Area Requirement of Visibility Representations of Trees (Extended Abstract) *

Goos Kant † Giuseppe Liotta ‡ Roberto Tamassia § Ioannis G. Tollis ¶

Abstract

The area of a drawing of a graph is the area of the smallest rectangle with sides parallel to the $x-$ and $y-$axis that covers the drawing. In this paper we study the area required by 1- and 2-strong visibility representations of trees. We also present linear time algorithms for drawing 1- and 2-strong visibility representations of trees with optimal area.

1 Introduction

The problem of drawing a graph in the plane has received increasing attention recently due to the large number of applications [3]. Examples include VLSI layout, algorithm animation, visual languages and CASE tools. Vertices are usually represented by points and edges by simple open curves. Another interesting representation is to map vertices into horizontal segments and edges into vertical segments [8, 9]. Such a representation is called a *visibility representation*. In this paper we study the area requirement of various types of visibility representations of trees, and we present linear time algorithms for drawing such representations with optimal area.

The concept of *visibility* between objects plays an important role in various problems of computational geometry, where we say that two objects of a given set are *visible* if they can be joined by a segment which does not intersect any other object. Two objects of the set are *ϵ-visible* if they can be joined by a non-zero thickness band which doesn't intersect any other object. The objects are non overlapping. A visibility representation of a graph maps vertices into objects and edges into segments between visible vertex-objects. Various visibility representations have been considered in the literature, and received increasing attention recently (see [7] for an up to date overview).

Tamassia & Tollis [9] studied three types of visibility representations (weak, ϵ, and strong) of graphs. A *weak visibility representation* maps vertices to horizontal segments and edges to vertical segments having only points in common with the pair of horizontal segments corresponding to the vertices they connect. Algorithms for constructing weak visibility representations were presented in [9] and independently in [8]. This type of representation has become a core item in the field of graph drawing. Recently, Kant [6] showed that such a visibility representation can be constructed in linear time on a grid of size at most $(\lfloor \frac{3}{2}n \rfloor - 2) \times (n-1)$, where n is the number of vertices of the input graph. A *strong visibility representation* maps vertices to horizontal segments such that two segments are visible if and only if the corresponding vertices are adjacent [9]. Tamassia & Tollis showed that every triangular planar graph and 4-connected planar graph has a strong visibility representation [9]. However, deciding whether a general planar graph has a strong visibility representation is NP-complete [1].

Several years after the publication of the first papers, researchers started the study of the 2-dimensional variant of this problem: vertices are represented by isothetic rectangles, and edges are represented by horizontal or vertical segments, having only points in common with the pair of rectangles corresponding to the vertices they connect. We only consider rectangles with non-zero area and sides parallel to the x-axis and y-axis. This representation is called *2-weak visibility representation*. In [11] Wismath proved that every planar graph admits a *2-ϵ visibility representation*, that is a 2-weak visibility representation representation with the additional property that two rectangles are ϵ-visible if and only if the corresponding vertices in the graph are adjacent. A *2-strong visibility representation* maps each vertex to a rectangle such that two rectangles are visible if and only if the corresponding vertices in the graph are adjacent.

Recently, the *area* of the representation has gained a lot of attention, especially in the field of graph drawing [9, 4, 5, 2]. For a complete survey on graph drawing see [3]. The *area* of a drawing is the area of the smallest rectangle with sides parallel to the $x-$ and $y-$axis that covers the drawing. The *width* and the *height* of the drawing are the width and the

*Research supported in part by the National Science Foundation under grant CCR-9007851, by the U.S. Army Research Office under grant DAAL03-91-G-0035, and by the Office of Naval Research and the Advanced Research Projects Agency under contract N00014-91-J-4052, ARPA order 8225. This work was performed in part at the Bellairs research Institute of McGill University.

†Department of Computer Science, Utrecht.University P.O. box 80.089, 3508 TB Utrecht, NL; e-mail: goos@cs.ruu.nl.

