# A Note on Minimum-Area Upward Drawing of Complete and Fibonacci Trees

Luca Trevisan

*Dipartimento di Scienze dell'Informazione*
*Via Salaria 113, 00198 Roma, Italy*
*E-mail:* trevisan@dsi.uniroma1.it

**Abstract**

We study the area requirement for upward straight-line grid drawing of complete and Fibonacci tree. We prove that a complete tree with $n$ nodes can be drawn in $n + O(\log n\sqrt{n})$ area, and a Fibonacci tree with $n$ nodes can be drawn in $1.17n + O(\log n\sqrt{n})$ area.

*Keywords:* computational geometry, graph drawing.

## 1 Introduction

In this paper we consider planar straight-line upward drawings (in short, upward drawings) of rooted trees, that is, drawings such that *no two edges intersect*, each edge is drawn as a *straight-line segment*, each node is drawn on a point of an *integer-coordinate grid*, and is placed *below* its parent. Many references about upward drawing of arbitrary graphs can be found in the annotated bibliography maintained by Di Battista, Eades and Tamassia [1].

Upward drawings have applications in program animation and in data structure visualization, and, more generally, are a convenient representation of *hierarchical* structures. Since such drawings have to be presented on screens or other bounded resolution media, it is a natural requirement that they should take as little area as possible.

Crescenzi, Di Battista and Piperno showed that every binary tree admits an upward drawing into a grid of area $O(n \log n)$, where $n$ is the number of nodes, and that this bound is optimal within a constant factor because an infinite family of binary trees exists that require area $\Omega(n \log n)$ [2]. Moreover, they also show that both complete and Fibonacci trees can be drawn in linear area. In order to achieve this latter result, they introduce the notion of *h-v drawing*.

They develop an algorithm that finds linear-area h-v drawings for complete and Fibonacci trees, and then show that any h-v drawing of area $A$ can be easily transformed into an upward drawing of area $2A$.

Successively, Crescenzi and Piperno presented an algorithm that finds a linear area upward drawing for any AVL tree, again using the notion of h-v drawing [3]. It is also interesting to note that *minimum area* h-v drawings can be found for any binary tree in polynomial time [4].

In a related paper, Garg, Goodrich, and Tamassia proved that *any* binary tree can be drawn in linear area, provided that we slightly relax both the upward and the straight-line conditions [6].

In this paper we present a new algorithm that draws a complete binary tree with $n$ nodes into a grid of area $n + O(\log n\sqrt{n})$, so that, asymptotically, all the points of the grid hold a node, but a negligible ratio. Of course, the drawings produced by such an algorithm are not very readable, and they thus contrast with the recognition of minimum area upward drawings as "nice" drawings. Our algorithm can also draw $m$-ary complete trees with the same area requirement, and this is the first linear-area upward drawing algorithm that extends beyond the class of binary trees. Moreover, a straightforward application of the same idea yields an algorithm that draw a Fibonacci tree with $n$ nodes into a grid of area $1.171n + O(\log n\sqrt{n})$, thus improving over the area requirement of [2] (that is about $6n$).

### *1.1 Preliminaries*

We denote by $e$ the empty tree and by $\mathbf{1}$ the tree consisting of a single node. Given two binary trees $T_1$ and $T_2$, we denote by $T_1 \oplus T_2$ the binary tree such that $T_1$ (respectively, $T_2$), is its immediate left (respectively, right) subtree. For any $h \geq 0$, $c_h$ denotes the *complete binary tree* of height $h$, and $F_h$ denotes the *Fibonacci tree* of height $h$ [8]. We recall that the inductive definition of $F_h$ is the following: $F_0 = \mathbf{1}$, $F_1 = e \oplus \mathbf{1}$, $F_i = F_{i-2} \oplus F_{i-1}$ for $i > 1$. Let $T = T_1 \oplus T_2$ be a non-empty tree, we define $left(T) = T_1$ and $right(T) = T_2$.

Let $n_F(h)$ be the number of nodes of $F_h$: it is easy to prove by induction that $n_F(h) = f_{i+3} - 1$ where $f_i$ denotes the $i$th Fibonacci number.

