_{c_1}$, so it has an ideal drawing of width $i$. Combining this drawing of $T_{c_1}$ with the (recursively obtained) drawings of $T^R_{i-1}$ for $T_{c_2}$ and $T_{c_3}$, one can easily create an ideal drawing of $T^L_i$ of width $i$. See also the poly-line drawing in Figure 4(a), which can be stretched to remove the bend without adding width.

Now fix an arbitrary weakly-ideal drawing of $T^L_i$ that uses exactly $i$ columns. Since $T_{c_1}$ requires width $i$, its drawing contains a point $p_1$ in the rightmost column. The path from root $v_r$ to $p_1$ must be below the drawings of $T_{c_2}$ and $T_{c_3}$ by the order-property, and hence blocks both $T_{c_2}$ and $T_{c_3}$ from using the leftmost column.

Hence for $k = 2, 3$, tree $T_{c_k}$ is drawn with width at most $i - 1$. Since $T_{c_k} = T^R_{i-1}$, therefore induction applies. So $c_k$ is drawn in the rightmost column (i.e., in column $i$), and the drawing of $T_{c_k}$ contains a point $p_k$ that is in the leftmost column of the induced drawing of $T_{c_1}$ (i.e., in column 2) and has vertical distance at least $(i - 2)!$ from $c_k$.

Now we can prove the bound on the height. Consider the edge from the root $v_r$ to $c_2$, which is drawn as a straight-line segment $v_r c_2$. By order-property, edge $(v_r, c_3)$ must leave $v_r$ to the right of edge $(v_r, c_2)$. Since $c_2$ and $c_3$ are both in the rightmost column, therefore $c_3$ must be above $c_2$. By upwardness $c_3$ is no higher than $v_r$, therefore $c_3$ is to the right of $v_r c_2$. By planarity and upwardness, and because $c_2$ is in the rightmost column, hence all of $T_{c_3}$ (and in particular node $p_3$) must be to the right of $v_r c_2$. Since $p_3$ is in column 2 and $c_2$ is in column $i$, this forces $v_r$ to be in column 1 as desired. Furthermore, $c_2$ must be low enough for $v_r c_2$ to be to the left of $p_3$. For ease of calculation, translate so that the root has $y$-coordinate 0. We hence must have $0 \geq y(c_3) \geq y(p_3) + (i - 2)!$, and hence $v_r c_2$ has slope less than $-(i - 2)!$. Since it covers a horizontal distance of $i - 1$,
hence the vertical distance of $c_2$ to the root is at least $(i - 1)!$ as desired.

This finishes the construction for $T^L_i$, and the argument that it requires height at least $(i - 1)!$. It remains to analyze the size of the tree and hence obtain the asymptotic bound. The number $N(i)$ of nodes of $T^L_i$ is the same as the number of nodes of $T^R_i$ and satisfies the recursive formula $N(1) = 2$ and $N(i) = 1 + 2^i - 1 + 2N(i - 1) = i 2^i$. To express $i$ in terms of the number of nodes $n := N(i)$, observe that

$$(\log(n) - \log \log(n)) 2^{\log(n) - \log \log(n)} < \log(n) \frac{n}{\log(n)} = n = N(i) = i 2^i,$$

so $i > \log(n) - \log \log(n)$. For sufficiently large $n$ the required height is at least

$$(i - 1)! > (\log n - \log \log n - 1)! \geq (\log n - 2) \frac{\log n - 1}{\log n} = n \frac{\log \log n - 2}{\log \log n - 4} \geq \omega(n)$$

as desired.

This proof requires arity at least 3, because we needed one subtree to force the width and two recursively constructed subtrees to increase the height-bound. We suspect that some super-linear bound holds even for binary trees, but this remains open.

**Conjecture 1** There exists a binary tree such that any optimum-width order-preserved upward drawing has height $\omega(n)$. 

