ELSEVIER

Theoretical Computer Science 203 (1998) 51–67

Theoretical Computer Science

Strictly-upward drawings of ordered search trees

P. Crescenzi[a,*], P. Penna[b]

[a] *Dipartimento di Sistemi ed Informatica, Università di Firenze, Via C. Lombroso 6/17, 50134 Firenze, Italy*

[b] *Dipartimento di Scienze dell'Informazione, Università degli Studi di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy*

Abstract

We prove that any logarithmic binary tree admits a linear-area straight-line strictly-upward planar grid drawing (in short, strictly-upward drawing), that is, a drawing in which (a) each edge is mapped into a single straight-line segment, (b) each node is placed below its parent, (c) no two edges intersect, and (d) each node is mapped into a point with integer coordinates. Informally, a logarithmic tree has the property that the height of any (sufficiently high) subtree is logarithmic with respect to the number of nodes. As a consequence, we have that k-balanced trees, red–black trees, and BB[α]-trees admit linear-area strictly-upward drawings. We then generalize our results to logarithmic m-ary trees: as an application, we have that B-trees admit linear-area strictly-upward drawings.

Keywords: Graph drawing; Upward drawing; Area requirement

1. Introduction

In several applications, information is better displayed by a graphical representation emphasizing its structure in a readable way. The automatic design of these graphical representations is one of the main motivations for the growing interest in the research area of *graph drawing* whose typical problem is the following: given a graph G, produce a geometric representation of G according to some graphic standards and optimization criteria.

Several graphic standards and optimization criteria have been proposed in the literature depending on the application at hand. The annotated bibliography maintained by Di Battista et al. mentions most of them and refers to more than 300 papers in this research area [7]. In this paper, we are interested in *straight-line strictly-upward planar grid drawings*, in short *strictly-upward drawings*, of rooted trees, that is, drawings in which each edge is mapped into a single straight-line segment, each node is

* Corresponding author. E-mail: piluc@dsi.unifi.it.

0304-3975/98/$19.00
PII S0304-3975(97)00287-9

placed below its parent, no two edges intersect, and each node is mapped into a point with integer coordinates. Each of these standards is naturally justified by "readability" considerations [7].

A natural and important optimization criterion for evaluating these drawings is that they take as little area as possible where the area of a drawing equals the area of the smallest isothetic rectangle bounding the drawing. This criterion belongs to the family of the so-called *aesthetic* criteria [7] which are based on the fact that some drawings are better than others in conveying information regarding the tree.

1.1. Previous results

The first $O(n \log n)$-area algorithm to produce a strictly-upward drawing of a binary tree appeared in [12] and recently in [4] it has been proved that this algorithm is optimal (it is worth observing that, if we relax the upwardness requirement, then any binary tree of n nodes admits a linear-area planar grid drawing [13]). In [4] the authors also gave two algorithms producing a linear-area strictly-upward drawing of complete and Fibonacci binary trees, respectively. In [9], it has been proved that if we allow an edge to be represented by a chain of straight-line segments and a node to be on the same horizontal line as its parent, then *any* binary tree can be drawn in linear area. Subsequently, in [6] it has been shown that any AVL tree admits a linear-area strictly-upward drawing. In [2] the authors proved, among other things, that bounded-degree trees in some classes of balanced trees admit an $O(n \log \log n)$-area strictly-upward drawings: these classes include k-balanced trees, red–black trees, and BB[α]-trees. In [9] it has been proved that binary trees admit an $O(n \log \log n)$-area orthogonal upward drawings and that such an area is optimal. Finally, in [1] also the aspect ratio of orthogonal upward drawings has been considered.

1.2. Our results

In Section 2 we show that, for any "logarithmic" binary tree t with n nodes, a strictly-upward drawing of t can be produced with area $O(n)$ in time $O(n)$. Informally, a logarithmic tree has the property that the height of any (sufficiently high) subtree is logarithmic with respect to the number of nodes. For example, several binary search trees such as k-balanced trees, red–black trees, and BB[α]-trees are logarithmic trees. In particular, we prove that, for any constant $\alpha > 1$, a constant κ exists such that any logarithmic binary tree with n nodes can be strictly-upward drawn in any rectangle whose shorter side is at least $\log^{\alpha} n$ and whose area is equal to κn. To this aim, we make use of the top-down approach developed in [6] suitably modified in order to deal with the case of logarithmic trees. Observe that the bound on the length of the shorter side allows a very great flexibility to applications that need to draw a binary tree in a prespecified rectangular region: indeed, the only requirement is essentially that the length of the shorter side is a little bit greater than the height of the tree.

In Section 3, we extend the previous result to the case of logarithmic m-ary trees. In particular, we prove that, for any constant $\alpha > 1$, two constants λ and κ exist such that

any logarithmic binary tree with n nodes can be strictly-upward drawn in any rectangle whose shorter side is at least $\lambda \log^{\alpha} n$ and whose area is equal to κn and we show that, for any m, the class of m-ary B-trees admits linear-area strictly-upward drawings. In order to prove this result, we introduce the new notion of h–v–d^{m-2} drawing which is an extension of that of h–v drawing introduced in [4] and which is used as an intermediate drawing towards the strictly-upward one.

All our results apply to the case of ordered trees.

