Area Requirement of Visibility Representations of Trees

G. Kant, G. Liotta, R. Tamassia and I.G. Tollis

RUU-CS-93-33
October 1993

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Area Requirement of Visibility Representations of Trees

G. Kant, G. Liotta, R. Tamassia and I.G. Tollis

Technical Report RUU-CS-93-33
October 1993

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924-3275

Area Requirement of Visibility Representations of Trees*

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

Abstract

Trees are among the most common structures in computing and many algorithms for drawing trees have been developed in the last years. Such algorithms usually adopt different drawing conventions and attempt to solve several optimization problems. The aim of this paper is to study two different types of drawing conventions for trees, namely 1-strong visibility representation and 2-strong visibility representation. For both of them we investigate the problem of minimizing the area of the representation. The contribution of the paper is twofold: (*i*) we prove tight lower and upper bounds on the area of such representations; and (*ii*) we provide linear-time algorithms that construct representations with optimal area.

*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, by the Office of Naval Research and the Advanced Research Projects Agency under contract N00014-91-J-4052, ARPA order 8225, by ESPRIT Basic Research Actions of the EC under contract No. 7141 (project ALCOM II), and by Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the Italian National Research Council (CNR). This work was performed in part at the Bellairs research Institute of McGill University. An extended abstract of this paper has been presented in the Fifth Canadian Conference on Computational Geometry, Waterloo, Aug. 5–9, 1993.

[†]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.

1 Introduction

The problem of drawing a graph in the plane has received increasing attention recently due to the large number of applications, including VLSI layout, algorithm animation, visual languages and CASE tools. Vertices are usually represented by points and edges by simple Jordan curves. Graph drawing algorithms attempt to construct a representation of the input graph that satisfies a set of geometric properties, such as minimizing edge crossings, emphasizing symmetries and minimizing the size of the required area. For an up to date overview on graph drawing problems and algorithms, the reader is referred to the bibliography by Di Battista, Eades, Tamassia and Tollis [6].

The *area* of a drawing of a graph is the area of the smallest covering rectangle with sides parallel to the axes, where we assume the existence of a *resolution rule* that prevents the drawing from being arbitrarily scaled down. Typical resolution rules are requiring integer coordinates or a minimum distance between vertices. Asymptotic bounds on the area of drawings of graphs do not depend on the specific choice of resolution rule. The *width* and the *height* of the drawing are the width and the height of the covering rectangle. Since the first results on graph drawing were published, the area required by different drawing conventions has been one of the most intriguing problems in the field.

A *planar drawing* of a graph is a drawing where no two edges intersect, except possibly at their endpoints. A *planar* graph is a graph that admits a planar drawing. It is well known that every planar graph admits a *straight-line* planar drawing, where vertices are represented by points on the plane and edges by segments between pairs of adjacent vertices. De Fraysseix, Pach and Pollak [5, 4] and, independently, Schnyder [21] showed that a planar graph with n vertices has a planar straight-line drawing with $O(n^2)$ area.

An *upward* drawing of a directed graph is such that every edge is a directed curve monotonically nondecreasing in the vertical direction. Upward drawings are ofter used to visualize hierarchic structures. Only acyclic digraphs admit upward drawings.

In this paper we consider the area-efficient planar drawings of trees. Let T be a tree with n vertices. Crescenzi Di Battista and Piperno [3] showed that if T is a complete binary tree or a Fibonacci tree, then T admits an upward planar straight-line drawing with $O(n)$ area; Garg, Goodrich and Tamassia [12] proved that if T is a rooted tree, then T has an upward planar drawing with $O(n)$ area. They also showed that if T is a binary tree, then T has an upward planar orthogonal drawing with $O(n \log\log n)$ area, and that this area bound is optimal in the worst-case. If T is not rooted and the upward requirement is relaxed, then, as independently shown by Leiserson [15] and Valiant [26], T admits an $O(n)$-area planar orthogonal drawing. Eades, Lin and Lin [9] studied upward drawings of rooted trees where the covering rectangles of the subtrees of each node are disjoint and show how to optimize the area of such drawings.

The concept of *visibility* (see, e.g., [18, 16, 22, 10, 11, 17, 24]) plays an important role in computational geometry, and arises in art gallery problems, motion planning, and graphics. The study of *visibility representations* of graphs was originally motivated by VLSI layout and compaction problems (see, e.g., [23, 20]). Further applications of visibility representations concern PERT diagrams (see e.g., [7, 8]) and orthogonal graphs (see, e.g., [25]).

Given a set of disjoint objects in the plane (e.g., points, lines, rectangles), two objects of are said to be *1-visible* if they can be joined by a vertical segment, called *visibility segment* that does not intersect any other object. Two 1-visible objects are said to be $\epsilon-$ visible if they can be joined by a vertical band of nonzero width, called *visibility band*, that does not intersect any other object in the set.

A *1-visibility representation* of a graph G maps the vertices of G to disjoint horizontal segments such that any two adjacent vertices are associated with 1-visible segments. Graph G admits a 1-visibility representation only if it is planar. Variations of this representation are called *1-weak*, *1-strong* and *1-ϵ* visibility representations, and are illustrated in Figure 1. A 1-visibility representation of a directed graph G is said ot be upward if for every edge (u, v), the vertex-segment of u is placed below the vertex-segment of v.