Figure 4: (a) Tree $T^L_i$ for Theorem 3. (b) Tree $T$ for Theorem 4. For the lower bound for strictly-upward drawings, delete $c_d$.

In both drawings, we add bends to some edges for ease of drawing, but the edges are $x$-monotone and hence a straight-line drawing of the same width exists.
While the lower bound in Theorem 3 is asymptotically larger than any polynomial, it grows less than exponential. We now show that for trees higher arity, exponential height is sometimes required. Whether such exponential height is required also for some trees of constant maximum degree remains open.

**Theorem 4** For any \( d \geq 4 \), there exists a \( d \)-ary tree \( T \) with \( n = 6d - 3 \) nodes that has an ideal drawing of width 3, but any weakly-ideal drawing of width 3 is required to have height at least \( 3 \cdot 2^{d-2} - 1 = 3 \cdot 2^{(n-9)/6} - 1 \).

**Proof:** We construct tree \( T \) as follows (see also Figure 4(b)): The leftmost child \( c_1 \) is the root of a complete binary tree with 7 nodes which needs 3 units of width. The rightmost child \( c_d \) is a single node. All other children \( c_2, \ldots, c_{d-1} \) are the root of a subtree \( T' \) that satisfies the following: \( T' \) has an ideal drawing of width 2, but any weakly-ideal drawing of \( T' \) has the root in the right column, and there exists a node \( p \) in the left column and at least two rows below the root. One can easily show that the 6-node tree \( T' \) in Figure 4(b) satisfies this with the gray node as \( p \).

Fix an arbitrary weakly-ideal drawing of width 3 of tree \( T \). Since \( T_{c_i} \) requires width 3, there exists a point \( p_1 \) of \( T_{c_i} \) in column 3. The path from the root to \( p_1 \) blocks the leftmost column for all other subtrees, so \( T_{c_i} \) for \( i > 1 \) is drawn with width at most 2. For \( 1 < i < d \), therefore \( T_{c_i} \) is drawn with width 2, implying that \( c_i \) is in column 3 and there is a point \( p_i \) of \( T_{c_i} \) in column 2 that is at least two units below \( c_i \). The goal is to show that the vertical distance of \( p_i \) from the root increases exponentially with \( d - i \).

After possible translation, assume that the root \( v_r \) has \( y \)-coordinate 0. We also know that \( v_r \) is in column 1, because the line-segment \( v_r c_d \) must bypass point \( p_2 \), which is in column 2. We show that for \( 1 < i < d \) node \( p_{d-i+1} \) must be placed with \( y \)-coordinate at most \( -(3 \cdot 2^i - 3) \). Observe that \( c_{d-1} \) is strictly below the root since it is neither the leftmost nor the rightmost child. By assumption \( p_{d-1} \) is at least two units below \( c_{d-1} \), hence has \( y \)-coordinate at most \( -3 = -(3 \cdot 2^1 - 3) \).

For the induction step, assume \( i \geq 2 \) and that \( p_{d-i+1} \) is placed with \( y \)-coordinate at most \( -(3 \cdot 2^{i-1} - 3) \). The straight-line segment \( v_r c_{d-i} \) connects column 1 and 3 and by planarity and order-property must intersect column 2 at a point below \( p_{d-i+1} \). Let \( Y \) be the \( y \)-coordinate of this intersection, then \( Y < -(3 \cdot 2^{i-1} - 3) \) and the \( y \)-coordinate of \( c_{d-i} \) is \( 2Y < -(3 \cdot 2^i - 6) \). Since \( c_{d-i} \) has integral \( y \)-coordinate, therefore its \( y \)-coordinate is at most \( -(3 \cdot 2^i - 5) \). Since \( p_{d-i} \) is two units below, it has \( y \)-coordinate at most \( -(3 \cdot 2^i - 3) \) and the induction holds.

