A Linear Algorithm for Compact Box-Drawings of Trees

Masud Hasan[*], Md. Saidur Rahman[†] and Takao Nishizeki[‡]

Abstract

In a box-drawing of a tree each node is drawn by a rectangular box of prescribed size, no two boxes overlap each other, all boxes corresponding to siblings of the tree have the same x-coordinate at their left sides, and a parent node is drawn at a given distance apart from its first child. A box drawing of a tree is compact if it attains the minimum possible rectangular area enclosing the drawing. We give a linear-time algorithm for finding a compact box-drawing of a tree. A known algorithm does not always find a compact box-drawing and takes time $O(n^2)$ if a tree has n nodes.

1 Introduction

Automatic drawings of trees on a two-dimensional plane have many applications in VLSI layouts, information visualization such as displaying tree-structured diagrams, hierarchical file structures etc [DETT99]. The complexity of algorithms for finding a drawing of a tree significantly depends on the aesthetic conditions considered. The problem of finding a drawing of a binary tree on an integer grid of minimum area is NP-hard for a certain set of aesthetic conditions [SR83], whereas for a different set of aesthetic conditions an $O(n\sqrt{n \log n})$ time algorithm is known for finding a drawing of a binary tree on an integer grid of minimum area [ELL92]. Reingold and Tilford gave a linear-time algorithm for tidier drawing of trees with a certain set of aesthetic conditions where the nodes are drawn as points and the drawing does not give optimum compactness [RT81]. In this paper we consider a box-drawing of a tree where each node is drawn by a rectangular box of prescribed size, no two boxes overlap with each other, all boxes corresponding to siblings of the tree have the same x-coordinate at their left sides, and a parent node is drawn at a given distance apart from its first child. The size of a node is described by the width and height of a box representing the node. Fig. 1(a) shows a tree with prescribed width and height for each node; the first integer attached to each node represents the node number, and the two integers written inside brackets are the width and height. Two box-drawings of the tree in Fig. 1(a) are shown in Figs. 1(b) and 1(c), where the y-coordinate of the top side of a parent box is one less than that of its first child if the parent has two or more children, otherwise the top sides of a parent and the child have the same y-coordinate. Moreover, the ordering of the children of each node in Fig. 1(a) is preserved in both of Fig. 1(b) and Fig. 1(c). Box-drawings of trees have practical applications in VLSI layout where a placement is to be found for modules with prescribed width and height [L90, S95].

We call a box-drawing a *compact box-drawing* if the smallest rectangle enclosing the drawing has the minimum possible area. The box-drawing in Fig. 1(c) is a compact box-drawing whereas the box-drawing of Fig. 1(b) is not a compact box-drawing. In this paper we give a linear-time algorithm for finding a compact box-drawing of a tree. The algorithm in [MAUY98] finds a box-drawing of a tree in time $O(n^2)$ if the tree has n nodes, but the found drawing is not always a compact drawing. Walker [W90] gave a linear-time algorithm with a set of aesthetic conditions similar to that of Reingold and Tilford [RT81] where the sizes of all the nodes are variable in only one direction (i.e. either in horizontal or in vertical direction) and fixed in the other direction. A straightforward implementation of the algorithms in [RT81] and [W90] for nodes with variable sizes is not trivial and it would lead to unaesthetic drawings [B93]. Bloesch [B93] gave an algorithm for box-drawing of trees with similar type of aesthetic conditions, but its time complexity is not linear in the number of nodes and it does not always give a compact drawing. In contrast our algorithm is very simple and always gives a compact box-drawing of trees in linear time under the given aesthetic conditions.

The rest of the paper is organized as follows. Section 2 introduces some definitions and known results. Section 3 gives our main result on compact box-drawings of trees. Finally Section 4 is conclusion.

*Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. On leave from Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh. Email: m2hasan@hopper.math.uwaterloo.ca

†Graduate School of Information Sciences, Tohoku University, Aoba-yama 05, Sendai 980-8579, Japan. On leave from Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh. Email: saidur@nishizeki.ecei.tohoku.ac.jp

‡Graduate School of Information Sciences, Tohoku University, Aoba-yama 05, Sendai 980-8579, Japan. Email: nishi@ecei.tohoku.ac.jp

2 Preliminaries

In this section we give some definitions and present some known results.

In this paper a *tree* T means the so-called ordered rooted tree in which there is an ordering of the children of v for each node v in T.