1.3. Preliminaries

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

We refer to directed rooted ordered trees. We denote by e the empty tree. Given m trees $t_1, \ldots, t_m$, we denote by $t_1 \oplus \cdots \oplus t_m$ the tree whose immediate subtrees are $t_1, \ldots, t_m$ (note that since t is ordered, the $\oplus$ operator is not necessarily commutative). The definition of k-balanced trees, red–black trees, BB[α]-trees and B-trees can be found in any of the several textbooks on the design and analysis of algorithms such as [3, 11].

A *straight-line strictly-upward ordered planar grid drawing*, in short *strictly-upward drawing*, of a tree t is a drawing of t such that:

(i) Edges are straight-line segments.
(ii) Each node has an ordinate greater than that of its parent (we are thus assuming that the y-axis is downward oriented).
(iii) Edges do not intersect.
(iv) Nodes are points with integer coordinates.
(v) The order of the tree is preserved, that is, for any subtree $t' = t'_1 \oplus \cdots \oplus t'_m$ of t, the root of t'_i has abscissa smaller than that of t'_j, for any i, j with $i < j$.

The width (respectively, height) of a drawing is the width (respectively, height) of the smallest isothetic rectangle bounding the drawing. We adopt the convention that both the width and the height are measured by the number of grid points, so that any drawing of a nonempty tree has both width and height greater than zero. The *area* of a drawing is then defined as the product of its width and height.

An *h–v drawing* of a binary tree is a strictly-upward drawing in which only rightward-horizontal and downward-vertical straight-line segments are allowed. More precisely, an h–v drawing of a non-empty binary tree $t = t_1 \oplus t_2$ is obtained by one of the two operations illustrated in Fig. 1 where δ_1 and δ_2 are two h–v drawings of t_1 and t_2, respectively. In the first operation, that is, the horizontal operation, δ_2 is translated to the right by as many grid points as the width of δ_1 and δ_1 is translated to the bottom by one grid point. The semantics of the second operation, that is, the vertical operation, is defined similarly.

In [4] the following lemma has been shown (see also Lemma 10).

Lemma 1. *Any h–v drawing of a binary tree t of area A can be transformed into a strictly-upward drawing of t of area at most $2A$.*

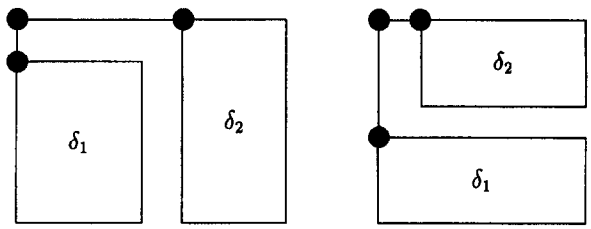

Fig. 1. The two operations of an h–v drawing.

2. Binary search trees

In this section we give a sufficient condition to apply the techniques of [6] in order to obtain a linear-area strictly-upward drawing of a binary tree. This condition will then be applied to show that several classes of search trees admit strictly-upward drawings with area $O(n)$. These classes include k-balanced trees, red–black trees, and BB[α]-trees.

Let us first introduce the notion of a class of (h_0, β)-logarithmic trees.

Definition 2. A tree t is (h_0, β)-*logarithmic* if, for any subtree t' of t whose height is greater than h_0,

$$n' \geqslant 2^{\beta h'},$$

where n' and h' denote the number of nodes and the height of t', respectively. A class of trees is said to be (h_0, β)-*logarithmic* if any of its members is an (h_0, β)-*logarithmic* tree.

It is well-known that the class of k-balanced trees is $(1, 1/k)$-logarithmic, that the class of red–black trees is $(1, \frac{1}{2})$-logarithmic, and that the class of BB[α]-trees is $(1, \log 1/(1-\alpha))$-logarithmic.

In order to apply the techniques of [6], we have to define, for each class of (h_0, β)-logarithmic trees, two functions which denote a lower bound on the smaller size of the rectangle in which the tree has to be drawn and the constant factor in the area function, respectively. Indeed, these two functions are similar to those used in [6] and are defined as follows:

$$l(h) = h^{\alpha},$$

where $\alpha > 1$ and

$$k(h+1) = \begin{cases} k_0 & \text{if } 1 \leqslant h < h_0, \\ k(h)\left(1 + \frac{2}{h^{\alpha}}\right) & \text{otherwise,} \end{cases}$$

where k_0 is a constant that will be specified later.

Clearly, for any h, $l(h+1) \geqslant l(h) + 1$. Moreover, the following four lemmas, whose proofs are contained in the appendix, hold.

Lemma 3. *A constant κ exists such that*

$$\lim_{h\to\infty} k(h) = \kappa.$$

Lemma 4. *For any h and for any n_1 and n_2 with $n_1 \leqslant n_2$,*

$$k(h+1)n \geqslant k(h)(n_1+n_2) + \max\left\{\frac{k(h+1)n - k(h)n_2}{l(h+1)}, \frac{k(h)n_2}{l(h+1)-1}\right\},$$

where $n = n_1 + n_2 + 1$.

Lemma 5. *Let C be an (h_0, β)-logarithmic class of binary trees. A constant $\bar{h} \geqslant h_0$ exists such that, for any $t \in C$ of height $h+1 > \bar{h}+1$ with n nodes,*

$$\frac{k(h+1)n(1-1/l(h)) - k(h)n_2}{\sqrt{k(h+1)n}} \geqslant l(h),$$

where n_2 is the number of nodes of the immediate subtree of t with larger number of nodes.