‡Dipartimento di Informatica e Sistemistica Universita' di Roma La Sapienza, 00185 Roma, Italy. This work has been done while this author was visiting the School of Computer Science of McGill University, Montreal; e-mail: liotta@infokit.ing.uniroma1.it.

§Department of Computer Science, Brown University, Providence, RI 02912-1910; e-mail: rt@cs.brown.edu.

¶Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688; e-mail: tollis@utdallas.edu.

height of the covering rectangle. We assume the existence of a *resolution rule*, which implies that the width and the height of a drawing cannot be arbitrarily scaled down. A typical resolution rule for 1- and 2- visibility representations is requiring for the endpoints of the vertex segments or vertex rectangles to be placed at the points of an integer grid. The existence of such a resolution rule naturally raises the problem of computing 1- and 2-visibility representations of a graph with minimum area.

We investigate the strong visibility problem for trees. The contribution is twofold. First we show lower bounds on the area occupied by any 1- and 2-strong visibility representation of trees. Next we present linear time drawing algorithms that obtain such representations achieving these bounds. Since in this paper we only study 1- and 2-*strong* visibility representations, we call them 1- and 2-visibility representations, for brevity.

The paper is organized as follows. Preliminaries are in Section 2; in Section 3 and in Section 4 we study the area requirement for 1- and 2-strong visibility representations of trees; finaly, we present conclusions and open problems in Section 5.

Omitted proofs can be found in the forthcoming full paper.

2 Definitions

Given an integer grid, and two points $p_1 \equiv (x_1, y_1)$ and $p_2 \equiv (x_2, y_2)$, a *horizontal strip* (*vertical strip*) between p_1 and p_2 is the set of points with y-coordinate in the set $[y_1, y_2]$ (x-coordinate in the set $[x_1, x_2]$). A *row* (*column*) of the grid is a vertical strip such that $y_2 = y_1 + 1$ ($x_2 = x_1 + 1$).

We denote with $S(v)$ the vertex segment (or rectangle when considering two dimensions) associated to a vertex v; $S(v)$ is identified by specifying its rightmost abscissa, its leftmost abscissa, its lower and upper abscissa. We denote this by $S(v) = (x(v), y(v)) = ([x_1, x_2], [y_1, y_2])$. If $x_1 = x_2$ ($y_1 = y_2$), then we denote $S(v) = (x_1, [y_1, y_2])$ ($S(v) = ([x_1, x_2], y_1)$) for short. We also denote x_1 by $x_L(v)$ and x_2 by $x_R(v)$. If T is a rooted tree, the *height* of T, denoted by $h(T)$, is the length of the longest path from the root of T to any leaf; an *ancestor* of a vertex v is any vertex $v' \neq v$ in the path from the root of T to v.

Given a drawing Γ we denote with $x_R(\Gamma)$ ($x_L(\Gamma)$) the rightmost (leftmost) x-coordinate of Γ; we denote with $y(\Gamma)$ the maximum y-coordinate of Γ. Let $\Gamma' \subseteq \Gamma$, the *right* (*left*) of Γ' is the subdrawing of Γ whose points don't belong to Γ' and have x−coordinate greater than or equal to (less than or equal to) $x_R(\Gamma')$ ($x_L(\Gamma')$); the right and the left of Γ' are denoted with $RIGHT(\Gamma')$ and $LEFT(\Gamma')$, respectively.

3 1-Visibility Representations

In this section we tackle the problem of the area required by 1-visibility representations of trees. We distinguish the cases of rooted trees and free trees.

3.1 Rooted Trees

Let T be a rooted tree. Then the following lemma holds.

Lemma 1 *Let T be a rooted tree with l leaves and height $h(T)$. The area required by a 1-visibility representation of T is $\Omega(h(T) \cdot l)$.*

Proof: Let $\Gamma(T)$ be a 1-visibility representation of T. To avoid overlappings, no two leaves of T can be represented by two vertex segments in the same column of the grid. Thus the width of $\Gamma(T)$ is at least l. Also, each vertex segment of $\Gamma(T)$ must lie in a strip below its ancestors. Thus the height of $\Gamma(T)$ is at least $h(T)$. □

To obtain a 1-visibility of a rooted tree, we introduce the following algorithm (notice that in a *post-order* numbering, the children of a vertex v are numbered before v):

Algorithm 1-ROOTED

Input: Rooted tree T.