A *planar straight-line strictly upward grid drawing* (in short, an *upward drawing*) of a tree $T$ is a drawing of $T$ such that:

(i) edges are straight-line segments that do not intersect;
(ii) nodes are points with positive integer coordinates;
(iii) a node has the ordinate greater than that of its parent.

We assume that the topmost leftmost point of the grid has coordinates $(1,1)$, and that the first and the second coordinate increase rightwards and downwards, respectively.

The *width* (respectively, the height) of a drawing is the maximum value of the first (respectively, the second) coordinate of the points of the drawing. The *area* of a drawing is the product of its width times its height.

## 2 Complete Trees

In this section we will describe an algorithm that upward draws an $n$-node complete binary tree in a grid of area $n + O(\log n\sqrt{n})$.

The basic idea is to divide the drawing into two parts: given a complete binary tree of height $h$, we first draw in a trivial way the levels between 0 and $\lfloor (h+1)/2 \rfloor$ (using a rectangle of area $(\lfloor (h+1)/2 \rfloor + 1) \times 2^{\lfloor (h+1)/2 \rfloor}$). Successively, we have to draw $2^{\lfloor (h+1)/2 \rfloor}$ subtrees of height $h - \lfloor (h+1)/2 \rfloor = \lfloor h/2 \rfloor$ (indeed the roots of these subtrees have been already drawn). These trees are drawn in the following way. Let $T_u$ be one of these trees, let $u$ be its root, and let $(x_u, y_u)$ be the coordinates of the point where $u$ has been drawn during the first phase; for any $l = 1, \ldots, \lfloor h/2 \rfloor$, the nodes of level $l$ are drawn at horizontal position $x_u + l$ and at a vertical position between $\lfloor (h+1)/2 \rfloor + 2^l$ and $\lfloor (h+1)/2 \rfloor + 2^{l+1} - 1$.

In Figure 1 a detailed description of this algorithm is presented.

**Theorem 1** *For every complete tree $T = c_h$ with $n = 2^{h+1} - 1$ nodes, algorithm* `CompDraw` *finds a planar strictly upward drawing for $T$ into a grid of area $n + O(\log n\sqrt{n})$.*

**Proof.** It is easy to see that the algorithm places each node in a different point. Moreover, the drawing produced by this algorithm is strictly upward, because in the upper part of the drawing each son is exactly one row below the father, while, in the lower part, all the nodes at level $i$ are drawn strictly over all the nodes at level $i+1$. Finally the drawing is planar. The planarity of the upper part is trivial; in the lower part, every edge is a segment connecting a certain point $(x, y)$ to a point $(x+1, y+k)$ with $k > 0$; for any other segment $\overline{(x, y')(x+1, y'+k')}$ either $y' < y$ and $y' + k' < y + k$ or $y < y'$ and $y + k < y' + k'$: thus no intersection is possible.

Let us now consider the area requirement: the complete binary tree of height $h$ is drawn in a rectangle whose height is $\lfloor (h+1)/2 \rfloor + 2^{\lfloor h/2 \rfloor + 1} - 1$ and whose width is $2^{\lfloor (h+1)/2 \rfloor} + \lfloor h/2 \rfloor$. The area is thus $A_c(h) \leq 2^{h+1} + h2^{\lfloor h/2 \rfloor + 1} + \lfloor h/2 \rfloor^2$. If we express the area as a function of the numbers of nodes $n = 2^{h+1} - 1$, we

```
algorithm CompDraw(T);
begin
  h = height(T); h' = ⌊(h + 1)/2⌋;
  for l = 0 to h' do
    for x = 1 to 2^l do begin
      let u be the x-th node at level l in T;
      draw(u, x, l + 1);
    end;
  for x = 1 to 2^h' do begin
    let u be the x-th node at level h' in T;
    let T_u be the subtree of T rooted in u;
    for l = 1 to h − h' do
      for j = 1 to 2^l do begin
        let v be the j-th node at level l in T_u;
        draw(v, x + l, h' + 2^l − 1 + j);
      end
  end
end.
```

Fig. 1. The algorithm for drawing complete trees.