Lemma 6. *Let C be an (h_0, β)-logarithmic class of binary trees. A constant $\hat{h} \geqslant h_0$ exists such that, for any $t \in C$ of height $h+1 > \hat{h}+1$ with n nodes,*

$$\frac{k(h)n_2}{\sqrt{k(h+1)n}} \geqslant l(h),$$

where n_2 is the number of nodes of the immediate subtree of t with larger number of nodes.

We are now ready to prove the main result of this section. As stated before, the proof is based on the top-down approach used in [6].

Theorem 7. *Let C be an (h_0, β)-logarithmic class of binary trees. For any $t \in C$ of height h with n nodes, t can be h–v drawn within any rectangle R whose smaller dimension is at least $l(h)$ and whose area is equal to $k(h)n$.*

Proof. The proof is by induction on h. For $h \leqslant \max\{\bar{h}, \hat{h}\}$ where $\bar{h}$ and $\hat{h}$ are the values specified by Lemmas 5 and 6, respectively, the proof is straightforward: indeed, since the number of trees with bounded height is finite, it is sufficient to choose k_0 big enough (that is why we did not specify the constant k_0).

Let $h > \max\{\bar{h}, \hat{h}\}$ and let us assume that the theorem is true for any height less than $h+1$. Moreover, let l and L denote the smaller and the larger dimension of R, respectively, and let us assume that the longer side of R is the vertical one. Given an (h_0, β)-logarithmic tree t of height $h+1$ with n nodes we first map the root of t into the grid point whose coordinate are (x, y), where x and y denote the coordinates of the top leftmost corner of R. We then isolate two rectangles R_r and R_ℓ within the rectangle R as follows (see Fig. 2 reproduced from [5]).

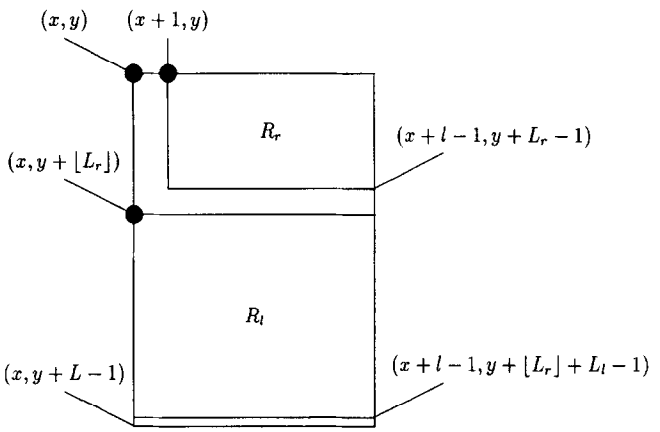

Fig. 2. The splitting of the rectangle R in the vertical case.

Definition (*of* R_r *and* R_ℓ). Let us denote the length of the vertical side of R_r and R_ℓ with L_r and L_ℓ, respectively. Let also be n_1 and n_2 be the number of nodes of the smaller and the larger immediate subtree of t, respectively.

If the larger subtree is the left one, then

$$L_\ell = \frac{k(h)n_2}{l}.$$

Otherwise,

$$L_\ell = L - \frac{k(h)n_2}{l-1}.$$

In both cases,

$$L_r = L - L_\ell.$$

The top leftmost corners of R_r and R_ℓ have coordinates $(x+1, y)$ and $(x, y+\lfloor L_r \rfloor)$, respectively. Moreover $l-1$ and l are their respective horizontal sides.

Shorter side bound. Let h_r and h_ℓ denote the heights of the two subtrees we are drawing within R_r and R_ℓ, respectively. We now shall prove that the shorter sides of R_r and R_ℓ have length at least $l(h_r)$ and $l(h_\ell)$, respectively. To this aim, we will repeatedly make use of the fact that $l \geqslant l(h+1)$ and of the fact that, since $l \leqslant L$ and $lL = k(h+1)n$, $l \leqslant \sqrt{k(h+1)n}$.

(i) The larger subtree is within R_ℓ.

(a) Rectangle R_r. If $L_r \geqslant l-1$, then

$$l - 1 \geqslant l(h+1) - 1 \geqslant l(h) \geqslant l(h_r).$$

Otherwise,

$$L_r = L - L_\ell = L - \frac{k(h)n_2}{l} = \frac{k(h+1)n - k(h)n_2}{l}$$

$$\geqslant \frac{k(h+1)n - k(h)n_2}{\sqrt{k(h+1)n}} > l(h) \geqslant l(h_r),$$

where the second-last inequality follows from Lemma 5.

(b) Rectangle R_ℓ. If $L_\ell \geqslant l$, then

$$l \geqslant l(h+1) > l(h_l).$$

Otherwise, we have that

$$L_\ell = \frac{k(h)n_2}{l} \geqslant \frac{k(h)n_2}{\sqrt{k(h+1)n}} \geqslant l(h) \geqslant l(h_\ell),$$

where the second inequality follows from Lemma 6.

(ii) The larger subtree is within R_r.

(a) Rectangle R_r. If $L_r \geqslant l-1$, then

$$l-1 \geqslant l(h+1)-1 \geqslant l(h) \geqslant l(h_r).$$

Otherwise,

$$L_r = \frac{k(h)n_2}{l-1} > \frac{k(h)n_2}{l} \geqslant \frac{k(h)n_2}{\sqrt{k(h+1)n}} \geqslant l(h) \geqslant l(h_\ell),$$

where the third inequality follows from Lemma 6.