Output: 1-visibility representation of T.

1. Enumerate the leaves of T from left to right in increasing order by assigning an integer in the set $[k, \ldots, k+l-1]$ ($k \geq 0$ is any arbitrarily fixed integer).
2. Draw each leaf i with a vertex segment $S(i) = ([2i, 2i+1], 0)$.
3. Number the internal vertices of T in post-order $v_1, \ldots, v_{n-l}$; let $v_{i_1} \ldots v_{i_j}$ be the children of v_i from left to right; let $y_{max} = max_{1 \leq k \leq j}\{y(v_{i_1}), \ldots, y(v_{i_j})\}$; draw v_i as a vertex segment $S(v_i) = ([x_L(v_{i_1}), x_R(v_{i_j})], y_{max})$.

end Algorithm.

In Fig. 1 an example of the output of Algorithm 1-ROOTED is given.

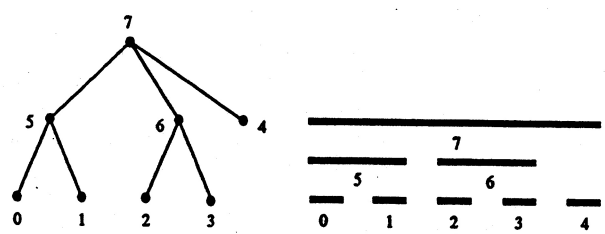

Figure 1: Illustration of Algorithm 1-ROOTED

Lemma 2 *Let T be a rooted tree with l leaves, n vertices, and height h. Algorithm* 1-ROOTED *produces a 1-visibility representation of T with area $(2l-1) \cdot h$ in $O(n)$ time.*

Theorem 1 *Let T be a rooted tree with n vertices, l leaves and height h. The area required by a 1-visibility representation $\Gamma(T)$ of T is $\Theta(h \cdot l)$. Also $\Gamma(T)$ can be computed in $O(n)$ time.*

3.2 Free Trees

A tree where no vertex has been chosen as the root is called a *free tree*. In this subsection we study the area required by a 1-visibility representations of a free tree. Let T be a free tree. Let v be a vertex of T and let $T_v^1, \ldots, T_v^k$ be the subtrees obtained by removing v and the edges incident on v. We root each subtree T_v^i at the unique vertex of T_v^i, adjacent to v in T. We assume that always $h(T_v^1) \geq h(T_v^2) \geq \ldots \geq h(T_v^k)$ in this paper. T_v^k is called the k-th highest subtree of v.

We denote with T_v the tree obtained by deleting from T the first and the second subtree of v and the incident edge of v to T_v^1 and to T_v^2. Root T_v at v. We call the *third vertex* of T the vertex v^* such that the height of the third highest subtree of v^* is maximum, i.e., for which $h(T_{v^*})$ is maximum. Let k^* be the corresponding height, i.e., $k^* = h(T_{v^*})$.

Lemma 3 *Let v be a vertex in a subtree of v^*; let T_v^1 and T_v^2 be the first and the second heighest subtrees of v respectively. Vertex v^* belongs to T_v^1 or T_v^2.*

Lemma 4 *Let T be a free tree with n vertices and l leaves; let k^* be the height of the third subtree of the third vertex of T. The area required by a 1-visibility representation of T is $\Omega(k^* \cdot l + n)$.*

Proof: Each vertex segment of $\Gamma(T)$ is associated to at least one cell of the grid; thus, the area of $\Gamma(T)$ is at least n. Also, to avoid overlappings, no two leaves of T can be represented by two vertex segments in the same column of the grid, unless they lie one in the strip above and the other one in the strip below the vertex segment representing a common ancestor; thus the width of $\Gamma(T)$ is at least $\lceil \frac{l}{2} \rceil$. Let $S(v)$ be the horizontal vertex segment representing a vertex v of T. All but two subtrees of v must be drawn in the vertical strip between the endpoints of $S(v)$. It follows that the height of $\Gamma(T)$ is at least the height of the third largest subtree among all subtrees of vertices with degree at least three. Hence the height is at least k^*. □

We discuss first the case that T has *internal vertices* (i.e. non-leaf vertices) with degree at least 3. The proposed algorithm consists of two main phases. In the first phase we find node v^* and we compute a special path called *main path*. In the second phase we define the drawing of the subtree T_v of each vertex v of the main path and we construct the drawing of the input tree.

