Optimal-area upward drawings of AVL trees
Extended abstract

P. Crescenzi* and A. Piperno

Dipartimento di Scienze dell'Informazione
Università degli Studi di Roma "La Sapienza"
Via Salaria 113, 00198 Roma
E-mail: {crescenzi,piperno}@dsi.uniroma1.it

Abstract. We prove that any AVL tree admits a linear-area planar straight-line grid strictly-upward drawing, that is, a drawing in which (a) no two edges intersect, (b) each edge is mapped into a single straight-line segment, (c) each node is mapped into a point with integer coordinates, and (d) each node is placed below its parent.

1 Introduction

In this paper we are interested in planar straight-line grid strictly-upward drawings (in short upward drawings) of binary rooted trees, that is, drawings in which (a) no two edges intersect, (b) each edge is mapped into a single straight-line segment, (c) each node is mapped into a point with integer coordinates, and (d) each node is placed below its parent.[1] A natural and important criterion for evaluating these drawings is that they take as little area as possible. Most of the known algorithms to upward draw a binary tree require quadratic area in the worst case [10, 11]. Recently, Crescenzi, Di Battista, and Piperno proved that $\Omega(n \log n)$-area is required in order to upward draw any binary tree and described an algorithm to obtain such a drawing. Moreover, they gave two algorithms producing a linear-area upward drawing of complete and Fibonacci binary trees, respectively [2]. Successively, Garg, Goodrich, and Tamassia 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 [8].

This latter result, however, doesn't settle the question whether a larger class of binary trees exists which can be upward drawn in linear area. Since the Fibonacci and complete binary trees are the (unlabeled) AVL trees with the least and the most number of nodes, respectively, it seems natural to ask whether the result of Crescenzi, Di Battista, and Piperno can be generalized to any AVL tree.

In this paper we positively answer this question, that is, we prove that, for any AVL tree t with n nodes, an upward drawing of t can be produced with area $O(n)$ in time $O(n)$. In particular, our main result shows that any AVL tree with n nodes can be upward drawn in any rectangle whose shortest side is at least $\log^{\alpha} n$ and whose area is equal to κn where α and κ are two constants. This result improves that obtained in [8] (when applied to AVL trees) in two directions. On the one hand, our algorithm produces straight-line strictly-upward drawings, on the other the bound on the length of the shortest side provides a greater flexibility to applications that need to fit the drawing in a prescribed rectangular region.

* Research partially supported by the MURST project *Algoritmi, Modelli di Calcolo, Strutture Informative.*

[1] Upward drawing is just one of several graphic standards that have been proposed for the drawing of planar graphs. The annotated bibliography mantained by Di Battista, Eades, and Tamassia mentions many papers in this research area [3].

We also present some experimental results which illustrate how, in practice, the area requirements are much less than those specified by the theoretical results. These experiments show that even from a practical point of view our algorithm improves the one proposed in [8].

1.1 Preliminaries

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

We refer to directed rooted unordered binary trees, in short binary trees. In particular, c_h denotes the *complete binary tree* of height h while F_h denotes the *Fibonacci tree* of height h [5, 6, 7]. Moreover, $n_c(h)$ and $n_F(h)$ denote the number of nodes of c_h and F_h, respectively. A binary tree is said to be *k-balanced* if, for each node u, the heights of the two subtrees of u differ of at most k. A 1-balanced binary tree is also called *AVL* [1] and it is well known that, for any AVL tree t of height h, $n_F(h) \leq n \leq n_c(h)$ where n denotes the number of nodes of t.

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

1. Edges are straight-line segments that do not intersect.
2. Nodes are points with integer coordinates.
3. A node has the ordinate greater than that of its parent —we are thus assuming that the y-axis is downward oriented.

The width (respectively, height) of a drawing is the length of 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 binary tree has both width and height greater than zero. The *area* of an upward drawing is then defined as the product of the width and the height.

Crescenzi, Di Battista, and Piperno proved the following result [2].

Theorem 1. *An infinite class of binary trees exists requiring $\Omega(n \log n)$ area to be upward drawn.*

A question naturally arises as a consequence of the above theorem: do classes of binary trees exist which can be upward drawn in linear area? A preliminary answer to this question has been given by Crescenzi, Di Battista, and Piperno [2].