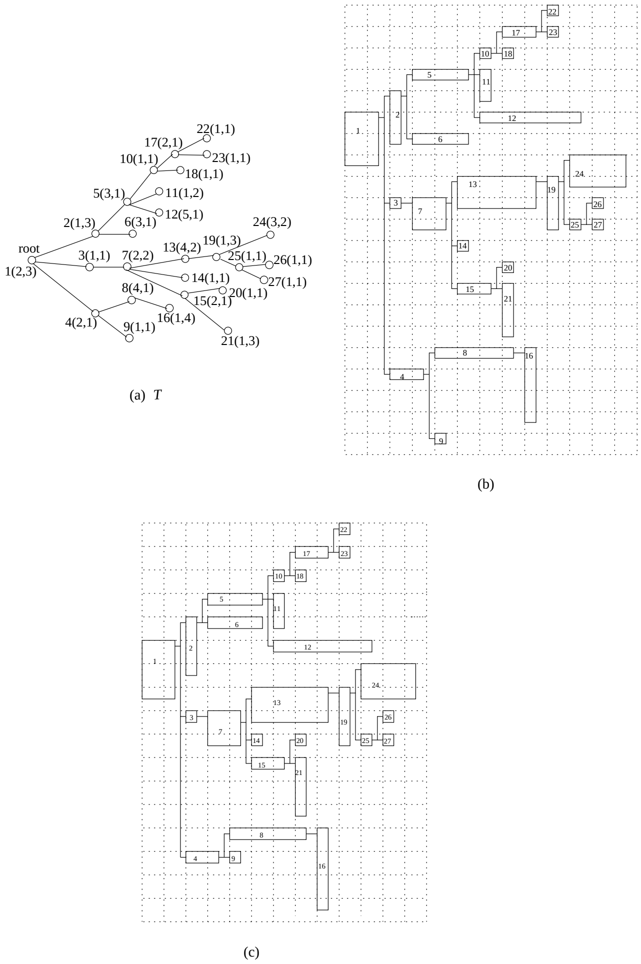

Figure 1: (a) A tree T with nodes of prescribed width and height, (b) an initial box-drawing of T, and (c) a compact box-drawing of T.

We call a drawing of a tree a *box-drawing* if each node of the tree is drawn as an axis-parallel rectangular box and the drawing satisfies the following three aesthetic conditions:

(C1) No two boxes overlap with each other;

(C2) All boxes corresponding to siblings have the same x-coordinate at their left sides, and the x-coordinate is equal to that of their parent plus the width of the parent; and

(C3) A parent node is drawn at a given distance vertically apart from its first child: the y-coordinate of the top side of a parent is smaller than that of its first child by a given integer.

Similarly as in [MAUY98], a box is not drawn in the exact scale of the prescribed size, but the width and the height of a drawn box are less than the prescribed size by a half of one coordinate unit. This amount of gap is used to show the interconnection among a parent and its children. A box is placed in the grid in such a manner that the top-left corner has an integral coordinate value but the bottom-right corner has a half-integral coordinate value. The interconnection among a parent and its children is represented by horizontal and vertical line segments as illustrated in Figs. 1(b) and 1(c). A horizontal line starts from the point which is on the right side of a parent box and is one fourth of one coordinate unit lower from the top-right corner of the box, and ends on a vertical line at the midle of the gap. From the vertical line each child box is connected by a horizontal line segment at one fourth of one coordinate unit lower from the top-left corner of the box.

In this paper we often use the terms *box* and *node* as a synonym. The children of each node in a tree are ordered from top to bottom, and the ordering is preserved in the box-drawing. The *drawing area of a tree* is the minimum rectangle that encloses all boxes of the tree. Similarly, the *drawing area of a subtree* is the minimum rectangle that encloses all boxes of the subtree. The *width of a drawing* is the width of the rectangle, and the *height of a drawing* is the height of the rectangle. Obviously all box-drawings of a tree have the same width because of Condition (C2), and are "optimally compact in its width" i.e., "optimally compact in the x-direction."

Miyadera *et al.* gave an $O(n^2)$ time algorithm for finding a box-drawing of a tree [MAUY98]. Their algorithm works in two phases. In the first phase it finds an initial box-drawing which is not always compact, but satisfies an additional constraint that the drawing areas of any two subtrees whose roots are siblings do not overlap with each other. In the second phase it tries to compact the box-drawing vertically by relaxing the additional constraint, but it does not always find a compact box-drawing. The first phase takes time $O(n)$, and the second phase takes time $O(n^2)$. The drawing in Fig. 1(b) is an initial box-drawing found by the algorithm. Clearly it is not compact. On the other hand, the box-drawing in Fig. 1(c) found by our algorithm is compact, and does not satisfies the additional constraint: for example, the drawing areas (i.e. minimum rectangles) of the two subtrees rooted at nodes 2 and 3 overlap with each other.

3 Compact Drawing

In this section we give a linear-time algorithm for finding a compact box-drawing of a tree T. It suffices to give a linear-time algorithm for finding a compact box-drawing from an arbitrary box-drawing Γ of T, for example, the initial box-drawing which can be found in linear time by the first phase of the algorithm in [MAUY98].