Procedure PATH

Input: Free tree T with all internal vertices of degree at least 3.

Output: Main path of T.

1. Find v^*; root T at v^*.
2. Label each vertex v of T with a label $L(v)$ in the set $\{U, D\}$ as follows; $L(v^*) = D$; perform a pre-order traversal of T; for each encountered vertex v delete T_v and mark its children in the remaining tree with an U if $L(v) = D$, with an D if $L(v) = U$.

End Procedure

The main path is the path P_{main} of T containing all labelled vertices of T. Observe that P_{main} contains two leaves of T. In Fig. 2 we show vertex v^*, and the main path of a given free tree.

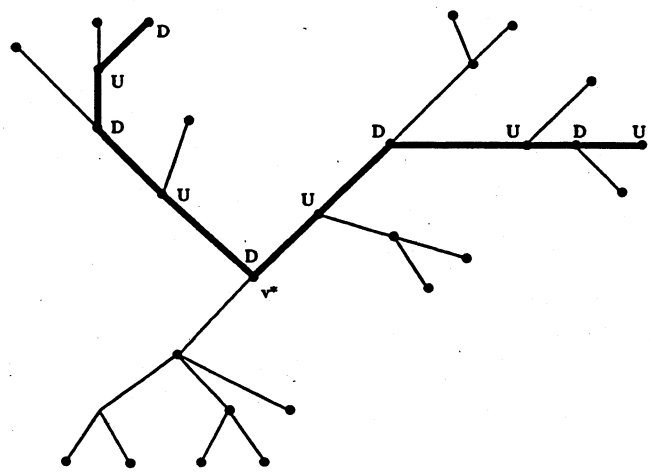

Figure 2: An example of the main path

Lemma 5 *Let T be a free tree and let P_{main} be the main path of T. The following formula holds.*

$$(\cup_{v \in P_{main}} T_v) \cup P_{main} = T.$$

Lemma 6 *Let T be a free tree with n vertices. Procedure* PATH *computes the main path of T in $O(n)$ time.*

In the following procedure v_i is a vertex of P_{main} and $\Gamma(T_{v_i})$ is the drawing obtained applying Algorithm 1-ROOTED to T_{v_i}.

Procedure DRAW TREE

Input: Free tree T rooted at v^*, P_{main}.

Output: 1-visibility representation of T.

1. Number the vertices of P_{main} in increasing order $v_1 \ldots v_k$ from one leaf to the other one.
2. Apply Algorithm 1-ROOTED to T_{v_1}; if $L(v_1) = U$, then increment $x_R(v_1)$ of one unit.
3. Visit P_{main} starting from v_2; for any encounterd vertex v_i do the following: (i)apply Algorithm 1-ROOTED to T_{v_i}; (ii)set $x_L(\Gamma(T_{v_i})) = x_R(\Gamma(T_{v_{i-1}})) + 1$; (iii)if $L(v_i) = D$ then set $y(\Gamma(T_{v_i})) = y(\Gamma(T_{v_{i-1}})) - 1$; (iv)if $L(v_i) = U$ then set $y(\Gamma(T_{v_i})) = y(\Gamma(T_{v_{i-1}})) + 1$ and enlarge $S(v_i)$ of one unit to the left and one unit to the right.

End Procedure

In Fig. 3 we show an example of output of Procedure DRAW TREE. The input graph is the one of Fig 2.