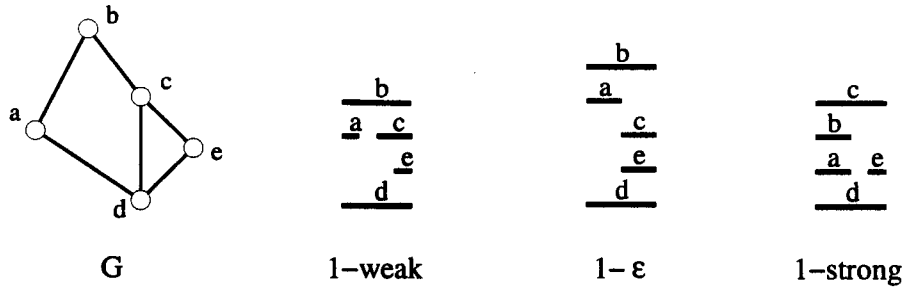

Figure 1: Example of 1-weak, 1-ϵ, and 1-strong visibility representations of a graph G.

In a *1-weak visibility representation*, adjacent vertices are associated with visible vertex-segments. However, visible vertex-segments are not necessarily associated with adjacent vertices. Linear-time algorithms for constructing 1-weak visibility representations of planar graphs were presented by Tamassia and Tollis [24] and, independently, by Rosenstiehl and Tarjan [19]. Recently, Kant [13] showed that a 1-weak visibility representation of an n-vertex planar graph with area at most $(\lfloor \frac{3}{2}n \rfloor - 3) \times (n-1)$ can be constructed in $O(n)$ time.

In a *1-strong visibility representation*, two vertex-segments are visible *if and only if* their associated vertices are adjacent. Tamassia & Tollis [24] showed that

every 4-connected planar graph has a strong visibility representation. However, Andreae [1] proved that deciding whether a general planar graph has a strong visibility representation is NP-complete.

In a *1-ϵ visibility representation* two vertex-segments are $\epsilon-$ visible *if and only if* their associated vertices are adjacent. Efficient algorithms for constructing 1-ϵ visibility representations and recognizing those graphs that admit them were independently given by Wismath [27] and by Tamassia & Tollis [24]. Recently, Kirkpatrick and Wismath [14] have presented a polynomial time solution to the problem of determining whether a given weighted graph admits a 1-ϵ visibility representation in which the weight of each edge is equal to the width of the maximal visibility band joining the corresponding vertex-segments.

Several years after the publication of the first papers on 1-visibility representations, researchers started the study of *2-visibility representations* of graphs, where the vertices are represented by disjoint isothetic rectangles in the plane, and the edges are represented by visibility-segments (or bands) in the horizontal or vertical direction. As for the 1-visibility case, we distinguish *2-weak*, *2-strong* and *2-ϵ visibility representations*. See the example of Figure 2. The only previously published result on on 2-visibility representations is by Wismath [28], who proved that every planar graph admits a 2-ϵ visibility representation.

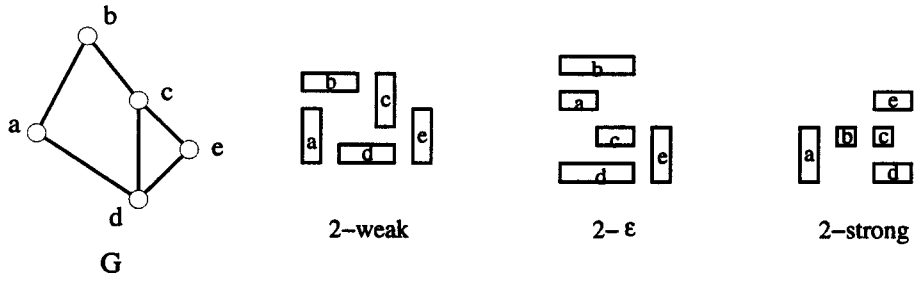

Figure 2: Example of 2-weak, 2-ϵ, and 2-strong visibility representations of a graph G.

In this paper we study the area required by 1-strong and 2-strong visibility representations of trees. As a resolution rule, we assume that vertex segments have integer coordinates. Our contribution is twofold: (*i*) we prove tight lower and upper bounds on the area of such representations; and (*ii*) we provide linear-time algorithms that construct representations with optimal area.

The paper is organized as follows. Preliminaries are given in Section 2. In Sections 3 and 4, we study the area requirement of 1- and 2-strong visibility representations of trees, respectively. Conclusions and open problems are discussed in Section 5.

For brevity, in the rest of this paper 1-strong visibility and 2-strong visibility representations will be simply called 1- and 2- visibility representations.

2 Preliminaries

We begin by defining some of the graph theoretic and geometric terminology used in this paper. For more details see [2] and [18].

2.1 Graphs and Trees

A *graph* $G = (V, E)$ consists of a finite nonempty set V of *vertices*, and a set E of unordered pairs of vertices known as *edges*. An edge e consisting of vertices u and v is denoted by $e = uv$; u and v are called the *endpoints* of e and are said to be *adjacent* vertices or *neighbors*. The *degree* of a vertex v, denoted by $deg(v)$, is the number of edges which have v as an endpoint. A *path* in a graph G is a finite non-null sequence $P = v_1 v_2 \ldots v_k$ where the vertices $v_1, v_2, \ldots, v_k$ are distinct and $v_i v_{i+1}$ is an edge for each $i = 1 \ldots k-1$. The vertices v_1 and v_k are known as the *endpoints* of the path. The *length* of a path P, denoted by $\ell(P)$, is the number of edges of P. A *cycle* is a path whose endpoints are the same. An *acyclic* graph is a graph that contains no cycles. A graph is *connected* if, for each pair of vertices $u, v \in V$, there is a path from u to v.