Any box-drawing is compact in the x-direction. Therefore, our task is to compact the box-drawing Γ in the y-direction. Our idea is as follows. Let l be the number of leaves of tree T, and let $p_1, p_2, \cdots, p_l$ be the leaves of T from top to bottom in a tree-traversal order. For the tree T in Fig. 1(a), $l = 14$ and all the leaves are ordered as follows: nodes 22, 23, 18, 11, 12, 6, 24, 26, 27, 14, 20, 21, 16 and 9. We decompose tree T into l node-disjoint paths $P_1, P_2, \cdots, P_l$; P_1 starts at the root and ends at leaf p_1; and each path P_i, $2 \leq i \leq l$, starts at the furthest ancestor of p_i not in any of paths $P_1, P_2, \cdots, P_{i-1}$, and ends at leaf p_i. In Fig. 1, path P_1 contains boxes 1, 2, 5, 10, 17, and 22; paths P_2, P_3, P_4, P_5, and P_6 contain boxes 23, 18, 11, 12, 6, respectively; and path P_7 contains boxes 3, 7, 13, 19, and 24. For each $i, 1 \leq i \leq l$, we denote by T_i the subtree of T induced by $P_1, P_2, \cdots, P_i$. The relative positions of all boxes on each path P_i are fixed in all box-drawings of T, due to Conditions (C2) and (C3). Therefore, the positions of all boxes on P_1 in Γ need not be changed in a compact drawing. We determine the positions of boxes on $P_2, P_3, \cdots, P_l$ in this order. All boxes on P_i, $i \geq 2$, should be moved upward by the same amount of distance for compaction because their relative positions are fixed. The distance is denoted by $dist(P_i)$. In order to compute $dist(P_i)$, one wishes to know the positions of boxes in the subtree T_{i-1} which can be seen from boxes on P_i. We construct a "visibility graph" for this purpose.

Two boxes u and v in Γ are said *visible* from each other if a vertical line segment can be drawn so that it connects u and v but it does not intersect or touch any other box in Γ. The *upward visibility graph* G_{uv} of Γ is defined as follows: each box b in Γ corresponds to a vertex v_b in G_{uv}; and there is a directed edge connecting vertex v_b to vertex $v_{b'}$ in G_{uv} if and only if in Γ box b' is visible from box b and the y-coordinate of the top side of b' is greater than that of b. The upward visibility graph G_{uv} for the box-drawing in Fig. 2(a) is shown in Fig. 2(b). Clearly the upward visibility graph of any box-drawing of a tree is planar. We now have the following lemma.

Lemma 3.1 *The upward visibility graph G_{uv} of a box-drawing Γ of a tree T can be constructed in linear time.*

Proof: We give an algorithm to construct G_{uv} in linear time as follows.

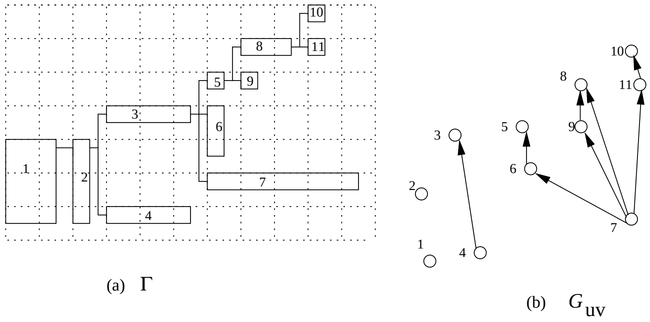

Figure 2: (a) A box-drawing Γ of a tree and (b) its upward visibility graph G_{uv}.

We traverse the paths $P_1, P_2, \cdots, P_l$ in this order and construct the visibility graph G_{uv}. For each $i, 1 \leq i \leq l$, we maintain a list of all boxes in T_i that can see $-\infty$ in the y-direction; the list contains boxes in the increasing order of x-coordinates of their left sides; the list is called the *lower envelope* $LE(T_i)$ of T_i. An entry in the lower envelope is not always a "full" box, but may be a part of a box which can see $-\infty$ in the y-direction. Clearly the visibility graph of a drawing of the subtree T_1 consists of isolated vertices corresponding to the boxes on P_1, and $LE(T_1)$ consists of all (full) boxes on P_1.

Assume that we have constructed the visibility graph of a drawing of the subtree T_{i-1}, $i \geq 2$, and found the lower envelope $LE(T_{i-1})$. We now find the edges of G_{uv} going out from boxes on P_i. Let u be the starting box of P_i, and let v be the sibling of u immediately above u. Clearly v is in $LE(T_{i-1})$. We find all edges of G_{uv} going out from boxes on P_i by comparing the x-coordinates of top-left and top-right corners of boxes on P_i with those of (full or partial) boxes in $LE(T_{i-1})$ starting from v in a manner of comparing two sorted lists for merging. We add the found edges to the visibility graph for T_{i-1} to obtain the visibility graph for T_i. Simultaneously, we update the lower envelope $LE(T_{i-1})$ to $LE(T_i)$; all (full) boxes on P_i appear in the lower envelope $LE(T_i)$, and several consecutive boxes starting from v in $LE(T_{i-1})$ and possibly a part of the last one may be hidden by boxes on P_i and disappear in $LE(T_i)$.