(b) Rectangle R_ℓ. If $L_\ell \geqslant l$, then

$$l \geqslant l(h+1) > l(h_\ell).$$

Otherwise, we have that

$$L_\ell = \frac{L(l-1) - k(h)n_2}{l-1} = \frac{k(h+1)n - \frac{k(h+1)n}{l} - k(h)n_2}{l-1}$$
$$> \frac{k(h+1)\left(1 - \frac{1}{l(h)}\right)n - k(h)n_2}{\sqrt{k(h+1)n}} \geqslant l(h) \geqslant l(h_l),$$

where the second-last inequality follows from Lemma 5.

Area bound. From the definition of L_r and L_ℓ it follows that the area of the rectangle within which we are drawing the larger subtree is equal to $k(h)n_2$. Moreover, from Lemma 4, we have that the area of the other rectangle is at least $k(h)n_1$ (and thus the rectangle contains a subrectangle whose area is equal to $k(h)n_1$). Indeed, if the larger subtree is the left one, then the area of R_r is equal to $L_r(l-1)$ and

$$L_r(l-1) = \left(L - \frac{k(h)n_2}{l}\right)(l-1) = Ll - k(h)n_2 - \left(L - \frac{k(h)n_2}{l}\right)$$
$$= k(h+1)n - k(h)n_2 - \frac{k(h+1)n - k(h)n_2}{l}$$
$$\geqslant k(h+1)n - k(h)n_2 - \frac{k(h+1)n - k(h)n_2}{l(h+1)} \geqslant k(h)n_1,$$

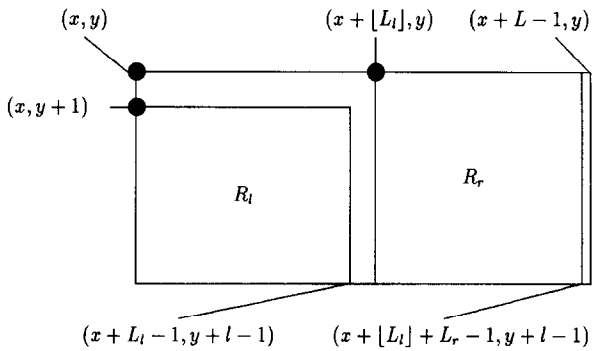

Fig. 3. The splitting of the rectangle R in the horizontal case.

where the second-last inequality follows from the fact that $l \geqslant l(h+1)$ and the last inequality follows from Lemma 4. Otherwise, that is, if the larger subtree is the right one, then the area of R_ℓ is equal to $L_\ell l$ and

$$\begin{aligned} L_\ell l &= \left(L - \frac{k(h)n_2}{l-1}\right) l = Ll - \frac{k(h)n_2}{l-1} l \\ &= k(h+1)n - k(h)n_2 - \frac{k(h)n_2}{l-1} \\ &\geqslant k(h+1)n - k(h)n_2 - \frac{k(h)n_2}{l(h+1)-1} \geqslant k(h)n_1, \end{aligned}$$

where the second-last inequality follows from the fact that $l \geqslant l(h+1)$ and the last inequality follows from Lemma 4.

We thus have that the inductive hypothesis is satisfied for both a rectangle contained in R_r and a rectangle contained in R_ℓ. That is, the right immediate subtree admits an h–v drawing within R_r and the left one within R_ℓ.

The proof in the case in which the longer side of R is the horizontal one is very similar (see Fig. 3). In this case, if the larger subtree is the left one, then $L_\ell = k(h)n_2/(l-1)$. Otherwise, $L_\ell = L - k(h)n_2/l$. In both cases, $L_r = L - L_\ell$. The top leftmost corners of R_r and R_ℓ have coordinates $(x + \lfloor L_\ell \rfloor, y)$ and $(x, y+1)$, respectively. Moreover, l and $l-1$ are their respective vertical sides. The rest of the proof is basically identical to the vertical case and the theorem thus follows. □

From the proof of the above theorem we can formally derive the algorithm to h–v draw any logarithmic binary tree in linear area. This algorithm is shown in Fig. 4 and is a slight modification of the algorithm described in [5]. In the figure, a rectangle is specified by the lengths l and L of its sides, by the coordinates x and y of its top leftmost corner which is always assumed to be a point with integer coordinates, and by a Boolean flag b which indicates the orientation of the longer side (if b is true then the longer side is vertical, otherwise it is horizontal).

```
procedure LT(R = ⟨l, L, (x, y), b⟩, t);
{construct h-v drawing of logarithmic binary tree t within rectangle R}
begin
    map the root of t into (x, y);
    h ← height of t;
    t1 ← immediate subtree of t with the smaller number of nodes;
    t2 ← immediate subtree of t with the larger number of nodes;
    n2 ← number of nodes of t2;
    if t2 is the left immediate subtree of t then
        if b then Ll ← L − k(h−1)n2 / l else Ll ← L − k(h−1)n2 / l−1
    else
        if b then Ll ← L − k(h−1)n2 / l−1 else Ll ← L − k(h−1)n2 / l;
    Lr ← L − Ll;
    if b then begin
        (xr, yr) ← (x + 1, y);
        (xl, yl) ← (x, y + ⌊Lr⌋);
        if Lr ≥ l − 1 then Rr ← ⟨l − 1, Lr, (xr, yr), b⟩
        else Rr ← ⟨Lr, l − 1, (xr, yr), not b⟩;
        if Ll ≥ l then Rl ← ⟨l, Ll, (xl, yl), b⟩
        else Rl ← ⟨Ll, l, (xl, yl), not b⟩;
    end;
    if not b then begin
        (xr, yr) ← (x + ⌊Ll⌋, y);
        (xl, yl) ← (x, y + 1);
        if Lr ≥ l then Rr ← ⟨l, Lr, (xr, yr), b⟩
        else Rr ← ⟨Lr, l, (xr, yr), not b⟩;
        if Ll ≥ l − 1 then Rl ← ⟨l, Ll, (xl, yl), b⟩
        else Rl ← ⟨Ll, l, (xl, yl), not b⟩;
    end;
    if t1 ≠ e then
        if t1 is the left immediate subtree of t then
            LT(Rl, t1)
        else
            LT(Rr, t1);
    if t2 ≠ e then
        if t2 is the left immediate subtree of t then
            LT(Rl, t2)
        else
            LT(Rr, t2);
end.
```