have that $A_c(h) = n + O(\log n\sqrt{n})$. □

In Figure 2 we show the drawing produced by algorithm `CompDraw` for the complete binary tree of height $h = 5$ with 63 nodes. The upper part of the drawing is the complete tree with 8 leaves (that is, $2^{\lfloor(5+1)/2\rfloor}$); in each of these leaves a complete tree with 7 nodes is rooted. These 8 trees are drawn in the lower part. The area of the drawing is

$$A_c(h) = (\lfloor(h+1)/2\rfloor + 2^{\lfloor h/2\rfloor+1} - 1) \times (2^{\lfloor(h+1)/2\rfloor} + \lfloor h/2\rfloor) = 10 \times 10.$$

Algorithm `CompDraw` can be straightforwardly extended to $m$-ary complete trees by simply replacing $2^l$ (respectively, $2^{h'}$) with $m^l$ (respectively, $m^{h'}$). The area requirement is still $n + O(\sqrt{n}\log n)$.

## 3 Fibonacci Trees

Let us now consider a possible implementation of the previous algorithm in the case of Fibonacci trees. Essentially, the algorithm described in the previous section first performs a breadth-first-search of the tree, stopping each time a node is found with a subtree with about $\sqrt{n}$ nodes; the subtree $T_1$ obtained by this first phase is drawn in a trivial way, then the $\simeq \sqrt{n}$ trees rooted in the leaves of $T_1$ are drawn "levelwise". We take advantage of the structure of

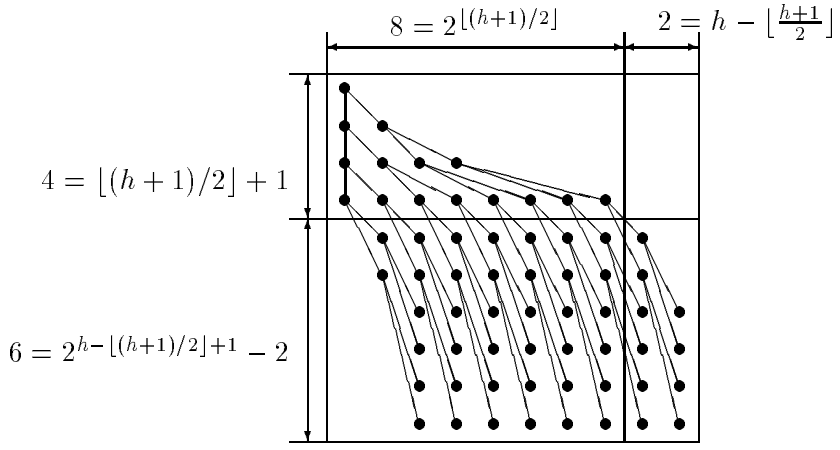


Fig. 2. How to draw the complete tree with 63 nodes in a grid of area 100.

complete trees, since we know that exactly $2^{\lfloor (h+1)/2 \rfloor}$ nodes exists in $T$ such that the size of their subtrees is $2^{h-\lfloor (h+1)/2 \rfloor + 1} - 1$. Moreover, all these nodes are at the same level $\lfloor (h+1)/2 \rfloor$ in $T$. Some more work is needed to achieve a similar construction on a Fibonacci tree, and, as we will see, a little efficiency is lost. First of all, let us define two functions that will be useful in the sequel.

**Definition 2** *For any $h \geq h' > 0$, let $T_F(h, h')$ be the smallest non-empty subtree of $F_h$ such that if $u$ is the root of a subtree of $F_h$ with more than $n_F(h')$ nodes, then $u$ and the children of $u$ belong to $T_F(h, h')$. We denote by $nl_F(h, h')$ the number of leaves of tree $T_F(h, h')$.*

**Lemma 3** *If $h \geq h' > 0$, then $nl_F(h, h') = f_{h-h'+2}$.*

**Proof.** We proceed by induction on $h - h'$. If $h = h'$ then $T_F(h, h')$ is just the root of $F_h$, and so $nl_F(h, h') = 1 = f_2$. If $h = h' + 1$ then $T_F(h, h')$ is the root of $F_h$ together with its two sons, thus $nl_F(h, h') = 2 = f_3$.