Theorem 2. *An algorithm exists producing an upward drawing of either a complete binary tree or a Fibonacci binary tree with n nodes in $O(n)$ area.*

In order to prove the above theorem, Crescenzi, Di Battista, and Piperno introduce the notion of *h-v drawing*. An h-v drawing is an upward drawing in which rightward-horizontal and downward-vertical straight-line segments only are allowed. That is, the notion of h-v drawing is a restriction of that of *orthogonal drawing* in which each edge is a chain of alternating horizontal and vertical segments [8]: on the one hand, each edge can be only one segment, on the other no leftward-horizontal segments are allowed.

More precisely, an h-v drawing of a binary tree t is obtained by one of the two operations illustrated in Fig. 1 where D_1 and D_2 are two h-v drawings of the two subtrees of t. In the first operation, D_2 is traslated to the right by as many grid points as the width of D_1 and D_1 is traslated to the bottom by one grid point. The semantic of the second operation is defined similarly.

The following fact shows that h-v drawings are a powerful tool to deal with upward drawings.

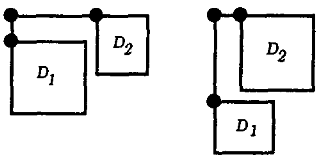


Fig. 1. The two operations of an h-v drawing

Proposition 1. *Any h-v drawing of area A can be transformed into an upward drawing of area at most* $2A$.

The above result will allow us to devote our attention to h-v drawings only (note that we are interested in upper bounds on the area requirements). In [4], finally, an algorithm is given yielding a minimum area h-v drawing of a binary tree with n nodes in time $O(n\sqrt{n \log n})$.

2 The Algorithm

In this section we shall show how to obtain linear-area upward drawings of AVL trees.

Let us first reconsider the approach of [2]. This approach mainly uses the fact that complete and Fibonacci binary trees are inductively defined and shows that it is possible to draw one of these trees of height $h+1$ by using the drawings produced for the corresponding trees of height h.

Any attempt to extend such a bottom-up construction to the case of AVL trees leads to the necessity of producing multiple drawings for any tree. Indeed, let us fix an AVL tree t of height h along with a drawing of t. An AVL tree of height $h+1$ having t as one of its subtrees can be obtained in many different ways depending on the choice of the other subtree. If the drawing of this subtree must be produced so that, when combined with the fixed drawing of t, the linear-area requirement is mantained, then it is immediate to realize that the same drawing cannot be used in combination with the drawings of all trees of height h, which could be considerably different from t (for example, in the number of nodes).

To overcome this difficulty, we shall instead follow a top-down approach. From the definition of the h-v drawing operations, any rectangle including a drawing for an AVL tree t of height h must contain two rectangles including the drawings of the immediate subtrees of t. From an algorithmic point of view, given a rectangle R in which we want to draw t, we must be able to cut it into two rectangles in which it is possible to draw the immediate subtrees of t. Our cut-rule can be roughly described as follows: *cut R proportionally to the number of nodes of the two subtrees.* Two problems arise from the previous rule. On the one hand, we need to maintain the linear-area requirement, on the other we must be able to treat 'highly rectangular' shapes (for example, consider the case in which one subtree is a Fibonacci tree of height $h-2$ while the other is a complete tree of height $h-1$).

This section is devoted to the study of the conditions under which this construction can be safely carried out. Given an AVL tree, we shall denote with n the number of its nodes and with l and L the length of the shortest and the longest side of the rectangle in which the tree has to be drawn, respectively. Intuitively, we shall prove that, if l and lL are 'large enough', then there is a way of cutting the rectangle which preserves the desired properties on the sides of the two obtained subrectangles.

