Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 143 (2004) 318–323
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Notes

Order-preserving, upward drawing of binary trees using fewer bends

Sung Kwon Kim

Department of Computer Science and Engineering, Chung-Ang University, 221 Huksuk-dong Dongjak-ku, Seoul 156-756, Republic of Korea

Received 30 April 2003; received in revised form 29 December 2003; accepted 9 February 2004

Abstract

For a binary tree with n nodes, we present a planar, polyline, order-preserving, upward, grid drawing that has $O(n/\log n)$ bends and matches the previously best area $O(n \log n)$. This is an improvement over the $O(n \log n)$ area drawing, which has $O(n)$ bends.

Keywords: Bends; Order-preserving; Tree drawing; Upward drawing

1. Introduction

A drawing of a binary tree T maps each node of T to a distinct point in the plane and each edge (u, v) of T to a chain of line segments with endpoints u and v. A *planar*, *polyline*, *order-preserving*, *upward*, *grid* drawing of a binary tree is a drawing satisfying the following conditions:

- (*Planar*) No two edges intersect.
- (*Polyline*) Each edge is drawn as a polygonal chain. A polygonal chain may have *bends* which occur when joining two adjacent line segments in the chain.
- (*Order-preserving*) The edge to the left child of a node is drawn to the left of the edge to its right child.
- (*Upward*) Each edge drawn is vertically monotone; so no node is allowed to locate above its parent, though they may lie on the same horizontal line.
- (*Grid*) The nodes and the bends of the edges have integer coordinates.

Planar and grid will be omitted hereafter as they are common to the drawings mentioned in this paper. The *height*, *width*, and *area* of a drawing are the height, width, and area of the smallest orthogonal rectangle containing the drawing.

2. Previous works

Before reviewing previous works, we need to define terminology on binary trees. Let T be a binary tree with n nodes. The root of T is at *level* 0. If a node is at level i, then its children are at level $i + 1$. If the maximum of the levels of the nodes in T is h, then the *depth* of T is $h + 1$. A *fragment* of T denotes a partial tree of T. In other words, if T is considered to be a graph, a fragment of T corresponds to a connected subgraph. A node and all of its descendants form a *subtree* of T rooted at the node. A subtree is a fragment, but the reverse is not necessarily true.

E-mail address: skkim@cau.ac.kr (S.K. Kim).

0166-218X/$ - see front matter
doi:10.1016/j.dam.2004.02.011

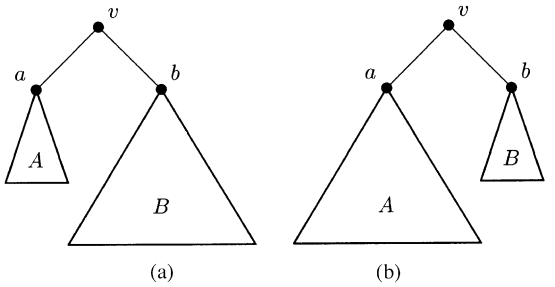

Fig. 1. A binary tree: the root and two subtrees.

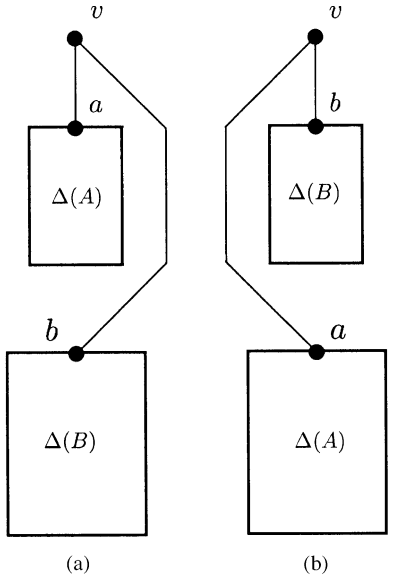

Fig. 2. Order-preserving drawing by the algorithm of Lemma 2.