All the boxes in $LE(T_{i-1})$ that are compared once disappear in $LE(T_i)$ except the last compared box, a part of which may remain in $LE(T_i)$. Therefore, the number of comparisons required throughout the execution of the algorithm is $O(\sum_{i=1}^{l}(|V(P_i)|+1)) = O(n+l) = O(n)$, where $|V(P_i)|$ is the number of nodes in P_i. Thus we can construct the visibility graph G_{uv} in time $O(n)$. $\Box$

Theorem 3.1 *A compact box-drawing of a tree can be obtained in linear time.*

Proof: The box-drawing Γ of tree T is optimally compact in the x-direction. Our task is to compact Γ in the y-direction. For this we give a linear-time algorithm as follows.

We first find paths $P_1, P_2, \cdots, P_l$. The positions of all boxes on P_1 in Γ need not be changed in a compact drawing. We now assume that the positions of all boxes in T_{i-1}, $i \geq 2$, have already been determined for a compact drawing. We now determine the positions of all boxes on P_i.

Let u be a box on P_i, let k be the out-degree of u in G_{uv}, and let $v_1, v_2, \cdots, v_k$ be the head nodes of edges going out from node u in G_{uv}. Let $dist(u, v_i)$ be the vertical distance from the top side of box u to the bottom side of box v_i, $1 \leq i \leq k$. Then the possible upward movement $um(u)$ of box u such that u will not overlap any box in T_{i-1} is $\min\{dist(u, v_1), dist(u, v_2), \cdots, dist(u, v_k)\}$. The distance $dist(P_i)$ of the possible upward movement of boxes on P_i is equal to the minimum of possible upward movements of all boxes u on P_i: $dist(P_i) = min\{um(u)| \ u \text{ is a node on } P_i\}$. We uniformly move upward all boxes on P_i by the distance $dist(P_i)$.

By Lemma 3.1 the visibility graph G_{uv} can be constructed in time $O(n)$. Using G_{uv}, we can compute the possible upward movement $um(u)$ of a box u on P_i in time proportional to the out-degree of u in G_{uv}. Since G_{uv} is planar, the sum of out-degrees of all vertices in G_{uv} is at most $3n$. Therefore, one can compute possible upward movements of all boxes in Γ in time $O(n)$. One can compute $dist(P_i)$ from the possible upward movements of all boxes on P_i in time $O(|V(P_i)|)$, and hence one can compute $dist(P_i)$ for $i = 1, 2, \cdots, l$ in time $O(\sum_{i=1}^{l}(|V(P_i)|) = O(n)$. Thus the algorithm takes time $O(n)$.

Clearly the above algorithm obtains a compact box-drawing of T. □

4 Conclusion

In this paper we give a linear-time algorithm for finding a compact box-drawing of a tree following some aesthetic conditions. The best known previous algorithm under the same set of aesthetic conditions runs in $O(n^2)$ time [MAUY98], but does not always find a compact box-drawing.

References

[B93] A. Bloesch, *Aesthetic layout of generalized trees*, Software-Practice and Experience, 23(1993), pp. 817-827.

[DETT99] G. De Battista, P. Eades, and R. Tamassia, *Graph Drawing*, Prentice-Hall, New Jersey, 1999.

[ELL92] P. Eades, T. Lin, and X. Lin, *Minimum size h-v drawing*, Proc. Advanced Visual Interface, World Scientific Series in Computer Science, 36(1992), pp. 386-394.

[L90] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout,* John Wiley & Sons, Chichester, and B. G. Teubner, Stuttgart, 1990.

[MAUY98] Y. Miyadera, K. Anzai, H. Unno, and T. Yaku, *Depth first layout algorithm for trees*, Information Processing Letters, 66(1998), pp. 187-194.

[RT81] E. M. Reingold and J. S. Tilford, *Tidier drawing of trees*, IEEE Transaction on Software Engineering, 7(1981), pp. 223-228.

[S95] N. Sherwani, *Algorithm for VLSI Physical Design Automation,* Kluwer Academic Publishers, Boston, 1995.

[SR83] K. J. Supowit and E.M. Reingold, *The complexity of drawing trees nicely*, Acta-Inform., 18(1983), pp. 377-392.

[W90] J. Q. Walker II, *A node-positioning algorithm for general trees*, Software-Practice and Experience, 20(1990), pp. 685-705.