For \( p_2 = p_{d-(d-2)} \), we hence have \( y \)-coordinate at most \( -(3 \cdot 2^{d-2} - 3) \). The path from the root to point \( p_1 \) in \( T_{c_1} \) must pass below \( p_2 \), and so uses at least one more row. Since the root was at \( y \)-coordinate 0 and the height counts the number of rows, the height of the drawing therefore is at least \( 3 \cdot 2^{d-2} - 1 \). The

\[ \text{For strictly-upward drawings, this node can be omitted and the height lower-bound then becomes } 3 \cdot 2^{d-1} - 1 \text{ with } d = \frac{2n^2}{e}. \]
number of nodes in $T$ is $n = 1 + 7 + (d - 2)6 + 1 = 6d - 3$, so $d - 2 = (n - 9)/6$ which proves the claim.

$\square$

4 A 2$\Delta$-approximation with linear height

In 2003, Garg and Rusu [10] showed that every binary tree has an ideal drawing of width $O(\log n)$ and height at most $n$. However, their construction does not generalize to higher arity (unless one drops “upward”). We now give a different construction that achieves these bounds for any tree that has constant arity.

**Theorem 5** Every $\Delta$-ary tree $T$ has an ideal drawing of width at most $(2\Delta - 1) \cdot (\text{rpw}(T) - 1) + 1$ and height at most $n$.

In particular any rooted tree has an ideal drawing of area $O(\Delta n \log n)$; this is an improvement over the area-bound of $O(4^{\sqrt{\log n}} n)$ by Chan [5] for small (but bigger than constant) values of $\Delta$.

**Proof:** For ease of description, define shortcuts $r := \text{rpw}(T)$ and $W(i) := (2\Delta - 1)(i - 1) + 1$; we aim to create drawings of width at most $W(r)$. As before we create drawings where the root is in the top-left corner, and a symmetric construction places the root in the top-right corner.

If $r = 1$ then $W(1) = 1$ and $T$ is a path from the root to a single leaf. We can draw $T$ in a single column as desired. So assume $r > 1$, which means that $\Delta \geq 2$ and that the root has children $c_1, \ldots, c_d$, $1 \leq d \leq \Delta$. We know that there can be at most one child rpw-heavy child $c_h$ with $\text{rpw}(T_{c_h}) = \text{rpw}(T) =: W$.

**Case 1:** There is no rpw-heavy child, or $c_h = c_1$ In this case, draw the tree as in the “standard” construction, i.e., recursively obtain drawings of each $T_{c_j}$, $j = 1, \ldots, d$, with $c_j$ in the top-left corner and combine as in Figure 2(a). The drawing of $T_{c_1}$ has width at most $W(r)$ and the drawing of each $T_{c_j}$ for $j > 1$ has width at most $W(r-1) \leq W(r) - 1$, to which we add at most one unit width. Clearly all conditions are satisfied.

**Case 2:** $c_h \neq c_1$: The construction in this case is much more complicated (and quite different from Garg and Rusu’s). We use $W(r) = W(r-1) + 2\Delta - 1$ columns for our drawing, and split them into 3 groups as follows:

- The leftmost $\Delta - 1$ columns are called left-detour columns. The rightmost of the left-detour columns is called the left-overhang column.

- The next $W(r-1) + 1$ columns are the middle columns; the leftmost and rightmost of the middle columns are called the left-path and right-path column, respectively.

- The rightmost $\Delta - 1$ columns are called the right-detour column. The leftmost of the right-detour columns is called the right-overhang column.
Figure 5(a) sketches the construction. The main tool is to use a rpw-heavy path $P = v_0, v_1, v_2, \ldots$. Note that $v_1$ must be child $c_h$, and so in particular $v_1$ is not the leftmost child of $v_0$ by case assumption.