Fig. 4. The algorithm to h–v draw a logarithmic binary tree.

From Theorem 7, from Lemma 1 and from the fact that the class of k-balanced trees, the class of red–black trees, and the class of BB[α]-trees are all (h_0, β)-logarithmic for some h_0 and β, we have the following result.

Corollary 8. *Any k-balanced trees, any red–black trees, and any BB[α]-tree admits a strictly-upward drawing with linear area.*

3. Logarithmic m-ary trees

In this section we will extend the previous results to the case of m-ary trees. For the sake of clarity, we will describe our techniques in the case of ternary trees and we will only sketch how these techniques can be generalized for $m > 3$.

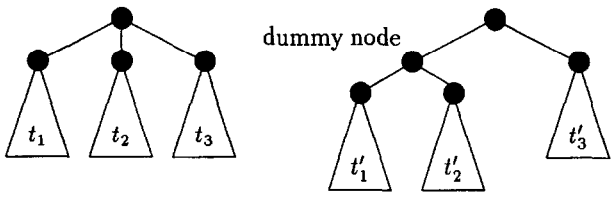

Fig. 5. From ternary to binary trees.

Basically, the linear-area strictly-upward drawing of a logarithmic ternary tree is obtained by the following four steps.

(i) Transform the ternary tree into a binary tree.
(ii) h–v draw the binary tree.
(iii) Transform the h–v drawing of the binary tree into an 'h–v–d drawing' of the ternary tree.
(iv) Transform the h–v–d drawing into a strictly-upward drawing.

3.1. *From ternary to binary trees*

The transformation from a ternary tree t to a binary tree t' is inductively defined as follows.

(i) If the height of t is 1 then $t' = t$.
(ii) If $t = t_1 \oplus e$ (respectively, $t = t_1 \oplus t_2$), then $t' = t'_1 \oplus e$ (respectively, $t' = t'_1 \oplus t'_2$).
(iii) If $t = t_1 \oplus t_2 \oplus t_3$, then t' is obtained from t'_1, t'_2, and t'_3 by adding a 'dummy' node as shown in Fig. 5.

Lemma 9. *If t is an (h_0, β)-logarithmic ternary tree, then the corresponding binary tree t' is $(2h_0, \beta/2)$-logarithmic.*

Proof. Let us first observe that any subtree of t of height h is transformed into a subtree of height h', with

$$h' \leqslant 2h - 1.$$

Moreover their roots are the same.

Let t'_u be any subtree of t' of height $h' > 2h_0$ with root u. If u is not a 'dummy' node, then the number of nodes of t'_u is at least the number of nodes of the corresponding tree t_u of t.

Since t is (h_0, β)-logarithmic and the height h of t_u is at least

$$\frac{h'+1}{2} > \frac{2h_0+1}{2} > h_0,$$

then the number of nodes of t'_u is at least

$$2^{\beta h} \geqslant 2^{\beta(h'+1)/2} > 2^{\beta h'/2}.$$

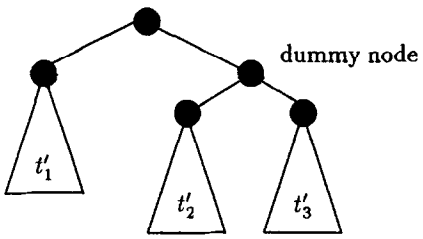

Fig. 6. The rearrangement of tree t'.

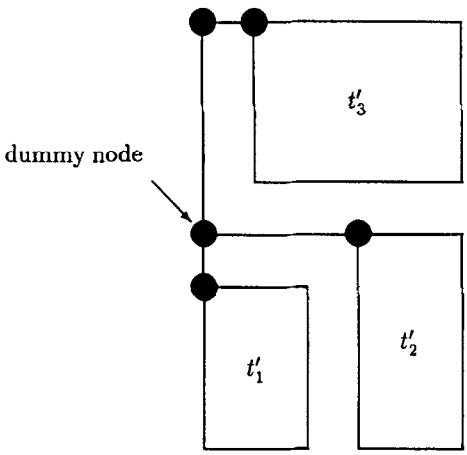

Fig. 7. A 'bad' h–v drawing of t'.