In the general case, $nl_F(h, h') = nl_F(h-1, h') + nl_F(h-2, h')$ and, by inductive hypothesis, $nl_F(h, h') = f_{h-h'+1} + f_{h-h'} = f_{h-h'+2}$. □

Recalling that $n_F(h) = f_{h+3} - 1$, we have that $nl_F(h, h') = n_F(h - h' - 1) + 1$.

**Definition 4** *For any $h \geq i \geq 0$, we denote with $nn(i, h)$ the number of nodes whose level is equal to or smaller than $i$ in tree $F_h$.*

The algorithm for drawing Fibonacci trees described in Figure 3 works as follows. Let $h_1 = \lfloor h/2 \rfloor$ and $h_2 = h - h_1 - 1$. First, tree $T_F(h, h_1)$ is drawn in the upper part. This tree has $nl_F(h, h_1) = n_F(h_2) + 1$ leaves, and in each of these leaves either a copy of $F_{h_1}$ or a copy of $F_{h_1-1}$ is rooted. Each of these trees is drawn levelwise; the separation between nodes at different levels is done in a worst-case way, using function $nn$. The parameter $x_f$ is the $x$-position in

```
algorithm FibDraw(T);
  procedure RecDraw(T, x_f, l);
  begin
    if T ≠ e then begin
      let u be the radix of T;
      if T has more than n_F(h_1) nodes then begin ;
        { u is in the upper part of the drawing }
        let x = min{x' ≥ 1| point (x', l + 1) is free};
        let y = l + 1;
      end
      else if the father of u is the root of a tree with more than n_F(h_1) nodes then begin
        { u is a leaf of the upper part }
        let x = min{x' ≥ 1| point (x', h − h_1 + 1) is free};
        let y = h − h_1 + 1;
        let l = 0;
      end
      else begin
        { u is in the lower part }
        let x = x_f + 1;
        let y = min{y' ≥ h − h_1 + 1 + nn(l − 1, h_1)| point (x_f + 1, y') is free};
      end
      draw(u, x, y);
      RecDraw(left(T), x, l + 1);
      RecDraw(right(T), x, l + 1);
    end;
  end;

begin
  h = height(T); h_1 = ⌊h/2⌋;
  RecDraw(T, 0, 0);
end;
```

Fig. 3. The algorithm for drawing Fibonacci trees.

which the father of current node $u$ has been drawn (it's used in the lower part of the drawing), while parameter $l$ is the level of the current node $u$. The value of $l$ has two different meanings depending on node $u$: if $u$ is in the upper part, then $l$ is its level in tree $T$; if $u$ is in the lower part, then $l$ is the level of $u$ *in the subtree it belongs to*. In both cases, $l$ is used to decide the vertical position where $u$ has to be drawn.

**Theorem 5** *Algorithm* `FibDraw` *finds a planar upward drawing for the Fibonacci tree* $T = F_h$ *with* $n = n_F(h)$ *nodes in a grid of area smaller than* $1.171n + O(\log n\sqrt{n})$.

**Proof.** Similarly to the proof of Theorem 1, we can show that algorithm `FibDraw` always finds a planar upward drawing of the Fibonacci tree given in input.

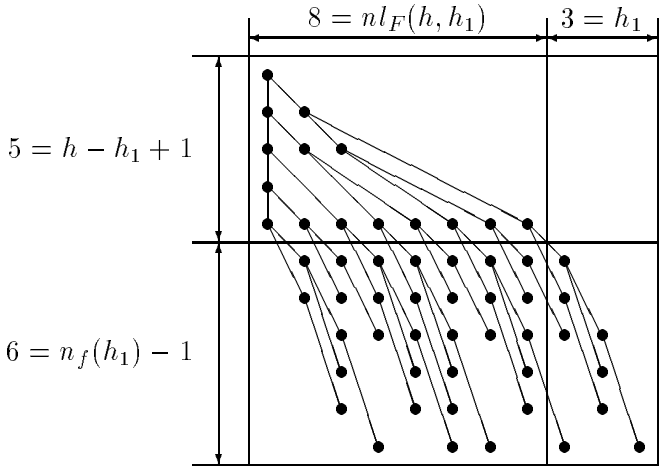


Fig. 4. How to draw the Fibonacci tree with 54 nodes in a $11 \times 11$ grid.