A *free tree* is a connected acyclic graph. If T is a free tree, a *leaf* of T is a vertex $v \in V(T)$, such that $deg(v) = 1$. A *rooted tree* is a free tree T along with a distinguished vertex called the *root* of T. The *height* of a rooted tree T, denoted by $h(T)$ is the length of a longest path from the root of T to a leaf; an *ancestor* of a vertex $v \in V(T)$ is any vertex $v' \neq v$ in the path from the root of T to v.

2.2 Visibility Representations

Given a set S of non overlapping horizontal segments in the plane. Two segments are *1-visible* if they can be joined by a vertical segment, which does not intersect any other segment.

A *1-visibility representation* of a graph G is a mapping of G into the plane such that (i) vertices of G are represented by non overlapping horizontal segments (*vertex-segments*); (ii) two vertex-segments are visible if and only if the corresponding vertices in G are adjacent.

Given a set S of nonoverlapping xy-rectangles in the plane, two rectangles of S are said to be *2-visible* if they can be joined by a horizontal or vertical segment which does not intersect any other rectangle.

A *2- visibility representation* of a graph G is a mapping of G on the plane such that (i) vertices of G are represented by non overlapping $xy-$rectangles

(*vertex-rectangles*); (ii) two vertex-rectangles are 2-visible if and only if the corresponding vertices in G are adjacent.

In this paper points, segments and rectangles will be drawn on an integer grid (i.e. our reference coordinates are integer numbers). Given two points $p_1 \equiv (x_1, y_1)$ and $p_2 \equiv (x_2, y_2)$ on the grid, 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 by $S(v)$ the vertex-segment (or vertex-rectangle) associated with vertex v; $S(v)$ is identified by specifying its rightmost and leftmost x-coordinates, and its lower and upper y-coordinates. 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 $x_R(v)$. For any vertex-rectangle $S(v_k) = ([x_1, x_2], [y_1, y_2])$ we can identify 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 $x \geq x_2$; the *below* and *left strips* are defined similarly. The above and below strips are vertical strips, the right and left strips are horizontal.

Finally, given a visibility representation of G, Γ, we denote by $x_R(\Gamma)$ ($x_L(\Gamma)$) the rightmost (leftmost) x-coordinate of $\Gamma(G)$ and by $y(\Gamma)$ the maximum y-coordinate of Γ. Let $G' \subseteq G$ and let $\Gamma' \subseteq \Gamma$ be the visibility representation of G'. The *right* of Γ' is the subdrawing of Γ composed by Γ' and the vertex-segments of Γ that are entirely drawn in the half-plane $(x_L(\Gamma'), \infty)$.

3 1-Visibility Representations

In this section we study the area required by 1-visibility representations of trees. We consider rooted trees separately from free trees.

3.1 Rooted Trees

In this section, we study upward 1-visibility representations of rooted trees.

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

Proof: Let Γ be an upward 1-visibility representation of T. To avoid unwanted visibilities, no two leaves of T can be represented by vertex segments in the same column of the grid. Thus, the width of Γ is at least $l - 1$. Also, each vertex segment of Γ must be placed above its children. Thus the height of Γ is at least h. □

The following algorithm constructs a 1-visibility representation of a rooted tree.

Algorithm 1-ROOTED

Input: Rooted tree T with height h and l leaves.

Output: An upward 1-visibility representation of T with area $O(h \cdot l)$.

1. Order arbitrarily the children of each node, and label the leaves of T as $v_1, \cdots, v_l$, from left to right.
2. Map each leaf v_i to vertex segment $S(v_i) = ([2i, 2i+1], 0)$.
3. For each internal vertex v, let T_v be the subtree rooted at v, $h(v)$ be the height of $T(v)$, and v_i and v_j be the leftmost and rightmost leaves in T_v. Map vertex v to vertex segment $S(v) = ([2i, 2j+1], 0]$.

end Algorithm.

In Figure 3 we give an example of the upward 1-visibility representation constructed by Algorithm 1-ROOTED.

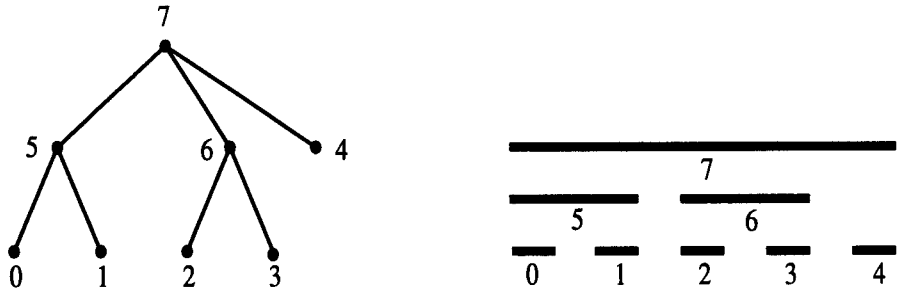

Figure 3: Example of the upward 1-visibility representation constructed by Algorithm 1-ROOTED.

Lemma 2 *Let T be a rooted tree with n vertices, l leaves, and height h. Algorithm* 1-ROOTED *constructs in $O(n)$ time an upward 1-visibility representation Γ of T with area $(2l-1) \cdot h$, i.e., the representation has width $2l-1$ and height h.*

Proof: The bounds on the width, height, and time complexity are trivial. We show that Γ is a 1-visibility representation of T. No two vertex segments of Γ overlap. For each internal vertex v of T the leftmost (rightmost) x-coordinate of $S(v)$ in Γ is the leftmost (rightmost) x-coordinate of the vertex segment associated with the leftmost (rightmost) child of v. The y-coordinate of $S(v)$ is

greater than the maximum y-coordinate of its children. Thus $S(v)$ and the vertex segments associated with the children of v are visible. Since the subtrees rooted at the children of v are entirely drawn in the vertical strips below the vertex segments associated with their roots, no other descendants of v are visible from $S(v)$. Thus, algorithm 1-ROOTED produces an upward 1-visibility representation correctly. □

Combining the results of Lemmas 1 and 2, we obtain the following theorem:

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

3.2 Free Trees

In this subsection we study the area required by 1-visibility representations of free trees.