If u is a 'dummy' node, then one of its immediate subtrees has height $h' - 1$ and a 'non-dummy' root. By reasoning similarly to the previous case, we have that the number of nodes of t'_u is at least $2^{\beta h'/2}$. The lemma thus follows. □

3.2. *h–v drawing of the binary tree*

Once we have reduced the ternary tree t into the binary tree t', we can now apply the technique of Section 2 in order to obtain a linear-area h–v drawing of t'. However, in order to successively make possible the removal of the dummy nodes of t' without introducing edge-crossings, it may be necessary to perform a rearrangement of the children of the root of the tree to be drawn.

In particular, assume that procedure **LT** described in Fig. 4 is called with parameters R and t'. If the orientation of the longer side of R is vertical (that is, b is true) and the left child of the root of t' is a dummy node (see the right part of Fig. 5), then t' has to be modified as shown in Fig. 6. Indeed, if we do not perform this rearrangement, the h–v drawing of t' obtained by the procedure could be as shown in Fig. 7. In this case, directly connecting the root of t' with the root of t'_2 could generate an edge crossing.

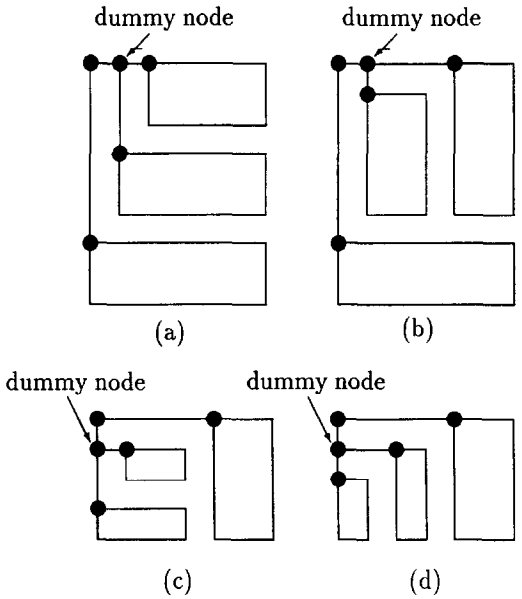

Fig. 8. The h–v drawings of t'.

By implementing this modification of procedure **LT**, we are then sure that, for any node u' of t' which has a 'dummy' child, the h–v drawing of the subtree $t'_{u'}$ rooted at u' is obtained in one of the four ways shown in Fig. 8.

3.3. *From h–v drawings to h–v–d drawings*

The h–v–d drawing standard is an extension of the h–v drawing one in the sense that we allow a restricted set of 'diagonal' rightward-downward segments to be used between one node and one of its children. More formally (but still intuitively), if δ_1, δ_2, and δ_3 are the h–v–d drawings of three ternary trees t_1, t_2, and t_3, respectively, then an h–v–d drawing of $t = t_1 \oplus t_2 \oplus t_3$ can be obtained in one of the four ways shown in Fig. 9.

Observe that these four operations basically correspond to the four h–v drawings shown in Fig. 8. Indeed, it is now easy to transform any of these h–v drawings into an h–v–d drawing: to this aim, it suffices (a) to directly connect the root with its middle child, (b) to delete the 'dummy' node, and (c) to eventually shift the h–v drawing of either the leftmost or the rightmost immediate subtree. Clearly, the area of the resulting h–v–d drawing is no more than the area of the original h–v drawing.

3.4. *From h–v–d drawings to strictly-upward drawings*

As in the case of h–v drawings, h–v–d drawings turn out to be a useful tool to produce intermediate drawings towards the final area-efficient strictly-upward drawings.

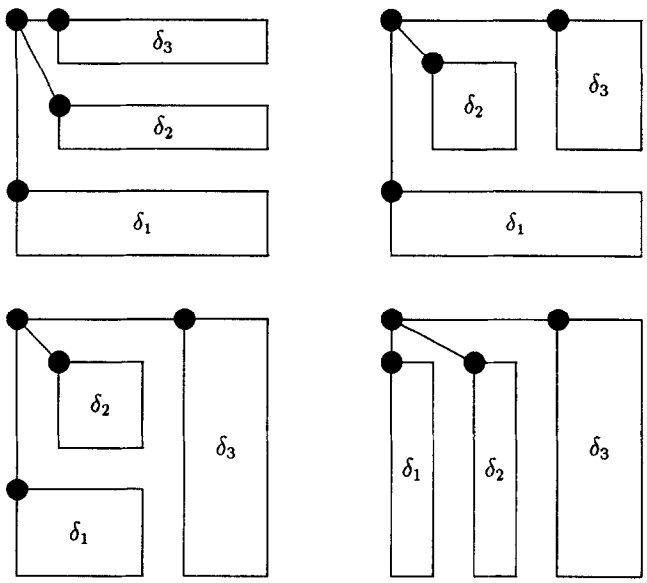

Fig. 9. The h–v–d operations.

Indeed, the following fact shows that transforming an h–v–d drawing into a strictly-upward drawing increases the area of the drawing by at most a factor of 2.

Lemma 10. *Any h–v–d drawing δ of height h and width w can be transformed in linear time into a strictly-upward drawing δ' of height $h+w$ and width w.*

Proof. Let us assume that the y-axis is downward oriented. We simply substitute each point (x,y) of δ with the point $(x,x+y)$. Clearly, the resulting drawing is strictly-upward and its height and width are equal to $h+w$ and w, respectively. It is also easy to see that this transformation preserves the planarity of the drawing. □

From the above construction and from Theorem 7, we have the following result.