Lemma 1. *Given a binary tree with n nodes and a constant* α, $0 < \alpha < 1$, *a polyline, upward drawing with* $O(n)$ *area,* $O(n^{\alpha})$ *width, and* $O(n^{1-\alpha})$ *height can be computed in* $O(n)$ *time.*

A proof can be found in [2]. Drawings by the algorithm of Lemma 1 are not order-preserving, and can have $O(n)$ bends. To obtain order-preserving drawings, more than $O(n)$ area is required as shown in the following lemma:

Lemma 2. *Given a binary tree with n nodes, a polyline, order-preserving, upward drawing with* $O(n \log n)$ *area,* $O(\log n)$ *width, and* $O(n)$ *height can be computed in* $O(n)$ *time. Moreover, there are binary trees that require* $\Omega(n \log n)$ *area.*

The algorithm in [2] for Lemma 2 can be recursively described. As in Fig. 1, let v be the root of binary tree T, and let A and B be its left and right subtrees, whose roots are a and b, respectively. Recursively draw A and B and obtain their drawings $\Delta(A)$ and $\Delta(B)$. To draw T, stack $\Delta(A)$ and $\Delta(B)$ vertically and place v above them so that v, a, and b are on a common vertical line. Between $\Delta(A)$ and $\Delta(B)$, the one with fewer nodes lies on top of the other. Now, two edges (v,a) and (v,b) are to be connected. If $\Delta(A)$ lies above $\Delta(B)$, then as in Fig. 2(a), (v,a) is drawn vertically, and (v,b) detours to the right of $\Delta(B)$ as b is the right child of v. Fig. 2(b) shows the case when $\Delta(B)$ lies above $\Delta(A)$.

In these drawings the nodes of T are all on a common vertical line. Since v, a and b are placed on a vertical line, if we recursively apply this to A and B it is easy to verify. So, the height of the drawings is $O(n)$. One of (v,a) and (v,b) is drawn vertically, and the other passes either to the right of or to the left of the top drawing, thus increasing its width. The width of the bottom drawing remains unchanged. This is why the subtree with fewer nodes is placed on top of the other; the subtree with fewer nodes usually gives a drawing with smaller width.

Let $w(T)$ be the width of the drawing T. Then, $w(T) \leqslant \max\{w(A)+1, w(B)\}$ for Fig. 2(a), and $w(T) \leqslant \max\{w(A), w(B)+1\}$ for Fig. 2(b). Generalizing this, if we let $width(n)$ be the maximum possible width of any binary tree with n nodes, then $width(n) \leqslant \max\{width(n_1), width(n_2)+1\}$, where $n_1 \geqslant n_2$, $n = n_1 + n_2 + 1$ and n_1 and n_2 denote the sizes of the two subtrees. If we solve this using $width(1) = 0$, we have $width(n) = O(\log n)$ and thus the width of a drawing computed by Lemma 2 is $O(\log n)$.

Let us count the number of bends in these drawings. One of (v,a) and (v,b) is a vertical line and has no bends, and, however, the other has either one or two bends as it passes around the top drawing. For a binary tree in which each internal node has exactly two children, one of two edges connecting a parent to its children must have a bend, and so, $n/2$ edges have bends. Hence, the number of bends in drawings obtained by Lemma 2 is $O(n)$. Since [2] mainly focuses on minimizing the area of drawings, there is no mention of the number of bends in [2] and Lemma 2.

In this paper we will show that the number of bends can be reduced to $O(n/\log n)$, still achieving $O(n \log n)$ area, using the same drawing conventions, i.e., polyline, order-preserving, upward drawings.

Notice that if we drop both polyline and order-preserving conditions, upward drawings with $O(n \log n)$ area and no bends are possible [1].

3. Reducing bends in polyline, order-preserving, upward drawings

We will employ the following well-known technique. If we remove i edges from a binary tree T with n nodes, then T will be partitioned into $i+1$ fragments. Each fragment is also a binary tree. Shin et al. [3] proved the following lemma and applied it in drawing algorithms.

Lemma 3. *We can remove $\theta(n/\log n)$ edges from T so that each of the remaining $\theta(n/\log n)$ fragments has $O(\log n)$ nodes. Moreover, this can be done in $O(n)$ time.*

It is important that the fragments have almost equal number of nodes. Lemma 3 can be used in drawing T in the following way.