3.2.1 Area Lower Bound

We observe that for each internal vertex v of a free tree, we can choose two subtrees of v that can be drawn to the right and to the left of $S(v)$, respectively. This leads us to the following definitions.

Let T be a free tree, and let v be a vertex of T. A *subtree* of v in T is a subtree of the rooted tree obtained by rooting T at v. Let $T_1, T_2, \cdots, T_d$ be the subtrees of v sorted by nonincreasing height (i.e., $h(T_1) \geq h(T_2) \geq \cdots \geq h(T_d)$). We call $h(T_k)$ the *k-th height* of v. A subtree of v with height equal to the k-th height of v is called a *k-tallest subtree* of v. If T has degree at least 3, we call *critical height* of T the maximum third-height of any vertex of T, and *critical vertex* of T a vertex with maximum third-height. In the example of Figure 4, each degree-3 vertex is labeled with the heights of its subtrees, and the unique critical vertex is shown.

The following lemma gives a lower bound on the area.

Lemma 3 *Let T be a free tree with n vertices and l leaves, and let h^* be the critical height of T. The area required by a 1-visibility representation of T is $\Omega(n + l \cdot h^*)$.*

Proof: Let Γ be a 1-visibility representation of T. Each vertex-segment of Γ uses at least one grid point. Thus, the area of Γ is at least n. Since the vertex-segment associated with a leaf of T can see only one other vertex-segment, a grid column can contain at most two vertex-segments, and hence the width of Γ is at least $\lceil \frac{l}{2} \rceil$. Also, for each vertex v, all but two subtrees of v must be drawn upward in the vertical strip between the endpoints of $S(v)$. It follows that the height of Γ is at least h^*. □

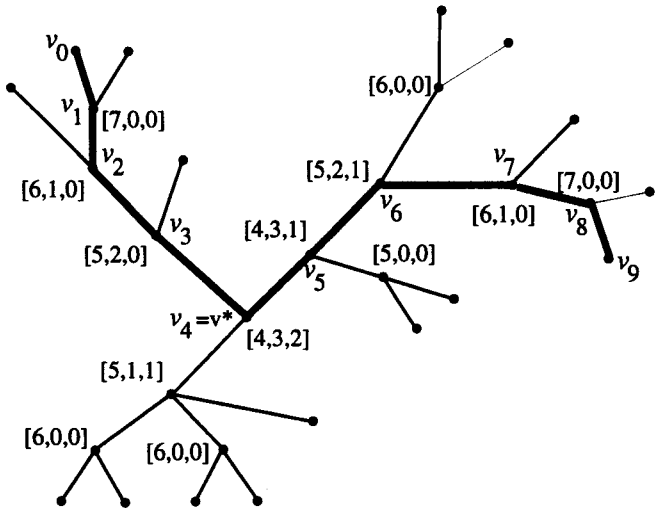

Figure 4: Example of free tree with critical height $h^* = 3$. A longest path is drawn with thick lines. Each internal vertex is labeled with the heights of its subtrees, and the unique critical vertex v^* is shown.

3.2.2 Critical Vertices and Longest Paths

The following lemma shows how critical vertices and longest paths are related. Its proof is illustrated in Figure 5.

Lemma 4 *Let P be a longest path of free tree T, and v^* a critical vertex of T. Then v^* is on P.*

Proof: Assume, for a contradiction, that v^* is not in P. Then, P must be entirely contained in a subtree of v^*. We claim that any third-tallest subtree T^* of v^* does not contain path P.

Proof of the Claim: Assume, for a contradiction, that T^* contains P. Let w be the vertex of P closest to v^*, and let Q be the path of T joining v^* to w. Vertex w splits P into subpaths P' and P'' , where $\ell(P') \geq \ell(P'')$ and $\ell(P') \geq \ell(P)/2$. (We recall that $\ell(Z)$ denotes the length of a path Z).) Let R be a longest path from v^* to a leaf. Since the first-tallest subtree of v^* does not contain P, we have that $\ell(R) \geq \ell(Q) + \ell(P')$. Hence, the path obtained by the concatenation of P', Q, and R has length at least $2(\ell(Q) + \ell(P')) > \ell(P)$, which contradicts the hypothesis that P is a longest path. This concludes the proof of the claim.

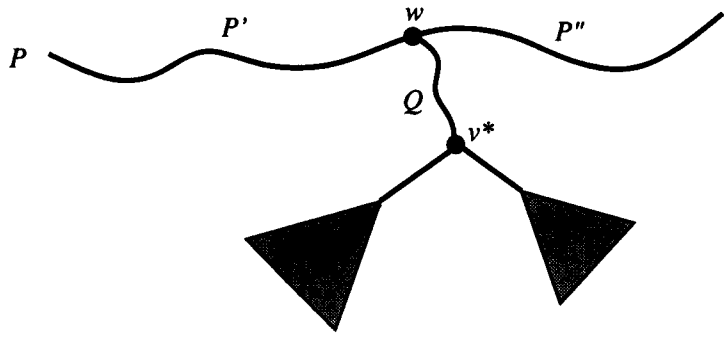

Figure 5: Illustration of the proof of Lemma 4.