Theorem 11. *Any (h_0,β)-logarithmic ternary tree can be strictly-upward drawn in linear area.*

3.5. Linear-area strictly-upward drawings of logarithmic m-ary trees

Theorem 11 can be generalized to the case of m-ary trees by slightly modifying steps (i) and (iii). More precisely, the transformation of an m-ary tree t into a binary tree t' is similar to that of ternary trees, with the only difference that, for any node with $k \leqslant m$ children, we add $k-2$ 'dummy' nodes.

The following fact can thus be proved similarly to Lemma 9.

Lemma 12. *If t is an (h_0, β)-logarithmic m-ary tree, then the corresponding binary tree t' is $((m-1)h_0, \beta/(m-1))$-logarithmic.*

The h–v drawing for t' can be obtained by procedure **LT** modified according to Section 3.2. By removing all the 'dummy' nodes, we thus obtain an h–v–d^{m-2} drawing of t, where the definition of h–v–d^{m-2} drawing is the following.

Definition 13. An h–v–d^{m-2} drawing for an m-ary tree t is a grid drawing such that

(i) Edges are mapped into straight-line segment.
(ii) Edges from any node to its children are represented with one horizontal-rightward, one vertical-downward and at most $m-2$ 'diagonal' rightward-downward segments.
(iii) Enclosing rectangles of drawings for the immediate subtrees of a node are disjoint.
(iv) Edges do not intersect.
(v) For any subtree $t' = t'_1 \oplus \cdots \oplus t'_m$ of t with $m \geqslant 2$, the root of t'_i has abscissa not greater than that of t'_j, for any i and j with $i < j$. Moreover, (a) the abscissa (respectively, ordinate) of the root of t'_1 (respectively, t'_m) is equal to the abscissa (respectively, ordinate) of the root of t', and (b) the abscissa (respectively, ordinate) of the root of $t'_2, \ldots, t'_k$ (respectively, $t'_k, \ldots, t'_{m-1}$) is equal to the abscissa (respectively, ordinate) of the root of t' plus one, for some k in $\{2, \ldots, m-1\}$.

As explained in the case of ternary trees, the transformation of the h–v drawing of t' into the h–v–d^{m-2} drawing of t does not introduce edge-crossings.

Finally, it is easy to see that Lemma 10 holds also for h–v–d^{m-2} drawings. We thus have the following results.

Theorem 14. *Any (h_0, β)-logarithmic m-ary tree admits a linear-area strictly-upward drawing.*

Corollary 15. *For any fixed m, the family of m-ary B-trees admits linear-area strictly-upward drawings.*

4. Open questions

The main problem left open by this paper is that of reducing the value of the constant within the area function to draw binary trees. At the moment, this value is in the order of thousands which is clearly infeasible. However, both the results contained in [5] and experimental results show that this value should drastically decrease even though, clearly, it should be bigger as more unbalanced is the tree.

Another interesting question is to decide whether the minimum-area strictly-upward drawing is computable in polynomial time. This problem is NP-complete for general graphs [10] but it might be solvable in polynomial time for special classes of graphs such as search trees. Observe that in [8], an algorithm is given yielding a minimum

area h–v drawing of a binary tree with n nodes in time $O(n\sqrt{n \log n})$. Moreover, from our results this bound can be improved to $O(n^{3/2})$ in the case of logarithmic binary trees.

Appendix

Proof of Lemma 3. Let us first observe that, for any $h > h_0$,

$$k(h) = k_0 \prod_{i=h_0}^{h-1} \left(1 + \frac{2}{i^\alpha}\right).$$

Since $\ln(1+x) \leqslant x$ for all x, we have that

$$\ln \prod_{i=h_0}^{h-1} \left(1 + \frac{2}{i^\alpha}\right) = \sum_{i=h_0}^{h-1} \ln\left(1 + \frac{2}{i^\alpha}\right) \leqslant \sum_{i=h_0}^{h-1} \left(\frac{2}{i^\alpha}\right) < 2\zeta(\alpha),$$

where $\zeta(\cdot)$ denotes the Riemann zeta function defined as

$$\zeta(\alpha) = \sum_{i=1}^{\infty} \frac{1}{i^\alpha}.$$

Since, for any $\alpha > 1$, $\zeta(\alpha) < \infty$, the lemma thus follows. □

Proof of Lemma 4. Let us first suppose that $k(h)n_2/(l(h+1)-1) \geqslant (k(h+1)n - k(h)n_2)/l(h+1)$. Thus, from the definition of $k(h)$ and from the fact that $n = n_1 + n_2 + 1$, it follows that proving Lemma 4 is equivalent to proving that

$$k(h) + \frac{2k(h)n}{l(h)} \geqslant \frac{k(h)n_2}{l(h+1)-1},$$

that is

$$\frac{2n}{l(h)} \geqslant \frac{n_2}{l(h+1)-1} - 1$$

which is clearly true, since $l(h+1) \geqslant l(h)+1$ and $2n > n_2$.