1. Partition T into $\theta(n/\log n)$ fragments, each of size $O(\log n)$, by removing edges as in Lemma 3.
2. Draw the fragments using the method in Section 3.1.
3. Determine the layout of the fragment drawings using the layout method in Section 3.2.
4. Restore the "removed" edges and draw them among the fragment drawings using the restoring method in Section 3.2.

For each fragment obtained by Lemma 3, merge its nodes and edges into a *supernode*. The removed edges connect supernodes, i.e., a removed edge $e = (u,v)$ connects two supernodes that contain u and v, respectively. Then we have a tree FT of $\theta(n/\log n)$ supernodes. FT is called a *fragment* tree.

An important result given by Shin et al. [3] is that the removal of edges in Lemma 3 can be done so that the following lemma holds. A fragment that has only one node is called *trivial*. A supernode from a trivial fragment is also called *trivial*.

Lemma 4. (1) *FT is a binary tree.* (2) *Only trivial supernodes can have two children in FT.*

In other words, Lemma 4(2) implies that a non-trivial supernode can have at most one child. This plays an important role in reducing the area or the number of bends when the removed edges are restored and drawn. In FT, a non-trivial supernode has at most two edges: one connects the supernode to its parent and the other, if any, connects the supernode to its child. In T, the former edge connects the root of the non-trivial fragment to a node of another fragment, and the latter edge connects a node of the non-trivial fragment (called a *connection* node) to the root of another fragment.

3.1. Drawing fragments

We will explain how to draw fragments here. Since a trivial fragment, having a single node, is easy to draw, we will concentrate on drawing non-trivial fragments. Let F be a non-trivial fragment with m nodes. Since F is a binary tree, we may use the terminology defined for binary trees. We first introduce a *straight-line*, order-preserving, upward drawing of F with both width and height $O(m)$ and area $O(m^2)$. In this drawing each edge is a straight-line that has no bends. Let h be the depth of F. For each level i, $0 \leqslant i \leqslant h-1$, assign *order numbers* $1, 2, \ldots$ from left to right to the nodes at the level. If a node is at level i and its order number is j, then it is drawn at position $(j, h-i)$. Each edge is drawn as a

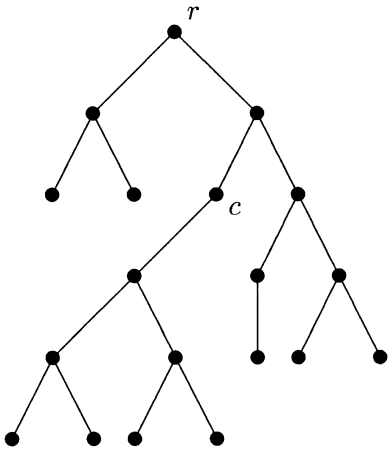

Fig. 3. An example binary tree.

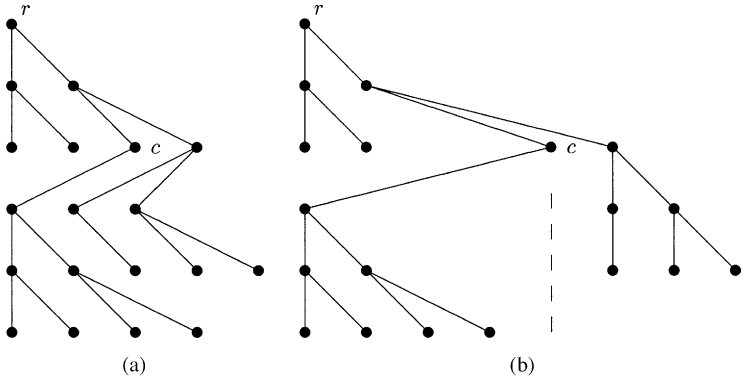

Fig. 4. (a) A straight-line, order-preserving, upward drawing. (b) Preparing a channel for a connection node.

straight-line connecting its two endnodes. In this drawing, the root of F is at coordinate $(1, h)$, and its children, if both exist, are at $(1, h-1)$ and $(2, h-1)$. The other nodes are placed in a similar way.