As in the above proof of the claim, we let w be the vertex of P closest to v^*, and Q be the path of T joining v^* to w. Since P is a longest path, the two subtrees of w containing a subpath of P are each a first- or a second-tallest subtree. Hence, a third-tallest subtree of w has height at least equal to $\ell(Q) + h^* > h^*$, which contradicts the fact that v^* is a critical vertex. □

Let P be a longest path of free tree T, and v a vertex of P. We denote with T_v the tree rooted at v obtained by deleting from T the subtrees of v rooted at the neighbors of v in P.

Lemma 5 *Let P be a longest path of free tree T with critical height h^*, and v a vertex of P. Then $h(T_v) \leq h^* \dot{+} 1$.*

Proof: Since P is a longest path, the two subtrees of v containing a subpath of P are a first- and second-tallest ones. □

Lemma 6 *A longest path P in a free tree T with n vertices can be computed in $O(n)$ time.*

Proof: This result is folklore. We sketch the algorithm for the sake of completeness. First we find the center of tree T by iteratively removing the leaves of the tree, one level at a time. At the end we are left with either one node (the center of T) or two nodes joined by an edge (a double center of T). Next we root T at the single center (or one vertex of the double center) of T, call it r, and find the first- and second-tallest subtrees. A longest path P in free tree T can be obtained by concatenating a longest path in the first-tallest subtree followed by r followed by a longest path in the second-tallest subtree. Clearly the above steps can be computed in $O(n)$ time.

□

3.2.3 Drawing Algorithm

The algorithm for constructing a 1-visibility representation of a free tree T consists of two phases. In the first phase we find a longest path P of T and draw it. In the second phase, for each vertex v of P we draw T_v using Algorithm 1-ROOTED.

First, we show how to draw trees when all the internal (i.e., nonleaf) vertices of T have degree at least 3. Later we extend the drawing algorithm to work for any tree.

Algorithm DRAW-TREE

Input: Free tree T with n vertices, l leaves, and critical height h^*, such that all the internal vertices of T have degree at least 3.

Output: 1-visibility representation Γ of T with area $O(l \cdot h^*)$.

1. Compute a longest path $P = (v_0, v_1, \cdots, v_{m+1})$ of T.
2. For each $i = 1, \cdots, m$, apply Algorithm 1-ROOTED to T_{v_i}, which yields an upward 1-visibility representation of T_{v_i}. If i is odd, extend $S(v_i)$ two units to the left and two units to the right. If i is even and $S(v_i)$ has length ≤ 2, extend $S(v_i)$ one unit to the left and one unit to the right. Let Γ_i be the resulting 1-visibility representation of T_{v_i}.
3. Let $S(v_0) = ([0, 2], 0)$.
4. For $i = 1, \cdots, m$, translate Γ_i such that $x_L(S(v_i)) = x_R(S(v_{i-1})) - 1$ and $y(S(v_i)) = i \bmod 2$.
5. Let $S(v_{m+1}) = (x_R(S(v_m)) - 1, [x_R(S(v_m)) + 2], (m + 1) \bmod 2)$.

End Algorithm

In Figure 6 we show the 1-visibility representation constructed by Algorithm DRAW-TREE for the tree of Figure 4.

Lemma 7 *Let T be a free tree with n vertices, l leaves, and critical height h^*, such that all the internal vertices of T have degree at least 3. Algorithm* DRAW-TREE *constructs in $O(n)$ time a 1-visibility representation Γ of T with area $O(l \cdot h^*)$, width at most $3 \cdot l$, and height at most $h^* + 2$.*

Proof: The correctness is established by observing that two vertex-segments of Γ are visible if and only if its associated vertices are adjacent in T.

By Lemma 2, the height of Γ_i is equal to $h(T_{v_i})$, and by Lemma 5, $h(T_{v_i}) \leq h^* + 1$. Hence, the height of Γ is at most $h^* + 2$.

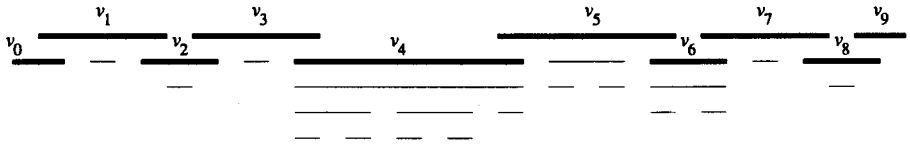

Figure 6: 1-visibility representation constructed by Algorithm DRAW-TREE for the tree of Figure 4. The vertex-segments associated with vertices of the longest path are drawn with thick lines.

By Lemma 2,the width of Γ_i is $2 \cdot l_i + 3$ if i is odd, and at most $2 \cdot l_i + 1$ if i is even, where l_i is the number of leaves of T_{v_i} (recall that segment $S(v_i)$ may be extended in Step 2) . Hence, since Γ_i and Γ_{i+1} have overlap in a vertical strip of unit width, each leaf contributes to at most three units of width.

The time-complexity bound follows from Lemmas 2 and 6. □

Suppose now T has vertices of degree 2. The following algorithm constructs a linear-area drawing of a path with a prescribed height, and will be used as a subroutine.

Algorithm DRAW-PATH

Input: Path P with n vertices. Integer parameter $k \leq n$.

Output: 1-visibility representation of P with height k and width $2 \cdot \lceil \frac{n}{k} \rceil + 1$.

1. Partition P into subpaths $P_1, \cdots, P_s$, where P_s has at most k vertices, and the remaining subpaths have exactly k vertices ($s = \lceil \frac{n}{k} \rceil$).

2. For each subpath P_i, draw P_i as a stack of left-aligned horizontal segments, vertically spaced by one unit. The last segment has length three, and the remaining segments have length one. The stack extends downward or upward according to whether i is odd or even.