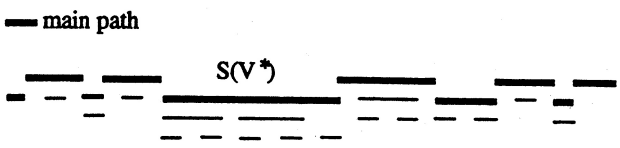

Figure 3: Illustration of the Procedure DRAW TREE

Observe that all vertices of P_{main} with the same label are represented by vertex segments with the same y–coordinate.

Lemma 7 *Let T be a free tree with n vertices and l leaves; let k^* be the height of the third subtree of the third vertex of T. Procedure* DRAW TREE *produces a 1-visibility representation $\Gamma(T)$ of T in $O(n)$ time. Also the area required by $\Gamma(T)$ is is $O(k^* \cdot l)$.*

Suppose now T has vertices of degree 2. Clearly, if T is a path a 1-visibility representation of T requires linear area and can be computed in linar time. Hence, suppose there are some vertices of T with degree at least three. In this case it is possible to define the main path of T by slightly modifying Procedure PATH. Namely, we compute first v^* and mark v^*. We also mark the two neighbors of v^*, say v_1 and v_2, belonging to $T^1_{v^*}$ or $T^2_{v^*}$. Similar we mark the two neighbors of v_1, say v_{1_1} and v_{1_2}, belonging to $T^1_{v_1}$ or $T^2_{v_1}$ (by Lemma 3, $v_{1_1} = v^*$ or $v_{1_2} = v^*$). The same is done for the two neighbors of v_2. Repeating this argument yields a path of marked vertices between two leaves that is the main path P_{main}. Also, the marking operation can be done in such a way that all the vertices of P_{main} that in T don't have degree two are alternatively labelled U or D. Observe now that if no vertex $v \in T$ such that $deg(v) = 2$ is in the main path, Procedure DRAW PATH can be applied and Lemma 7 still holds.

In order to tackle the case when there is a subset of vertices of degree two in the main path of T, we need the following procedure.

Procedure DRAW PATH

Input: Path P, k^*, x, y.

Output: 1-visibility representation of P.

1. Number the vertices of P in increasing order $v_1, \ldots, v_m$.
2. Divide P in subpaths, each one (except, eventually the one containin g vertex v_m) composed by k^* vertices.
3. Mark the subpaths with *down* or *up* according to the following rule. The first subpath P_1(i.e. the subpath including v_1) is marked *down*; a subpath $P_j = (v_j, \ldots, v_k)$ is marked *up* if the predecessor of v_j belongs to a subpath marked *down*; otherwise P_j is marked *down*.
4. Draw a subpath $P_d = (v_p, \ldots, v_q)$ marked *down* as follows. If $v_p = v_1$ then draw v_1 as a vertex segment $S(v_1) = ([x, x+1], y)$, else draw v_p as $S(v_p) = ([x', x'+1], y'-1)$, where $x' = x_R(v_{p-1})$ and $y' = y(v_{p-1})$; draw any vertex v_i $(p < i < q)$ as $S(v_i) = ([x', x'+1], y_i)$ where $y_i = y(v_{i-1}) - 1$; draw v_q as $S(v_q) = ([x'+1, x'+2], y_q)$ where $y_q = y(v_{q-1}) - 1$.
5. Draw a subpath $P_u = (v_r, \ldots, v_s)$ marked *up* as follows. Dra w v_r as $S(v_r) = ([x', x'+1], y'+1)$, where $x' = x_R(v_{r-1})$ and $y' = y(v_{r-1})$; draw any vertex v_i $(r < i < s)$ as $S(v_i) = ([x', x'+1], y_i)$ where $y_i = y(v_{i-1})+1$; draw v_s as $S(v_s) = ([x'+1, x'+2], y_s)$ where $y_s = y(v_{s-1})+1$.

End Procedure

An example of a path drawn by this procedure is given later in the paper (see Fig. 4)

Lemma 8 *Let P be a path with n vertices. Procedure* DRAW PATH *produces a 1-visibility representation $\Gamma(P)$ of P in $O(n)$ time. Also the area required by $\Gamma(P)$ is $O(n)$.*

We are now ready to give Algorithm 1-FREE TREE.