Fig. 3 shows a binary tree (or a fragment) and its drawing is in Fig. 4(a). It is easy to verify that drawings so obtained are straight-line, order-preserving, and upward. The height of a drawing is h, and the width equals the number of nodes at a level having the most nodes. Hence, both the height and width are $O(m)$, and the area is $O(m^2)$.

Notice that in this drawing the root is at the top left position. Since the root is to be connected to a node of other fragment by an upward edge, its position is good for this. As mentioned before, a non-trivial fragment has at most one connection node. So, if F has one, we need to prepare a channel through which the connection node connects to the root of another fragment.

Let c be the connection node of F. We will prepare a vertical channel immediately below c by moving those nodes and edges that currently obstruct the channel. Let r be the root of F, and let P be the path between r and c in F. Partition the nodes in $F-P$ into L and R in the following way. Let w be the first node in P when we walk from a node $v \in F-P$ to r. If $w=c$, put v into L. Otherwise, let u be the child of w not in P. If u is the left child of w, put v into L; otherwise, put it into R.

Let k be the level of c in F. We are to determine new positions for the nodes of F.

- The nodes at levels $0, 1, \ldots, k-1$, and those at levels $k, k+1, \ldots, h$ and in L remain at the same positions as before.
- The new position of c is $(q+1, h-k)$, where

$$q = \max_{i=k+1,\ldots,h} \{\text{number of nodes in } L \text{ and at level } i\}.$$

- A node in R and at level $i = k, k+1, \ldots, h$ is newly positioned at $(q+j+1, h-i)$, where j is its order number at level i, counting only the nodes in R.

In Fig. 4(b), since $q = 4$, the x-coordinate of c is $q+1 = 5$. In this new drawing c is at position $(q+1, h-k)$ and the downward ray from c does not intersect any part of the new drawing. Along this ray (dashed in Fig. 4(b)) the edge from c will be drawn.

Since this new drawing at most doubles its width, its area is doubled, but is still $O(m^2)$.

Lemma 5. *For a non-trivial fragment with m nodes a straight-line, order-preserving, upward drawing of both height and width* $O(m)$ *and area* $O(m^2)$ *can be computed in* $O(m)$ *time. Moreover, if it has a connection node, then the downward ray from the connection node does not intersect any part of the drawing.*

3.2. Layout of fragment drawings and restoring removed edges

In this section we will explain how to determine the layout of the fragment drawings and to draw the removed edges among them to complete the whole drawing of a binary tree.

Let $\Delta(F)$ be the drawing of F obtained by Lemma 5. If F is trivial then $\Delta(F)$ consists of a single node. For a non-trivial fragment F, $\Delta(F)$ is of both height and width $O(\log n)$ as $m = O(\log n)$ by Lemma 3.

By Lemma 4, FT is a binary tree with $\theta(n/\log n)$ supernodes. Applying Lemma 2 to FT results in a drawing $\Delta(FT)$ of height $O(n/\log n)$, width $O(\log n)$, and area $O(n)$. Remember that in $\Delta(FT)$ the supernodes are all vertically stacked on a common vertical line, denoted by ℓ. Each supernode in $\Delta(FT)$ corresponds to a fragment. To obtain $\Delta(T)$ each supernode in $\Delta(FT)$ is replaced by its fragment drawing, $\Delta(F)$, so that the left side of $\Delta(F)$ is on ℓ. This completes the layout of the fragments.

Now, we will explain how the removed edges are drawn between the fragment drawings. From now on, supernodes in FT or $\Delta(FT)$ and fragments corresponding to them will be used interchangeably.

If F is trivial, then $\Delta(F)$ consists of a single node, f. Let A and B denote the children of F, and let a and b be the roots of A and B, respectively. The two edges (F, A) and (F, B) in $\Delta(FT)$ will be replaced by two edges (f, a) and (f, b) in $\Delta(T)$. As in Fig. 5 one of them is drawn as a straight line and the other is drawn as a polyline having one or two bends.