We first outline the idea. To place path $P$, we split it into many sub-paths of length at least 2. These sub-paths are alternatingly placed “on the left” (in the left-path column or the left-overhang column) and “on the right” (in the right-path column or the right-overhang column). Whenever possible, subtrees are placed in the middle columns. However, this is not always possible for the top-most and bottom-most node of a sub-path. For these, we use the detour-columns, either for placing the node or for placing its children. However, the subtrees at these nodes or children cannot be placed here; instead we put them “much farther down”, namely, at such a time when path $P$ has veered to the other side and therefore the middle columns are accessible.

The precise split into sub-paths is determined at the same time as we place the path $v_0, v_1, v_2, \ldots$. We maintain a counter $i$ and running indices $\ell_i$ and $r_i$. The $i$th left path consists of vertices $v_{\ell_{i-1}+1}, \ldots, v_{\ell_i}$ while the $i$th right path consists of vertices $v_{\ell_i+1}, \ldots, v_{r_i}$. In other words, $\ell_i$ and $r_i$ are the indices of the last vertex of the corresponding path.

Place the root $v_0$ in the top-left corner. Initialize $i = 1$ and set $\ell_1 = 0$. The first left path hence consists of just $v_0$; the first right path starts at $v_1$ (it ends at $v_{r_1}$, where $r_1$ will be determined below). We hence know that all nodes in $v_0, \ldots, v_\ell$ have been placed already. Now repeat:

- $v_{\ell_i+1}$ is placed in the right-overhang column, one row below $v_{\ell_i}$.
- $v_{\ell_i+2}$ is placed in the right-path column, some rows below.
- While $v_j$ is the rightmost child of $v_{j-1}$ (for $j = \ell_i + 3, \ell_i + 4, \ldots$), place it in the right-path column, some rows below.
- Let $r_i \geq \ell_i + 2$ be the maximal index for which $v_{r_i}$ was placed in the right-path column in the previous two steps. So $v_{r_i+1}$ is not the rightmost child of $v_{r_i}$.
- Place $v_{r_i+1}$ in the left-overhang column, one row below $v_{r_i}$.
- Place $v_{r_i+2}$ in the left-path column, some rows below.
- While $v_j$ is the leftmost child of $v_{j-1}$ (for $j = r_i + 3, r_i + 4, \ldots$), place it in the left-path column, some rows below.
- Let $\ell_{i+1} \geq r_i + 2$ be the maximal index for which $v_{\ell_{i+1}}$ was placed in the left-path column in the previous two steps.
- Update $i := i + 1$, and repeat until we reach the end of path $P$.

---

Some rows below" means "so that this node is below all the subtrees that need to be inserted above it by later steps". For this particular situation here, this is the height of the drawings of the subtrees at left children of $v_{\ell_i}$ and $v_{\ell_i+1}$ and (for $i > 1$) also of the subtrees at children of the left children of $v_{r_{i-1}}$.\footnote{Some rows below" means "so that this node is below all the subtrees that need to be inserted above it by later steps". For this particular situation here, this is the height of the drawings of the subtrees at left children of $v_{\ell_i}$ and $v_{\ell_i+1}$ and (for $i > 1$) also of the subtrees at children of the left children of $v_{r_{i-1}}$.}
Figure 5: The construction for order-preserving straight-line drawings if the rpw-
heavy child of the root is not the leftmost child. Path $P$ is purple and dashed. 
(a) The construction for arbitrary $\Delta$. The numbers on the right indicate the 
step during which these children/subtrees were placed. (b) The modified version 
for $\Delta = 2$. Some subtrees now use an overhang column.

For any non-leaf node $v$ on $P$, let the left (right) children of $v$ be all those
children of \(v\) that are strictly left (right) of the child of \(v\) on \(P\). We now explain how to place all the subtrees at right children of nodes \(v_{\ell_i}, \ldots, v_{\ell_i+1-1}\), for \(i = 1, 2, \ldots\). The subtrees at left children are placed symmetrically.