3. Combine the drawings of the subpaths as follows. Let v_i and w_i be the first and last vertex of P_i, respectively. For $1 < i \leq s$, we have that $x_L(v_{i+1}) = x_R(w_i) - 1$ and $|y(v_{i+1}) - y(w_i)| = 1$. Also segment $S(v_{i+1})$ is below or above $S(w_i)$ according to whether i is odd or even.

End Algorithm

An example of 1-visibility representation produced by Algorithm DRAW PATH is given in Figure 7.

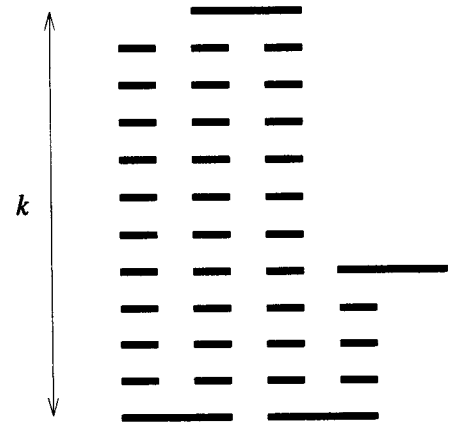

Figure 7: Example of 1-visibility representation of a path produced by Algorithm DRAW-PATH.

Lemma 8 *Let P be a path with n vertices, and let k be a positive integer parameter with $k \leq n$. Procedure* DRAW PATH *constructs in $O(n)$ time an $O(n)$-area 1-visibility representation $\Gamma(P)$ of P with height k and width $2 \cdot \lceil \frac{n}{k} \rceil + 1$.*

Proof: The time complexity bound is immediate. Each vertex segment in $\Gamma(P)$ does not overlap any other vertex segment. Also, a vertex segment $S(v)$ is visible only by the vertex segments representing the predecessor and the successor of v in P. The stack constructed for each subpath of P in Step 2 has width 3 and height $k-1$. In Step 3, we form $\Gamma(P)$ by combining the stacks. We use one extra grid row to make visible the first and last segment of two consecutive stacks, and we add two new grid columns each time we add a new stack. Since there are $\lceil \frac{n}{k} \rceil$ stacks, we conclude that $\Gamma(P)$ has height k and width $2 \cdot \lceil \frac{n}{k} \rceil + 1$. □

We are now ready to describe Algorithm 1-FREE-TREE for general free trees. The basic idea of Algorithm 1-FREE-TREE is to draw first tree T as if no vertices of degree 2 were in the longest path P. Then, we apply Procedure DRAW-PATH to draw each maximal subpath of P consisting of degree-two vertices.

Algorithm 1-FREE TREE

Input: Free tree T with n vertices, l leaves, and critical height h^*.

Output: 1-visibility representation of T with area $O(h^* \cdot l + n)$.

1. Compute a longest path P of T.

2. For each maximal subpath Q of P consisting of degree-two vertices, contract Q into a single edge e_Q. Let T' and P' be the resulting modified tree and path.

3. Apply Algorithm DRAW-TREE to compute a 1-visibility representation Γ' of T', where we use path P' instead of the longest path of T'. Note that each edge e_Q of P' associated with a subpath Q of P consisting of degree-two vertices is mapped to a unit square s_Q in Γ'.

4. For each maximal subpath Q of P, construct a 1-visibility representation $\Gamma(Q)$ of Q by applying Algorithm DRAW-PATH with parameter h^*, and replace s_Q with $\Gamma(Q)$, as shown in Figure 8.

End Algorithm

Lemma 9 *Let T be a free tree with n vertices, l leaves and critical height h^*. Algorithm* 1-FREE-TREE *constructs in $O(n)$ time a 1-visibility representation of T with area $O(n + l \cdot h^*)$.*

Proof: By Lemma 7, the 1-visibility representation Γ' of T' constructed in Step 3 has width $O(l)$ and height $O(h^*)$. By Lemma 8, each 1-visibility representation $\Gamma(Q)$ constructed in Step 4 has height h^* and area proportional to the number of vertices n_Q of Q. The replacements performed in Step 4 increase the height by at most a constant. Hence, each replacement increases the area by $O(n_Q)$. We conclude that the area of the 1-visibility representation $\Gamma(T)$ is $O(n + l \cdot h^*)$. The time complexity bound follows from Lemmas 6, 7, and 8. □

By combining Lemmas 3 and 9, we can summarize the results of this section in the following theorem.

Theorem 2 *Let T be a free tree with n vertices, l leaves, and critical height h^*. The area required by a 1-visibility representation of T is $\Theta(h^* \cdot l + n)$. Also, a 1-visibility representation of T with $O(n + l \cdot h^*)$ area 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.

Every leaf v of a free tree has only one neighbor u. Hence, only $S(u)$ can intersect one of the strips (above, below, right, or left) defined by $S(v)$. This observation yields the following lemma.