If F is non-trivial, it has at most one child. Let A, with root a, be the child of F in FT, and let c be the connection node of F. We need to connect edge (c, a), replacing edge (F, A) in $\Delta(FT)$. In $\Delta(FT)$, F and A are adjacent, i.e.,

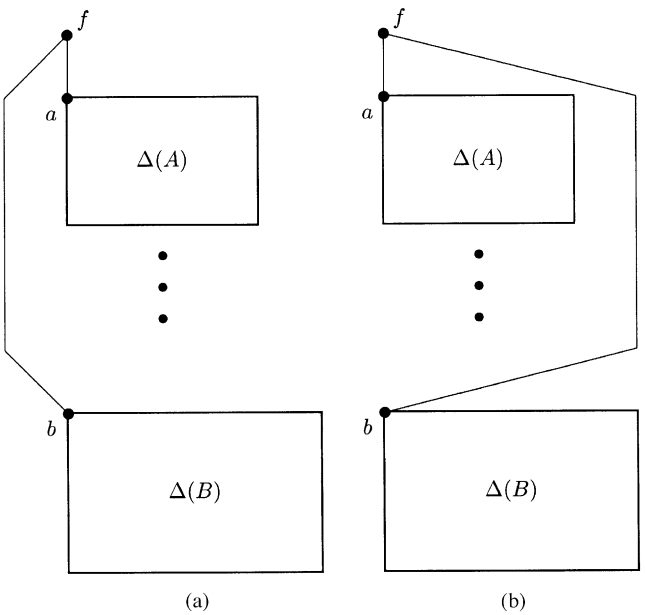

Fig. 5. Connecting edges from a trivial fragment.

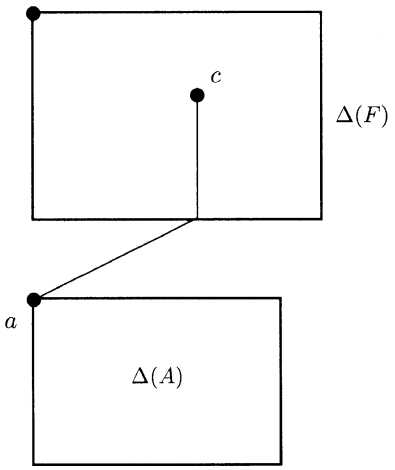

Fig. 6. Connecting edges from a non-trivial fragment.

F is immediately above A. So, $\Delta(F)$ is immediately above $\Delta(A)$ in $\Delta(T)$. Let R be the smallest orthogonal rectangle containing $\Delta(F)$. From c, draw a downward vertical line until it intersects the bottom side of R (Lemma 5 guarantees this is possible), and from the intersection draw a straight line to a. This drawing of (c, a) has one bend. See Fig. 6.

Let us analyze the complexities of our drawings. The height of the resulting drawing is $O(n)$ as each of the stacked $O(n/\log n)$ supernodes in $\Delta(FT)$ is replaced by a drawing of height $O(\log n)$, but its width is still $O(\log n)$ as another $O(\log n)$ width is added to the width of $\Delta(FT)$. So, the area is $O(n \log n)$.

Theorem 1. *For a binary tree with n nodes, a polyline, order-preserving, upward drawing with* $O(n/\log n)$ *bends can be computed in* $O(n)$ *time. Moreover, the drawing is of height* $O(n)$, *width* $O(\log n)$, *and area* $O(n \log n)$.

Proof. Height, width, and area have been analyzed, and it is easy to see that time complexity is $O(n)$. For the number of bends, each fragment drawing, $\Delta(F)$, has no bends as it is a straight-line drawing, and only the edges connecting the fragments can have at most two bends each as in Figs. 5 and 6. So, there are $O(n/\log n)$ bends. □

References

[1] P. Crescenzi, G. Di Battista, A. Piperno, A note on optimal area algorithms for upward drawings of binary trees, Comput. Geom. 2 (1992) 187–200.

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

[3] C.S. Shin, S.K. Kim, K.Y. Chwa, Area-efficient algorithms for upward straight-line tree drawings, Comput. Geom. 15 (4) (2000) 175–202.