Arbitrarely fixed one sense of percorrence in P_{main}, suppose there is a path $P \subseteq P_{main}$ joining a vertex v' and a vertex v of P_{main}such that (i)all vertices belonging to P are vertices of T with degree two; (ii)$deg(v') \geq 3$ and $deg(v) \geq 3$; (iii)when going along P_{main} in positive direction, vertex v' is encountered before vertex v. Vertex v' is the *predecessor* of path P and vertex v is the *successor* of path P. The basic idea of Algorithm 1-FREE TREE is to draw first tree T as if no vertices of degree two were in P_{main}. Then, we apply Procedure Draw Path to P and we insert $\Gamma(P)$ in the drawing between its predecessor and its successor. In what follows we adopt the following notation; $\Gamma(P)$ is the drawing of P produced applying Procedure DRAW PATH to P; W is the width of $\Gamma(P)$; $\Gamma(T_v)$ and $\Gamma(T_{v'})$ are the drawings obtained applying Algorithm 1-ROOTED to T_v and $T_{v'}$; y' is the y–coordinate of any vertex segment labelled D.

Algorithm 1-FREE TREE

Input: Tree T.

Output: 1-visibility representation of T.

1. Compute P_{main}.
2. Apply Procedure DRAW TREE ignoring any vertex of P_{main} that is not labelled U or D. Let $\Gamma(T')$ be the obtained drawing.
3. For each path P that has been ignored in the Step 2 do the following: (i)apply Procedure DRAW PATH with paramters k^*, $x = x_R(\Gamma(T_{v'}))+1$ and $y = y'-2$. (ii)Let $RIGHT((\Gamma(T_{v'}))$ be the right of $\Gamma(T_{v'})$ in $\Gamma(T')$; add $W+1$ to all x–coordinates of $RIGHT((\Gamma(T_{v'}))$. (iii)Increment $x_R(v')$ of one unit; decrement $x_L(v)$ of one unit.

End Algorithm

In Fig. 4 an example of path P between $(\Gamma(T_{v'}))$ and $(\Gamma(T_v))$ is shown.

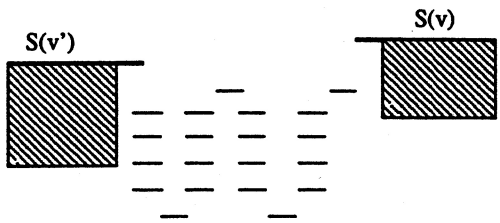

Figure 4: Illustration of Step 3 of Algorithm 1-FREE TREE

Lemma 9 *Let T be a free tree with n vertices and l leaves; let k^* be the height of the third subtree of the third vertex of T. Algorithm* 1-FREE TREE *produces an 1-visibility representation $\Gamma(T)$ of T in $O(n)$ time. Also the area required by $\Gamma(T)$ is is $O(k^* \cdot l + n)$.*

Theorem 2 *Let T be a free tree with n vertices and l leaves; let k^* be the height of the third subtree of the third vertex of T. The area required by a 1-visibility representation $\Gamma(T)$ of T is $\Theta(k^* \cdot l + n)$. Also $\Gamma(T)$ can be computed in $O(n)$ time.*

4 2-Visibility Representations

In this section we consider the problem of constructing 2-visibility representations of free trees. We present linear time algorithms for this problem and show that the required grid size is asymptotically optimal.

Let T be a free tree with n vertices and l leave. Every leaf v_k has only one neighbor, say v'_k. For any $S(v_k) = ([x_1, x_2], [y_1, y_2])$ we can define four strips; the *above strip* is composed by all points (x, y) with $x \in [x_1, x_2]$ and $y \geq y_2$; the *right strip* is composed by all points (x, y) with $y \in [y_1, y_2]$ and $x \geq x_2$; similarly are defined the *below strip* and the *left strip*. The above and below strips are vertical strips, the right and left strip are horizontal strips. Observe that for each $S(v_k)$, only $S(v'_k)$ can intersect one of such strips (otherwise it would be visibility between not-adjacent vertices). This observation yelds the following lemma.