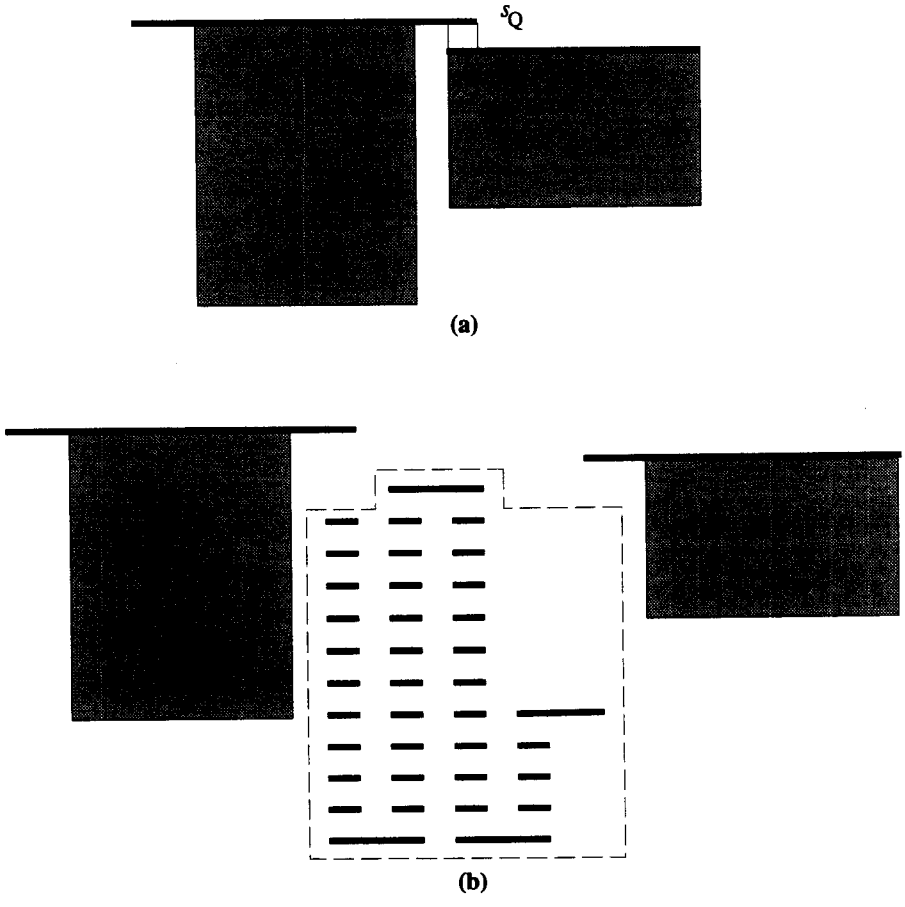

Figure 8: Illustration of Step 4 of Algorithm DRAW-PATH, where a unit rectangle s_Q is replaced with $\Gamma(Q)$, (a) before the replacement; (b) after the replacement.

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

Proof: Each leaf uses at least one grid point. Hence, for every leaf at least one vertical and one horizontal strip are *wasted*, i.e., no rectangle of another vertex can intersect them. Notice that each leaf defines two horizontal (vertical) strips on every row (column) of the grid. Hwcne, the area of the grid containing a 2-visibility representation has to be 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 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 $2(n-1) \geq l + 2k + 3p \geq 2k + 4p + 2 \geq 4p + 2$. This yields $p \leq \frac{2n-2-2}{4} = \frac{n}{2} - 1 \leq \lfloor \frac{n}{2} \rfloor$. Thus $k + l \geq \lceil \frac{n}{2} \rceil$, which completes the proof. □

Next we will describe an algorithm for constructing a 2-visibility representation of a free tree T.

Algorithm 2-FREE TREE

Input: Free tree T with n vertices and l leaves.

Output: 2-visibility representation of T with area $O(n \cdot l)$.

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 i, $1 \leq i \leq l$.
2. Draw each leaf i as 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 $S(v_i) = ([x_L(v_{i_1}), x_R(v_{i_k})], [2i, 2i+1])$.

end Algorithm.

Figure 9 shows an example of a 2-visibility representation obtained by Algorithm 2-FREE TREE.

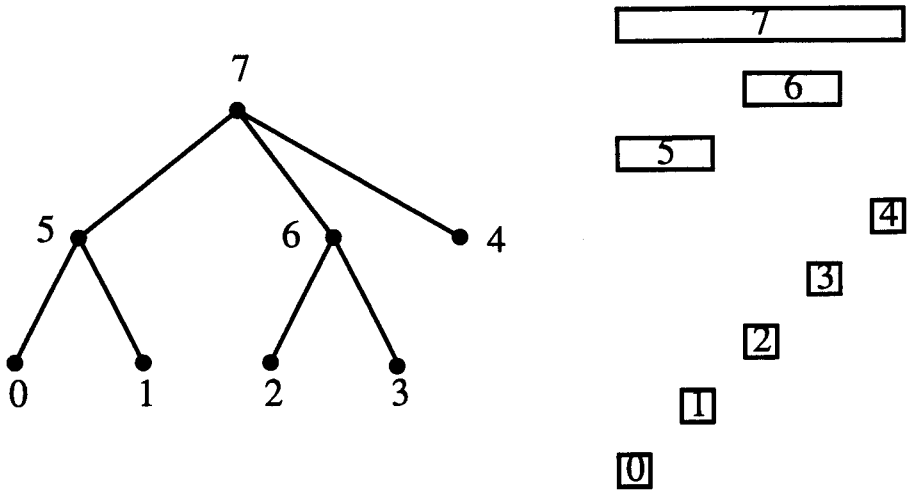

Figure 9: Illustration of Algorithm 2-FREE TREE.

Lemma 12 *Let T be a free tree with n vertices and l leaves. Algorithm* 2-FREE TREE *constructs in $O(n)$ time a 2-visibility representation for T with $O(n \cdot l)$ area.*

Proof: The algorithm can be clearly implemented to run in linear time. It is easy to see that the area of the drawing is $(2l-1) \cdot (2n-1)$. To show that the drawing is indeed a 2-visibility representation, we observe that $S(v_i)$ lies in the horizontal strip between ordinates $2i$ and $2i+1$. Hence no two rectangles can see each other in the horizontal direction, so that visibility is only in the vertical direction. Using an argument similar to the one in the proof of Lemma 2, we conclude that the drawing is a 2-visibility representation. □

By combining Lemmas 11 and 12, we obtain the following theorem.