Our proofs will refer to rectangles with real coordinates. However, it is clear that if we can draw a tree whithin a real-coordinate rectangle R by mapping nodes into points with integer coordinates, then the tree itself can be drawn whithin the largest integer-coordinate rectangle included in R.

```
algorithm $BT(R = \langle l, L, (x,y), b\rangle, t)$;
begin
    cut $L$ into two segments of length $l_1$ and $l_2$, respectively;
    {$l_1$ and $l_2$ will be specified in the proof of Theorem 3}
    if $b$ then $(x_1,y_1) := (x+1,y)$ else $(x_1,y_1) := (x,y+1)$;
    if $l_1 > l-1$ then $R_1 := \langle l-1, l_1, (x_1,y_1), b\rangle$ else $R_1 := \langle l_1, l-1, (x_1,y_1), \textbf{not } b\rangle$;
    if $b$ then $(x_2,y_2) := (x, y+\lfloor l_1 \rfloor)$ else $(x_2,y_2) := (x+\lfloor l_1 \rfloor, y)$;
    if $l_2 > l$ then $R_2 := \langle l, l_2, (x_2,y_2), b\rangle$ else $R_2 := \langle l_2, l, (x_2,y_2), \textbf{not } b\rangle$;
    map the root of $t$ into $(x,y)$;
    $t_1$ := subtree of $t$ with the least number of nodes;
    $t_2$ := subtree of $t$ with the most number of nodes;
    if $t_1$ is not empty then $BT(R_1,t_1)$;
    if $t_2$ is not empty then $BT(R_2,t_2)$;
end.
```

Fig. 2. The algorithm to draw an AVL tree

In the following, a rectangle is specified by the lengths l and L of its sides, by the coordinates x and y of its leftmost-topmost corner which is always assumed to be a point with integer coordinates, and by a Boolean flag b which indicates the orientation of the longest side (if b is true then the longest side is vertical, otherwise it is horizontal). Our algorithm is then shown in Fig. 2 (see also Fig. 3).

In order to precisely specify the above algorithm, let us first introduce some functions together with properties they are required to satisfy.

1. The *factor* function

$$k : \mathbf{N}^+ \rightarrow \mathbf{R}^+,$$

that is, the function specifying the constant factor in the area bound, is supposed to be a non decreasing function which satisfies the following

Property I. A constant κ exists such that, for any h, $k(h) \leq \kappa$.

Since we will require that any AVL tree of height h with n nodes can be drawn in area $k(h)n$, the linear-area bound immediately follows from Property I.

2. The *shortest side* function

$$l : \mathbf{N}^+ \rightarrow \mathbf{R}^+,$$

that is, the function specifying the lower bound for l, is supposed to satisfy the following

Property II. $l(1) \geq 1$ and, for any h, $l(h+1) \geq l(h) + 1$.

This property implies that the length of the shortest side of the drawing of a tree of height h is always greater than h. Such a requirement is due to the nature of the h-v drawing operations.

3. The *area* function

$$A : \mathbf{N}^+ \times \mathbf{N}^+ \rightarrow \mathbf{R}^+,$$

that is, the function specifying the area of the drawing, is defined as

$$A(h,n) = k(h)n.$$

and is supposed to satisfy the following

Property III. For any $h \geq 2$ and for any n_1, n_2 such that $n_F(h-1) \leq n_1 \leq n_2 \leq n_c(h)$,

$$A(h, n_1) \geq \sqrt{A(h+1, n)}\; l(h),$$

where $n = n_1 + n_2 + 1$.

Observe that if $l \geq l(h)$, then the length L of the longest side is at most

$$L(h, n) = \frac{A(h, n)}{l(h)}.$$

Clearly, if an AVL tree of height h with n nodes exists, then $L(h, n)$ should be at least equal to $l(h)$. This is guaranteed, for $h = 1$, taking $k(1)$ sufficiently large and, for $h > 1$, by the following proposition.

Proposition 2. *For any $h \geq 2$ and for any n such that $n_F(h) \leq n \leq n_c(h)$,*

$$L(h, n) > l(h).$$

Proof. By using Property III with $n_1 = n_2$ and the assumption for k to be non decreasing, we have

$$\frac{L(h,n)}{l(h)} = \frac{A(h,n)}{l^2(h)} \geq A(h,n)\frac{A(h+1, 2n+1)}{A^2(h,n)} = \frac{A(h+1, 2n+1)}{A(h,n)} > 1,$$

that is, $L(h, n) > l(h)$. □

Finally, the area function is supposed to satisfy the following

Property IV. For any h and for any n_1, n_2 with $n_1 \leq n_2$,

$$A(h+1, n) \geq A(h, n_1) + A(h, n_2) + \frac{A(h+1, n) - A(h, n_2)}{l(h+1)}$$

where $n = n_1 + n_2 + 1$.