Lemma 10 *Let T be a free tree with l leaves. A 2-visibility representation of T requires an area $\Omega(l \cdot l)$.*

Proof: For every leaf at least one vertical and one horizontal strip are *wasted*, i.e. no other vertex can intersect them. Since for every row (column) of the grid we can define two horizontal (vertical) strips, this yields a grid of size at least $\lceil \frac{l}{2} \rceil \cdot \lceil \frac{l}{2} \rceil$. □

Similarly we can prove that for every vertex of degree 2 at least one vertical or horizontal strip is wasted. Let k be the number of vertices of degree 2, then in every 2-visibility representation the height or width has size at least $\lceil \frac{k}{2} \rceil$. Combining this result with Lemma 10 gives the following lemma.

Lemma 11 *Let T be a free tree with n vertices and l leaves. A 2-visibility representation of T requires an area $\Omega(l \cdot n)$.*

Proof: Drawing the leaves already requires $\lceil \frac{l}{2} \rceil$ columns and $\lceil \frac{l}{2} \rceil$ rows. The vertices of degree 2 also require $\lceil \frac{k}{2} \rceil$ columns or $\lceil \frac{k}{2} \rceil$ rows. The sum of the degrees of all vertices is $2(n-1)$. Let p be the number of vertices with degree at least 3, then $p = n - k - l$. It follows that $l + 2k + 3p \leq 2(n-1)$, thus $p \leq l - 2$. Hence $2(n-1) \geq l + 2k + 3p \geq 2k + 4p + 2 \geq 4p + 2$. $p \leq \frac{2n-2-2}{4} = \frac{n}{2} - 1 \leq \lfloor \frac{n}{2} \rfloor$. Thus $k + l \geq \lceil \frac{n}{2} \rceil$, which completes the proof. □

The algorithm for constructing a 2-visibility representation of a tree T is now as follows:

Algorithm 2-FREE TREE

Input: Free tree T.

Output: 2-visibility representation of T.

1. Root T at an arbitrary vertex v.
2. Enumerate the leaves of T from left to right in increasing order by assigning an integer in the set $[1, \ldots, l]$.
2. Draw each leaf i with a vertex rectangle $S(i) = [2i, 2i+1], [2i, 2i+1])$.
3. Number the internal vertices of T in post-order $v_{l+1}, \ldots, v_n$.
4. Visit the vertices v_i $(l+1 \leq i \leq n)$ in increasing order; let $v_{i_1} \ldots v_{i_j}$ be the children of v_i from left to right; Draw v_i as a vertex rectangle $S(v_i) = ([x_L(v_{i_1}), \ldots, x_R(v_{i_k})], [2i, 2i+1])$.

end Algorithm.

In Figure 5 an example of a 2-visibility representation is given.

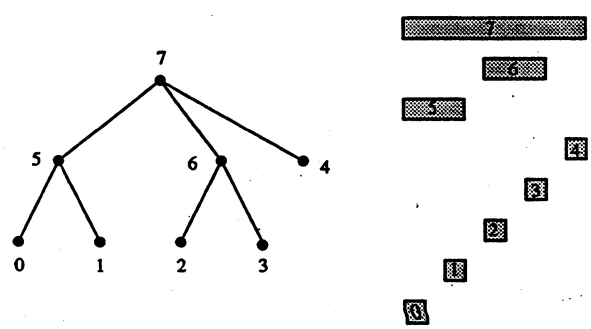

Figure 5: Illustration of Algorithm 2-FREE TREE.