Otherwise, that is, if $k(h)n_2/(l(h+1)-1) < (k(h+1)n - k(h)n_2)/l(h+1)$, we have to prove that

$$\left[k(h) + \frac{2k(h)n}{l(h)}\right] l(h+1) \geqslant k(h+1)n - k(h)n_2.$$

From the definition of $k(h)$, we have that

$$\begin{aligned} k(h+1)n - k(h)n_2 &= k(h)n + \frac{2k(h)n}{l(h)} - k(h)n_2 \\ &< k(h)n + \frac{2k(h)n}{l(h)} \end{aligned}$$

$$= \frac{2k(h)n}{l(h)}\left(1+\frac{l(h)}{2}\right)$$
$$< \frac{2k(h)n}{l(h)}l(h+1),$$

where the last inequality is due to the fact that $l(h+1) \geqslant l(h)+1$. The lemma thus follows. □

Proof of Lemma 5. Let us suppose that $h \geqslant 4$. Then, $k(h+1) = k(h)(1+2/l(h)) \leqslant \frac{3}{2}k(h)$. Since $n_2 < n \leqslant 2n_2 + 1 \leqslant 3n_2$, we have that

$$\frac{k(h+1)n(1-1/l(h)) - k(h)n_2}{\sqrt{k(h+1)n}} = \frac{k(h)n(1+2/l(h))(1-1/l(h)) - k(h)n_2}{\sqrt{k(h+1)n}}$$
$$> \frac{k(h)\left(\frac{l(h)-2}{l^2(h)}\right)n_2}{\sqrt{k(h)\frac{9}{2}n_2}} = \sqrt{\frac{2}{9}k(h)n_2}\frac{l(h)-2}{l^2(h)}$$
$$\geqslant \sqrt{\frac{2}{9}k(h)n_2}\frac{1}{2l(h)} > \sqrt{\frac{2}{9}n_2}\frac{1}{2l(h)},$$

where the second-last inequality is due to the fact that, for any $h \geqslant 4$, $l(h)-2 \geqslant l(h)/2$. Let $\bar{h}$ be the minimum integer greater than or equal to $\max(4, h_0)$ such that

$$\sqrt{\frac{2^{\beta h+1}}{9}} > 2h^{2\alpha}.$$

From the definition of an (h_0, β)-logarithmic class of trees and from the fact that $h > \bar{h}$, it follows that $n_2 \geqslant 2^{\beta h}$ which implies

$$\sqrt{\frac{2}{9}n_2} \geqslant \sqrt{\frac{2^{\beta h+1}}{9}} > 2h^{2\alpha} = 2l(h)^2.$$

In conclusion, we have that

$$\frac{k(h+1)n(1-1/l(h)) - k(h)n_2}{\sqrt{k(h+1)n}} > \sqrt{\frac{2}{9}n_2}\frac{1}{2l(h)} > \frac{2l(h)^2}{2l(h)} = l(h),$$

and the lemma is thus proved. □

Proof of Lemma 6. The proof is similar to that of Lemma 5. Indeed, since $k(h+1) = k(h)(1+2/l(h)) \leqslant 3k(h)$, we have that

$$\frac{k(h)n_2}{\sqrt{k(h+1)n}} \geqslant \frac{k(h)n_2}{\sqrt{3k(h)n}} > \frac{n_2}{\sqrt{3n}} \geqslant \sqrt{\frac{1}{9}n_2},$$

where the last inequality is due to the fact that $n \leqslant 2n_2+1 \leqslant 3n_2$. Let $\hat{h}$ be the minimum integer greater than or equal to h_0 such that

$$\sqrt{\frac{2^{\beta h}}{9}} > h^{\alpha}.$$

From the definition of an (h_0, β)-logarithmic class of trees and from the fact that $h > \hat{h}$, it follows that $n_2 \geqslant 2^{\beta h}$ which implies

$$\sqrt{\frac{1}{9}n_2} \geqslant \sqrt{\frac{2^{\beta h}}{9}} \geqslant h^{\alpha} = l(h).$$

In conclusion, we have that

$$\frac{k(h)n_2}{\sqrt{k(h+1)n}} > \sqrt{\frac{1}{9}n_2} > l(h),$$

and the lemma is thus proved. □

References

[1] T. Chan, M.T. Goodrich, S.R. Kosaraju, R. Tamassia, Optimizing area and aspect ratio in straight-line orthogonal tree drawings, in: Proc. Graph Drawing, 1996, pp. 63–75.

[2] C.-S. Shin, S.K. Kim, K.-Y. Chwa, Area-efficient algorithms for upward straight-line drawings, in: Proc. COCOON '96, to appear.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, McGraw-Hill, New York, 1990.

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

[5] P. Crescenzi, P. Penna, A. Piperno, Linear area upward drawings of AVL trees, Comput. Geometry: Theory Appl., to appear.

[6] P. Crescenzi, A. Piperno. Optimal-area upward drawings of AVL trees, in: Proc. Graph Drawing, 1994, pp. 307–317.

[7] G. Di Battista, P. Eades, R. Tamassia, I. Tollis, Algorithms for drawing graphs: an annotated bibliography, Comput. Geometry: Theory Appl. 4 (1994) 235–282.

[8] P. Eades, T. Lin, X. Lin. Minimum size h–v drawings, in: Proc. Internat. Workshop AVI '92, 1992, pp. 386–394.

[9] A. Garg, M.T. Goodrich, R. Tamassia, Area-efficient upward tree drawing, in: Proc. ACM Symp. on Computational Geometry, 1993, pp. 359–368.

[10] A. Garg, R. Tamassia, On the computational complexity of upward and rectilinear planarity testing, in: Proc. Graph Drawing, 1994, pp. 286–297.

[11] D.E. Knuth, The Art of Computer Programming: Sorting and Searching, Addison-Wesley, Reading, MA, 1975.

[12] Y. Shiloach, Linear and planar arrangements of graphs, Ph.D. Thesis, Department of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1976.

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