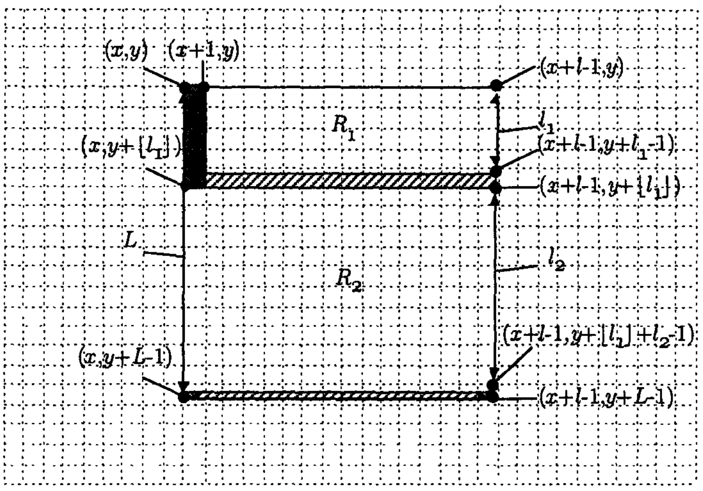
Properties III and IV express the conditions that a rectangle has to satisfy in order to allow its cutting into two suitably large subrectangles.

We are now in a position to prove the main result of this section. Intuitively, this result shows that any AVL tree of height h with n nodes can be upward drawn in any rectangle whose shortest side and whose area are large enough.

Theorem 3. *Let k, l, and A be three functions satisfying Properties II-IV and let $h, n \in \mathbb{N}^+$ and R be any rectangle whose sides have sizes l and L, respectively, satisfying the following two conditions:*

$$L \geq l \geq l(h) \quad \textit{and} \quad lL = A(h, n).$$

Then any AVL tree of height h with n nodes admits an upward drawing whithin the rectangle R.



(x,y)
$(x+1,y)$
$(x+l-1,y)$
R_1
l_1
$(x,y+\lfloor l_1 \rfloor)$
$(x+l-1,y+l_1-1)$
$(x+l-1,y+\lfloor l_1 \rfloor)$
L
l_2
R_2
$(x+l-1,y+\lfloor l_1 \rfloor+l_2-1)$
$(x,y+L-1)$
$(x+l-1,y+L-1)$

Fig. 3. The splitting of a rectangle

Proof. The proof is by induction on h. For $h = 1$, the proof is straightforward.

Let $h \geq 1$ and let us assume that the theorem is true for any height less than $h+1$. Given an AVL tree of height $h+1$ with n nodes where $n = n_1 + n_2 + 1$ and $n_F(h-1) \leq n_1 \leq n_2 \leq n_c(h)$, let us define

$$l_2 = \frac{A(h,n_2)}{l} \quad \text{and} \quad l_1 = L - l_2.$$

Intuitively, we are isolating two rectangles R_1 and R_2 within the rectangle R. The sides of R_1 have length l_1 and $l-1$, respectively, while the sides of R_2 have length l_2 and l, respectively. Fig. 3 illustrates the case in which the longest side of R is the vertical one. Note that according to the algorithm of Fig. 2, the topmost-leftmost corners of R_1 and R_2 have coordinates $(x+1,y)$ and $(x, y+\lfloor l_1 \rfloor)$, respectively, where x and y denote the coordinates of the topmost-leftmost corner of R. Since the length of a segment is measured by the number of grid points, we thus have that the y-coordinate of the rightmost-bottommost corner of R_2 is equal to $y + \lfloor l_1 \rfloor + l_2 - 1$ which is less than or equal to the y-coordinate of the rightmost-bottommost corner of R, that is, $y + L - 1$.

Clearly, the area of R_2 is equal to $A(h,n_2)$. Moreover, since

$$L - l_2 = \frac{lL - A(h,n_2)}{l} = \frac{A(h+1,n) - A(h,n_2)}{l} \leq \frac{A(h+1,n) - A(h,n_2)}{l(h+1)},$$

Property IV guarantees that the area of R_1 is at least equal to $A(h,n_1)$.