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

5 Conclusions and Open Problems

In this paper we have studied the area required by 1- and 2-strong visibility representations of trees. We have provided linear-time algorithms for constructing 1- and 2-visibility representation of trees with asymptotically optimal area.

While our results have been proved under the strong visibility assumption, they also hold under the $\epsilon-$ and weak-visibility assumption.

Open problems include:

- Characterize the class of graphs that can be represented by a 2-strong visibility representation. It is not even known whether series-parallel graphs or planar graphs are in this class.
- Find the best possible constant factors in the asymptotic area bounds.
- Characterize the graphs that admit a 2-ϵ visibility representation where no two (visibility) edges cross. We recall that Wismath [28] proved that every planar graph admits a 2-ϵ visibility representation. However, the (visibility) edges in Wismath's construction can cross.

Acknowledgments

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

References

[1] T. Andreae, Some Results on Visibility Graphs, *Discrete Applied Mathematics*, 40, 1992, pp. 5–18.

[2] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications*, Elsevier Science, New York, New York, 1976.

[3] P. Crescenzi, G. Di Battista, and A. Piperno, A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees, *Comp. Geometry: Theory and Applications*, 2, 1993, pp. 187–200.

[4] H. de Fraysseix, J. Pach, and R. Pollack, How to Draw a Planar Graph on a Grid, *Combinatorica*, 10, 1990, pp. 41–51.

[5] H. de Fraysseix, J. Pach, and R. Pollack, Small Sets Supporting Fary Embeddings of Planar Graphs, *Proc. ACM 20th Symposium on Theory of Computing*, 1988, pp. 426–433.

[6] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis Algorithms for Automatic Graph Drawing: An Annotated Bibliography, to appear in *Comp. Geometry: Theory and Applications*, Preliminary version also available via anonymous ftp from `wilma.cs.brown.edu` (128.148.33.66), files `/pub/gdbiblio.tex.Z` and `/pub/gdbiblio.ps.Z`.

[7] G. Di Battista, E. Pietrosanti, R. Tamassia, and I.G. Tollis, Automatic Layout of PERT Diagrams with XPERT, *Proc. IEEE Workshop on Visual languages*, 1989, pp. 171–176.

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

[9] P. Eades, T. Lin, and X. Lin, Minimum Size $h-v$ Drawings, *Advanced Visual interfaces (Proc. of AVI 92)*, World Scientific Series in Comp. Science, 36, 1992, pp. 386–394.

[10] H. Everett, and D. Corneil, Recognizing Visibility Graphs of Spiral Polygons, *Journal of Algorithms*, 11, 1990, pp. 1–26.

[11] H. Everett, A. Lubiw, and J. O'Rourke, Recovery of Convex Hulls from External Visibility Graphs, *Proc. Canadian Conference on Computational Geometry*, 1993, pp. 309–314.

[12] A. Garg, M. T. Goodrich, and R. Tamassia, Area-Efficient Upward Drawings, *Proc. ACM Symp. on Computational Geometry*, 1993, pp. 359-368.

[13] 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.

[14] D.G. Kirkpatrick, and S.K. Wismath, Determining Bar- Representability for Ordered Weighted Graphs, 1993, manuscript.

[15] Leiserson, C.E., Area-efficient graph layouts (for VLSI), *Proc. 21st IEEE Symp. on Foundations of Computer Science*, 1980, 270–281.

[16] J. O'Rourke, *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

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

[18] F. Preparata, and M.I. Shamos, *Computational Geometry: an Introduction*, Springer-Verlag, 1985.

[19] P. Rosenstiehl, and R. E. Tarjan, Rectilinear Planar Layouts and Bipolar Orientations of Planar Graphs, *Discr. and Comp. Geometry*, 1, 1986, pp. 343–353.

[20] M. Schlag, F. Luccio, P. Maestrini, D.T. Lee, and C.K. Wong, A Visibility Problem in VLSI Layout Compaction, *Advances in Computing Research*, ed. F. Preparata, JAI Press Inc., 2, 1985, pp. 259–282.

[21] W. Schnyder, Embedding Planar Graphs on the Grid, *Proc. ACM-SIAM Symp. on Discrete Algorithms*, 1990, pp. 138–148.

[22] T.C. Shermer, Recent Results in Art Galleries, *IEEE Special Issue on Computational Geometry*, 80, 1992, pp. 1384–1399.

[23] J.A. Storer, On Minimal Node–Cost Planar Embeddings, *Networks*, 14, 1984, pp. 181–212.

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

[25] R. Tamassia, and I. G. Tollis, Planar Grid embedding in Linear Time, *IEEE Trans. on Circuits and Systems*, 9, 1989, pp. 1230–1234.

[26] Valiant, L., Universality Considerations in VLSI circuits, *IEEE Transactions on Computers*, vol. c-30, no. 2, 1981.

[27] S.K. Wismath, Characterizing Bar Line-of-Sight Graphs, *Proc. ACM Symp. on Computational Geometry*, 1985, pp. 147–152.

[28] S.K. Wismath, Bar-Representable Visibility Graphs and Related Flow Problems, Ph.D. Thesis, Dept. of Comp. Science, Univ. of British Columbia, 1989.

Area Requirement of Visibility Representations of Trees

G. Kant, G. Liotta, R. Tamassia and I.G. Tollis

RUU-CS-93-33
October 1993

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454