Let us now consider the area requirement of this algorithm. The upper part of the drawing has height $h - h_1 + 1$ and width $nl_F(h, h_1) = n_F(h_2) + 1$; the lower part has height $nn(h_1, h_1) - 1 = n_F(h_1) - 1$ and width $h_1 + n_F(h_2) + 1$. It is known [5,7] that

$$n_F(h) = \frac{1}{\sqrt{5}} \left( \frac{1+\sqrt{5}}{2} \right)^{h+3} + \Theta(1).$$

Thus we can express the area $A_F(h)$ required to draw tree $T_F(h)$ as

$$\begin{aligned} A_F(h) &= (h - h_1 + n_F(h_1))(h_1 + 1 + n_F(h_2)) \\ &< (h/2 + 1 + n_F(h_1))(h/2 + 1 + n_F(h_2)) \\ &\leq n_F(h_1)n_F(h_2) + (h+2)n_F(h_1) + h^2/4 + h + 1 \\ &= \frac{1}{5} \left( \frac{1+\sqrt{5}}{2} \right)^{h+5} + O(h\sqrt{n_F(h)}) \\ &= \frac{1}{\sqrt{5}} \left( \frac{1+\sqrt{5}}{2} \right)^{2} n_F(h) + O(h\sqrt{n_F(h)}). \end{aligned}$$

Let $n = n_F(h)$; since $h = \Theta(\log n)$, we have that $A_F(h) \leq \frac{1}{\sqrt{5}} \left( \frac{1+\sqrt{5}}{2} \right)^2 n + O(\log n \sqrt{n}) < 1.171n + O(\log n \sqrt{n})$.

□

In Figure 4 an example of the output of the algorithm is given with $h = 7$. A tree with 54 nodes is drawn in a $11 \times 11$ grid. Note that $h_1 = h_2 = 3$ and

indeed the tree in the upper part of the drawing has $nl_F(7,3) = n_F(3)+1 = 8$ leaves.

## 4 Conclusions and open problems

We presented an algorithm that upward draws a complete $m$-ary tree into a grid of area $n+O(\sqrt{n}\log n)$, and an algorithm that upward draws a Fibonacci tree into a grid of area $1.171n + O(\sqrt{n}\log n)$. This improves over the area requirement of previous algorithms [2,3,6]. It is not clear whether it is possible to draw any AVL tree (or, at least, any Fibonacci tree) in area $n + o(n)$. Moreover, it is an open question whether we can find *minimum area* upward drawings for any interesting class of trees. If we restrict ourselves to *square* grids (i.e., if we impose that the height of the grid must equal the width), we can prove that $n + \Omega(\sqrt{n}\log n)$ is a lower bound on the area of any upward tree drawing, and thus, in this case, our algorithm for complete trees is not far from being optimum (the $n + O(\sqrt{n}\log n)$ upper bound can be achieved also on square grids). We conjecture that it is indeed possible to find in polynomial time the minimum area upward drawing for complete trees. By the way, Mahaney's theorem implies that if such a problem is not in $P$, then it is not $NP$-complete either [9], so that it should be very difficult to prove its intractability.

## References


[1] G. Di Battista, P. Eades, and R. Tamassia. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*. To appear.

[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] P. Crescenzi and A. Piperno. Optimal-area upward drawings of AVL trees. In *Proceedings of Graph Drawing 94*, pages 307–317, 1994. LNCS 894.

[4] P. Eades, T. Lin, and X. Lin. Minimum size h-v drawings. In *Advanced Visual Interfaces*, pages 386–394, 1992.

[5] L. Euler. Observationes analyticae. *Novi commentarii academiae scientiarum Petropolitanae*, 11:124–143, 1765.

[6] A. Garg, M.T. Goodrich, and R. Tamassia. Area-efficient upward tree drawing. In *Proceedings of ACM Symposium on Computational Geometry*, pages 359–368, 1993.

[7] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison Wesley, 1989.

[8] D.E. Knuth. *The Art of Computer Programming*. Addison Wesley, 1975.

[9] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture by Berman and Hartmanis. *Journal of Computer and System Sciences*, 25:130–143, 1982.