Let h_1 and h_2 denote the heights of the two subtrees with n_1 and n_2 nodes, respectively. We now shall prove that the shortest sides of R_1 and R_2 have length at least $l(h_1)$ and $l(h_2)$, respectively. To this aim, we distinguish the following three cases.

1. $h_1 = h-1$ and $h_2 = h$.
   (a) Rectangle R_1. If $l_1 \geq l-1$, then from Property II it follows that

$$l - 1 \geq l(h+1) - 1 \geq l(h) \geq l(h-1).$$

   Otherwise,

$$l_1 \geq \frac{A(h,n_1)}{l-1} \geq l(h) \geq l(h-1),$$

where the second inequality follows from the fact that $l \leq \sqrt{A(h+1,n)}$ and from Property III.

(b) Rectangle R_2. If $l_2 \geq l$, then from Property II it follows that

$$l \geq l(h+1) \geq l(h).$$

Otherwise, since $A(h,n_2) \geq A(h,n_1)$, we have that

$$l_2 = \frac{A(h,n_2)}{l} \geq \frac{A(h,n_1)}{l} \geq l(h),$$

where the last inequality follows from the fact that $l \leq \sqrt{A(h+1,n)}$ and from Property III.

2. $h_1 = h$ and $h_2 = h-1$. The proof is similar to that of case 1: note that in this case we are simply decreasing the upper bound for n_2 and increasing the lower bound for n_1.
3. $h_1 = h_2 = h$. The proof is similar to that of case 1: note that in this case we are simply increasing the lower bound for n_1.

In all three cases we have that the inductive hypothesis is satisfied for both a rectangle contained in R_1 and a rectangle contained in R_2. The theorem thus follows. □

Corollary 1. *If the function k in the previous theorem satisfies Property I, then any AVL tree admits a linear-area upward drawing.*

3 The proof of existence

In this section we shall prove the existence of the functions $k(h)$ and $l(h)$ satisfying the properties of the previous section. The definition of these two functions is quite simple: indeed, $l(h) = h^\alpha$ and $k(h+1) = (1+i(h))k(h)$ where $\alpha > 1$ and $i(h)$ will be specified later. These definitions are motivated by two natural reasons: on the one hand, the shortest side cannot be smaller than the height of the tree, on the other the higher is the tree the bigger should be the constant factor (even though asymptotically bounded by a constant).

Since $\alpha > 1$, we have that $(h+1)^\alpha \geq h^\alpha + 1$, that is, the function $l(h)$ satisfies Property II.

Observe now that from the definition of $A(h,n)$ it follows that proving Property IV is equivalent to proving that

$$[i(h)A(h,n) + k(h)]l(h+1) \geq A(h+1,n) - A(h,n_2).$$

Since $n > 2n_1$, we have that

$$\begin{aligned} A(h+1,n) - A(h,n_2) &= A(h,n) + i(h)A(h,n) - A(h,n_2) \\ &= A(h,n_1) + k(h) + i(h)A(h,n) \\ &< A(h,n)/2 + i(h)A(h,n) + k(h) \\ &= i(h)A(h,n)[1 + 1/2i(h)] + k(h) \\ &\leq [i(h)A(h,n) + k(h)][1 + 1/2i(h)]. \end{aligned}$$

We then need to define $i(h)$ so that $1 + 1/2i(h) \leq l(h+1)$. From Property II it follows that we can simply define $i(h) = 1/2l(h)$.

In order to prove Property III, we have to show that, for any $h \geq 2$,

$$k^2(h)n_1^2 \geq A(h+1,n)h^{2\alpha} = k(h)n\left(\frac{2h^\alpha+1}{2h^\alpha}\right)h^{2\alpha}.$$