1. We start at \(v_{\ell_i}\). The right children of \(v_{\ell_i}\) are placed, in order, in the row below \(v_{\ell_i}\) and in distinct right-detour columns. By choice of \(\ell_i\) (or, for \(i = 1\), by case assumption) node \(v_{\ell_i+1}\) is not the leftmost child of \(v_{\ell_i}\). So \(v_{\ell_i}\) has at least one left child, therefore at most \(\Delta - 2\) right children, which means that there are sufficiently many right-detour columns for placing the right children as well as \(v_{\ell_i+1}\). Since these children are one row below \(v_{\ell_i}\), we can connect them to \(v_{\ell_i}\) with a straight-line segment (drawn curved in Figure 5(a) for increased visibility). The subtrees at these children are not being placed yet; this will happen in Step 6.

2. The next node is \(v_{\ell_i+1}\), which is in the right-overhang column one row below \(v_{\ell_i}\). The subtrees at its right children will be placed in Step 7.

3. The next nodes are \(v_{\ell_i+2}, \ldots, v_{r_i-1}\) (this set may be empty). By choice of \(r_i\) these nodes do not have right children. The rows for these nodes (as well as \(v_{r_i}\)) are determined by the symmetric version of Step 7 that places subtrees at left children.

4. The next node is \(v_{r_i}\), placed in the right-path column. We place the subtrees at its right children with the symmetric version of the standard construction of Figure 2(a). Thus, recursively obtain for each such subtree a drawing of width at most \(W(r-1)\) with the child in the top-right corner. Place these, in order, in the rows below \(v_{r_i}\) and in the columns to its left (except for the last child, which shares the column with \(v_{r_i}\)). This fits within the middle columns since there are \(W(r-1) + 1\) middle columns and \(v_{r_i}\) is in the rightmost of these.

5. Next comes node \(v_{r_i+1}\), in the row below \(v_{r_i}\) and the left-overhang column. This node might share a row with some right child of \(v_{r_i}\) but uses a different column. The subtrees at \(v_{r_i+1}\)’s right children will be will be placed later (in Step 6).

6. Now we place the three sets of subtrees that were deferred earlier.

   (a) First, draw the subtrees at right children of \(v_{\ell_i+1}\) recursively with their roots in the top-right corner. Place these drawings, flush right with the right-path column, below all the trees of right children of \(v_{\ell_i}\). Recall that \(v_{\ell_i+1}\) was placed in the right-overhang column while its children are now in the right-path column, which is adjacent. Hence the edges can be drawn with straight-line segments (shown again with curves in Figure 5(a)).

   (b) Next, we parse the right children of \(v_{\ell_i}\) in left-to-right order. If \(c\) is such a child, then \(c\) was placed much higher up already in one of the right-detour columns. Let \(g_1, \ldots, g_d\) be the children of \(c\) (hence
T. Biedl  

Ideal Tree-Drawings

grand-children of \(v_{\ell_i}\). For \(k = 1, \ldots, d\), create a drawing of \(T_{g_k}\) with \(g_k\) in the top-right corner. Place these drawings in the middle columns as well as the right-detour columns so that \(g_1, \ldots, g_d\) are one column to the left of \(c\). Then \(c\) can be connected with straight lines (shown again with curves).

(c) Finally place the subtrees at right children of \(v_{r_i+1}\). Recursively draw each such subtree with the root in the top-left corner. Place these, in order, flush left with the left-path column, and draw the edges to \(v_{r_i+1}\) as straight-line segments.

7. Next come nodes \(v_j\) for \(j = r_i + 2, r_i + 3, \ldots, \ell_i+1 - 1\). For each \(j\), place \(v_j\) in the next row (i.e., the first row below what was drawn so far) and in the left-path column. Recursively draw the subtree at each right child of \(v_j\) with the root in the top-left corner. Place these, in order, flush left with the column that is one right of the left-path column.