Lemma 12 *Let T be a free tree with n vertices and l leaves. Algorithm* 2-FREE TREE *produces an 2-visibility representation $\Gamma(T)$ of T in $O(n)$ time. Also the area required by $\Gamma(T)$ is $O(l \cdot n)$.*

Proof: It follows directly that the algorithm can be implemented to run in linear time. The size of the grid is $(2l-1) \cdot (2n-1)$. To show that the drawing indeed is a 2-visibility representation we notice that $S(v_i)$ is placed between heights $2i$ and $2i+1$, hence no two rectangles can see each other in horizontal direction. Assume for each vertex v_i that $x(v_i) = [x_{i_1}, x_{i_2}]$ with $x_{i_2} \geq x_{i_1}$. Let $v_l = parent(v_k)$ for an arbitrary vertex v_k, then $x_{l_1} \leq x_{k_1} \leq x_{k_2} \leq x_{l_2}$ and by the post-order numbering, there is no vertex v_i, with $k < i < l$ such that $x_{i_1} \leq x_{k_1} \leq x_{k_2} \leq x_{i_2}$. Hence in vertical direction only $S(v_l)$ is visible from $S(v_k)$. Since the same holds for the children of v_k with respect to v_k, this completes the proof. □

Theorem 3 *Let T be a tree with n vertices, l leaves and height h. The area required by a 2-visibility representation $\Gamma(T)$ of T is $\Theta(l \cdot n)$. Also $\Gamma(T)$ can be computed in $O(n)$ time.*

5 Conlusions and Open Problems

In this paper we have studied the area provided by 1- and 2-strong visibility representations. We have provided linear time algorithms for drawing 1- and 2-strong visibility representation of trees with optimal area. This is the first time that algorithms for 2-strong visibility representations are presented. However, several open problems remain. Indeed, the most natural question is to determine which kind of graphs can be represented by a 2-strong visibility representation? Even for series-parallel and planar graphs this problem is open, and is left for the interested reader.

Secondly, our algorithms are asymptotically optimal with respect to some existential lower bounds. It is interesting to study whether the algorithms are always asymptotically optimal, and to improve the constant factors in the grid size.

Acknowledgments

This research is a consequence of the authors' participation to the Workshop on Visibility Representation of Graphs organized by Sue Whitesides and Joan Hutchinson at the Bellairs Research Institute of McGill University, Feb. 12-19, 1993. We thank the other participants of the Workshop for useful discussions.

References

[1] T. Andreae, Some results on visibility graphs, 1989, preprint.

[2] P. Crescenzi, G. Di Battista, and A. Piperno, A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees, to appear in *Comp. Geometry: Theory and Applications.*

[3] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis Algorithms for Automatic Graph Drawing: An Annotated Bibliography, Dept. of Comp. Science, Brown Univ., Technical Report, 1993. Available via anonymous ftp from wilma.cs.brown.edu (128.148.33.66), files /pub/gdbiblio.tex.Z and /pub/gdbiblio.ps.Z.

[4] G. Di Battista, R. Tamassia and I.G. Tollis, Constrained Visibility Representations of Graphs, *Inform. Process. Letters* 41 (1992), pp. 1–7.

[5] G. Di Battista, R. Tamassia and I.G. Tollis, Area Requirement and Symmetry Display of Planar Upward Drawings, *Discr. and Comp. Geometry* (1992), pp. 381–400.

[6] G. Kant, A More Compact Visibility Representation, in: J. van Leeuwen (Ed.), *Proc. 19th Intern. Workshop on Graph-Theoretic Concepts in Comp. Science (WG'93)*, Lecture Notes in Comp. Science, Springer-Verlag, 1993, to appear.

[7] J. O'Rourke, Computational geometry column 18, *Int. Journal of Comp. Geometry & Appl.* 3 (1993), pp. 107–113.

[8] P. Rosenstiehl, and R. E. Tarjan, Rectilinear Planar Layouts and Bipolar Orientations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 343–353.

[9] R. Tamassia, and I. G. Tollis, A Unified Approach to Visibility Representations of Planar Graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 321–341.

[10] S.K. Wismath, Weighted Visibility Graphs of Bars and Related Flow Problems, in: *Proc. Workshop on Algorithms and Data Structures (WADS'89)*, Lecture Notes in Computer Science 382, Springer-Verlag, 1989, pp. 325–334.

[11] S.K. Wismath, Bar-Representable Visibility Graphs and Related Flow problems, Dept. of Comp. Science, Univ. of British Columbia, Technical Report, 1989.