This is clearly true if

$$k(h)\frac{n_1^2}{n} \geq \frac{5}{4}h^{2\alpha}.$$

Since n_1^2/n is minimum when n_1 is minimum and n_2 is maximum, we thus have to prove that

$$k(h)\frac{1}{\rho(h)} \geq \frac{5}{4}h^{2\alpha}$$

where, for any $h > 1$,

$$\rho(h) = \frac{n_F(h-1)+n_c(h)+1}{n_F^2(h-1)}.$$

We shall prove the above inequality by induction on h and by suitably choosing α and $k(1)$. For $h = 2$, if we set $k(1)$ equal to $\frac{25}{6} \cdot 2^{2\alpha}$, then

$$k(2)\frac{1}{\rho(2)} = \frac{3}{2}k(1)\frac{1}{5} = \frac{5}{4}2^{2\alpha}.$$

Let us assume that the inequality is satisfied for any height less than $h+1$. Then

$$\frac{k(h+1)}{\rho(h+1)} = \left(1+\frac{1}{2h^\alpha}\right)\frac{k(h)}{\rho(h+1)} \geq \left(1+\frac{1}{2h^\alpha}\right)\frac{5}{4}h^{2\alpha}\frac{\rho(h)}{\rho(h+1)}.$$

In order to satisfy the inequality for $h+1$, we must then define α so that

$$\left(1+\frac{1}{2h^\alpha}\right)\frac{5}{4}h^{2\alpha}\frac{\rho(h)}{\rho(h+1)} \geq \frac{5}{4}(h+1)^{2\alpha}.$$

It is easy to see that α can be any value between 1 and $\hat{\alpha}$ where $\hat{\alpha} \approx 1.111$ is the solution of the following equation:

$$(2 \cdot 2^x + 1)2^x = 3^{2x}.$$

Table 1 shows the first 10 values of $k(h)$, and, successively, its values for $h = 20, 40, \ldots, 120$ in the case $\alpha = 1.005$. As shown in the table, the value of $k(h)$ increases slower and slower: this is formally justified by the following fact (whose simple proof is here omitted).

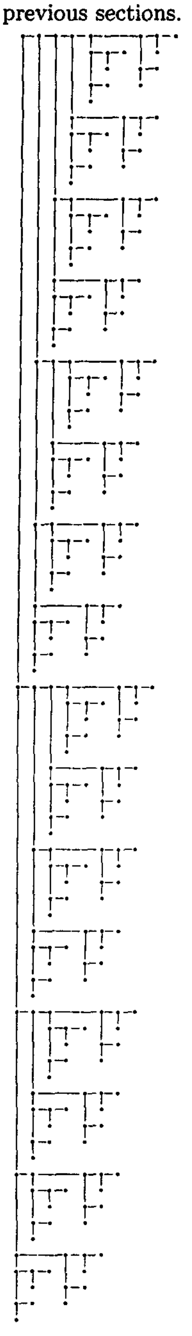
Proposition 3. *The function $k(h)$ satisfies Property I. In particular, for any h,*

$$k(h) \leq \frac{25}{6} \cdot 2^{2\alpha}\sqrt{2^{\zeta(\alpha)}}$$

where ζ denotes the Riemann zeta function [9].

4 Practical considerations, extensions, and open questions

In this section we shall discuss practical consequences and extensions of the results of the previous sections.



First of all, the theoretical upper bound on the area required to upward draw an AVL tree can be, in practice, substantially improved. The idea is to use the algorithm described in Fig. 2 in order to compute, for each node, the *h-v operation* to be performed at that node. More formally, this can be done by simply replacing in the algorithm the instruction

map the root of t into (x, y)

by the instruction

label the root of t with b

(if a node is labelled with a true value, then its operation is the second operation in Fig. 1, otherwise its operation is the first one). Once this preprocessing has been realized, the drawing of the tree is obtained by simply performing, for each node, the corresponding operation. Intuitively, this modification yields the smallest drawing which is obtained by the same sequence of h-v operations generated by the 'theoretical' algorithm. We have implemented such an algorithm. Table 2 presents experimental results obtained for complete binary trees, Fibonacci trees, and combinations of these two kind of trees. Note that from these experiments it can be conjectured that the real constant factor in the area bound is approximately 3. Unfortunately, we have not been able to prove this conjecture and leave it as an open problem.

Secondly, observe that the bound on the length of the shortest side allows a very great flexibility to applications that need to draw a binary tree in a prespecified rectangular region. For instance, at the left of this paragraph, we have been able to draw a complete binary tree of height 8 without reducing too much the width available to the text. The area of this drawing is 800 which is a little bit more than three times the number of nodes. It is still an open question whether a logarithmic bound on the length of the shortest side is also obtainable (recall that $\alpha > 1$).