8. Finally put \(v_{\ell_i+1}\) in the next row; and go to Step 1 with the next \(i\).

This ends the description of the construction, which has width \(W(r)\) as desired. All rows contain nodes, so the height is at most \(n\). \(\square\)

4.1 The special case of binary trees

We note that for binary trees, our construction gives a width of at most \(3rpw(T)\), hence a 3-approximation. This can be turned into a 2-approximation by decreasing the number of middle columns.

**Corollary 3** Every binary tree \(T\) has an ideal drawing of width \(2rpw(T) - 1 \leq 2\log(n + 1) - 1\) and height at most \(n\).

**Proof:** Define \(W'(r)\) recursively to be \(W'(1) = 1\) and \(W'(r) = W'(r-1) + 2\) (which resolves to \(W'(r) = 2r - 1\)). Apply exactly the same construction as before, using \(\Delta = 1\) overhang columns on each side, but use only \(W'(r-1)\) middle columns. See also Figure 5(b).

It remains to argue that we can do the construction using one less middle column per recursion. We show here only that the subtrees at right children “fit”; the argument is symmetric for the left children. This can be seen (for \(i = 1, 2, \ldots\)) as follows:

- Node \(v_{\ell_i}\) has no right child since the tree is binary and by choice of \(\ell_i\) it has a left child.

- The subtree at the right child of \(v_{\ell_i+1}\) (if any) has width at most \(W'(r-1)\) by induction. This fits into the middle columns. We can connect the child to \(v_{\ell_i+1}\) since the latter is in the right-overhang column, i.e., in an adjacent column.

- Node \(v_j\) with \(\ell_i + 2 \leq j \leq r_i - 1\) has no right child by choice of \(r_i\).
- Node $v_{r_i}$ has at most one right child since the tree is binary. This child is placed vertically below $r_i$, and hence its subtree can use all middle columns.

- The subtree at the right child of $v_{r_i+1}$ (if any) has width at most $W'(r-1)$ by induction. This fits into the middle columns. We can connect the child to $v_{r_i+1}$ since the latter is in the left-overhang column, i.e., in an adjacent column.

- Consider node $v_j$ for $j$ with $r_i + 2 \leq j \leq \ell_{i+1} - 1$, which is placed in the left-path column. The subtree at its right child may use $W'(r - 1)$ columns, but only $W'(r - 1) - 1$ of the middle columns are available for it, since the left-path column is used by $v_j$ and edge $(v_j, v_{j+1})$. However, at this $y$-range no node or edge uses the right-overhang column, so we can use the right-overhang column to place the subtree at the right child.

Hence the construction works, for binary trees, with only $W'(rpw(T))$ middle columns, and the width is hence at most $W'(rpw(T)) \leq 2rpw(T) - 1$. □

5 Concluding remarks

This paper gave approximation algorithms for the width of ideal drawings of trees, i.e., grid-drawings that are planar, strictly-upward, order-preserving and straight-line. It was shown that one can approximate the width within a factor of 2 (and even find the optimum width for ternary trees), albeit at the cost of a very large height. A second construction gave drawings with linear height for which the width is within a factor of $2\Delta$ of the optimum. In particular this implies ideal drawings of area $O(n \log n)$ for all trees with constant arity.

The algorithms implicit in the proofs clearly take polynomial time. With suitable data structures (storing drawings of subtrees as a black-box that only stores the size of the bounding box and the offset relative to some references point), the time to combine the $d$ subtrees of the root is $O(d)$, resulting in linear run-time overall. This assumes that arbitrarily large coordinates can be handled in constant time (a non-trivial assumption for those straight-line drawings where the height may be exponential).

Among the most interesting open problems is whether it is possible to find the minimum width of ideal tree-drawings in polynomial time. Secondly, what can be said if the height should be small? Does every rooted tree have a strictly-upward straight-line order-preserving drawing of area $O(n \log n)$, or is it possible to prove a lower bound of $\omega(\log n)$ width if (say) at most $n$ rows may be used?
References