Finally, the algorithm we presented can be easily extended to the class of 2-balanced binary trees. In this case, for any node u of the tree, the number n of nodes in the subtree rooted at u satisfies the following inequality:

$$n_2(h) \leq n \leq n_c(h)$$

where h denotes the height of the tree rooted at u, and n_2 is defined as

$$n_2(h) = \begin{cases} h & \text{if } h \leq 2, \\ n_2(h-1) + n_2(h-3) & \text{otherwise.} \end{cases}$$

The previous two sections can then be modified in order to prove that any 2-balanced binary tree admits a linear-area upward drawing. However, it is easy to see that in this case the constant factor in the area bound substantially increases.

Even though one might think of a different algorithm, the behaviour of our procedure seems to be sufficiently natural: as more unbalanced is the tree, as bigger must be the area of the drawing. This is in accordance with the lower bound of [2] which refers to a family of binary trees which are highly unbalanced.

In any case, it still remains open the question whether other types of balanced trees, such as k-balanced tress with $k > 2$, red-black trees, and weight-balanced trees, admit linear-area upward drawings. More generally, it would be interesting to know whether our results can be extended to any family of trees with logarithmic height.

References

1. G.M. Adelson-Velskii and E.M. Landis. An algorithm for the organization of information. *Soviet Math. Dokl.*, 3:1259–1262, 1962.
2. P. Crescenzi, G. Di Battista, and A. Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Computational Geometry: Theory and Applications*, 2:187–200, 1992.
3. G. Di Battista, P. Eades, and R. Tamassia. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*, to appear. A preliminary version is available via anonymous ftp from wilma.cs.brown.edu, gdbiblio.tex.Z and gdbiblio.ps.Z in /pub/papers/compgeo.
4. P. Eades, T. Lin, and X. Lin. Minimum size h-v drawings. In *Proc. Int. Workshop AVI '92*, pages 386–394, 1992.
5. Y. Horibe. An Entropy View of Fibonacci Trees. *Fibonacci Quarterly*, 20:168-178, 1982.
6. Y. Horibe. Notes on Fibonacci Trees and Their Optimality. *Fibonacci Quarterly*, 21:118-128, 1983.
7. D.E. Knuth. *The Art of Computer Programming*, Addison Wesley, 1975.
8. A. Garg, M.T. Goodrich, and R. Tamassia. Area-efficient upward tree drawing. In *Proc. ACM Symp. on Computational Geometry*, pages 359–368, 1993.
9. R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics*, Addison Wesley, 1989.
10. E. Reingold and J. Tilford. Tidier drawing of trees. *IEEE Trans. on Software Engineering*, SE-7:223–228, 1981.
11. K.J. Supowit and E. Reingold. The complexity of drawing trees nicely. *Acta Informatica*, 18:377–392, 1983.

Table 1. The first values of $k(h)$ in the case $\alpha = 1.005$

h	$k(h)$
1	4.166667
2	6.250000
3	7.807094
4	9.101149
5	10.230934
6	11.245828
7	12.174622
8	13.035817
9	13.842129
10	14.602734
20	20.687508
40	29.180922
60	35.637283
80	41.048038
100	45.793926
120	50.069237

Table 2. Experimental results on the drawings produced by the modification of the algorithm of Fig. 2 starting with $l = L = \sqrt{A(h,n)}$ for comparison with the theoretical upper bounds

Type	Nodes n	Width	Height	Area A	Ratio A/n
Complete binary tree	15	6	5	30	2.000
	63	12	12	144	2.286
	255	24	28	672	2.635
	1023	48	60	2880	2.815
	4095	96	119	11424	2.790
	16383	192	239	45888	2.801
	32767	267	349	93183	2.844
Fibonacci tree	20	6	5	30	1.5
	88	12	13	156	1.773
	232	20	22	440	1.897
	609	34	37	1258	2.066
	2583	70	74	5180	2.005
	4180	94	100	9400	2.249
Complete and Fibonacci tree	20	6	7	42	2.100
	76	16	12	192	2.526
	289	28	29	812	2.810
	1112	53	61	3233	2.907
	4328	104	125	13000	3.004
	16993	201	253	50853	2.993
	67132	396	509	201564	3.003