Tochtermann, Maurer (Eds.): Proceedings of the I-KNOW '07, Graz
7th International Conference on Knowledge Management
Journal of Universal Computer Science, pp. 408-415, ISSN 0948-695x

On the Generalized Box-Drawing of Trees
—Survey and New Technology—

Benno Stein
(Faculty of Media / Media Systems
Bauhaus University Weimar, Germany
benno.stein@medien.uni-weimar.de)

Frank Benteler
(Computer Science Department
University of Paderborn
33095 Paderborn, Germany)

Abstract: This paper is on the aesthetic layout of n-ary trees with nodes of *variable size*, also referred to as generalized box-drawing. For this layout problem a few algorithms have been proposed, which differ in their runtime performance and the attained aesthetic criteria.

We introduce a new approach to this layout problem, which is interesting because of both its simplicity and elegance, and which employs the following piggy-back metaphor: The original box-drawing problem, Π, is topologically reformulated as a layout problem with *uniform node size*, resulting in a new drawing problem Π', which then is handled by the best tree layout algorithm for uniform node size developed so far. The reformulation step can be done in linear time in the number of nodes of the original tree, resulting in an overall linear time complexity. Compared to the existing approaches our approach fulfills more aesthetic criteria; experiments have also shown its efficiency in complex layout settings.

Key Words: tree layout, generalized box-drawing, topology reformulation, meta drawing
Category: H.5, I.3.5

1 Introduction

The drawing of rooted, ordered trees of unbounded degree is considered a solved problem in the field of graph drawing—as long as the nodes in the trees are restricted to uniform size. A generalization of this layout problem is the so-called box-drawing, where each node represents a rectangular box of predefined size. In the relevant literature, in particular in [Cruz and Tamassia 1988; DiBattista et al. 1994, 1999], less attention is paid to this layout problem though various applications exist that require a solution of a certain instance of this drawing task:

- Presentation of linked information items and networks [Eklund et al. 2002].
- Browsing and visualizing hierarchical relations of the World Wide Web.
- Visualization of hierarchies in object-oriented systems [Girba and Lanza 2004].
- Drawing of structured flowcharts for diagram generation [Ogura et al. 1992].
- Drawing of proof trees for the analysis of logical formulas [Bajaj et al. 2003].
- VLSI layout, in particular the placement of modules [Lengauer 1990].

Before we delve into details of the layout problem, Definition 1 will recapitulate the terminology used in the paper.

Definition 1 (Terminology). A rooted tree, $T = \langle V, E \rangle$, is a connected, acyclic graph with node set V and edge set E, having a designated node $r \in V$, called root. Each node v has a unique path $p_r(v)$ to the root; the path length, $|p_r(v)|$, is called the level of v. Given an edge $\{v, w\} \in E$, v is called parent

node of w and w is called child node of v if $|p_r(v)| < |p_r(w)|$. $\beta : V \to \mathbf{N} \times \mathbf{N}$ is a mapping that assigns each node v a bounding box of predefined width and height.
A drawing is a mapping $\delta : V \to \mathbf{N} \times \mathbf{N}$ that assigns each node v a position; a boolean layout constraint γ is a two-valued mapping, $(T, \beta, \delta) \mapsto \gamma(T, \beta, \delta) \in \{0, 1\}$, indicating whether or not the constraint is fulfilled for δ [see 1]. A tree T along with an assignment of boxes, β, and a set of layout constraints, Γ, forms a generalized box-drawing problem Π. A drawing δ is called a valid layout for Π if all layout constraints $\gamma \in \Gamma$ are fulfilled.

Given a generalized box-drawing problem Π, let $bb_{\max}$ be a bounding box whose width (height) is the supremum of the widths (heights) of all bounding boxes β. A common ad-hoc solution to Π is the application of a standard tree layout algorithm that presumes uniform node size—either with or without scaling the uniform node size in order to provide sufficient room for $bb_{\max}$. Another technique is slicing, which organizes the bounding boxes in vertical or horizontal lanes. Such approaches are easy to implement but aesthetically unsatisfactorily, as Figure 1 shows.

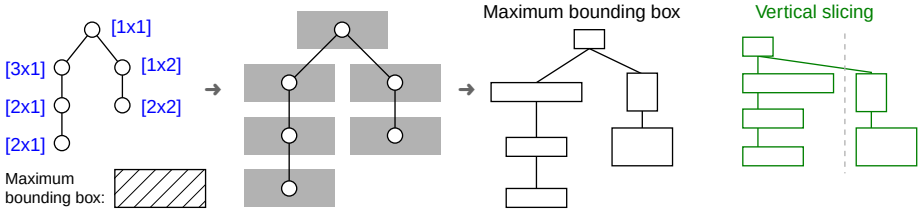

Figure 1: Ad-hoc solutions to Π. The use of standard tree layout algorithms for the generalized box-drawing problem yields unacceptable drawings in most cases.

The complexity of an algorithm for drawing a tree depends on the aesthetic criteria to be fulfilled. In fact, the problem of computing a drawing of a binary tree on an integer grid of minimum area becomes NP-hard under reasonable layout constraints [Supowit and Reingold 1983]. However, given a different set of aesthetic criteria, an $O(n\sqrt{n})$ runtime algorithm for finding a minimum area drawing can be stated [Eades 1992]. The famous algorithm of Reingold and Tilford runs in linear time, and, though the resulting drawing does not give minimum compactness, the entirety of the fulfilled layout constraints results in an aesthetic drawing. These considerations show that there is considerable room to develop new variants of algorithms for the generalized box-drawing of trees, each fulfilling a different set of aesthetic constraints or being bound by a different runtime complexity.

Both the development of a special purpose algorithm and the ad-hoc application of a standard layout algorithm characterize extreme points. In this paper we propose a third possibility: The reformulation of the original box-drawing problem Π as a layout problem of uniform node size, which then is handled by a standard tree layout algorithm for uniform node size. This possibility exploits the developed algorithmic know-how as well as the accepted practice of aesthetic criteria.

[1] Section 2 organizes the well-established boolean layout constraints.

	Approach	Runtime	Space	Layout constraints
Uniform-sized nodes	Wetherell and Shannon 1979	$O(n)$		1, 2, 5, 7
	Reingold and Tilford 1981	$O(n)$		1, 2, 3, 4, 5, 6, 7, 8
	Walker 1990	$O(n^2)$		1, 3, 4, 5, 6, 7, 8, 9, 10
	Buchheim et al. 2002	$O(n)$		1, 3, 4, 5, 6, 7, 8, 9, 10
Variable-sized boxes	Vaucher 1980	$O(nh)$	$O(n)$	1, 5, 6, 7, 8
	Bloesch 1993	$O(nh)$	$\Theta(h)$	3, 5, 6, 7, 8, 9, 11, 13, 14
		$O(nh)$	$O(nh)$	3, 4, 5, 6, 7, 8, 9, 11, 13, 14
	Miyadera et al. 1998	$O(n^2)$		5, 6, 7, 11, 13, 15
	Hasan et al. 2002	$O(n)$		5, 6, 7, 11, 13, 15

Table 1: Well-known algorithms for tree layout along with their runtime and the fulfilled layout constraints. n designates the number of nodes; h designates the drawing height of the tree, which is measured in unit bounding boxes for example (cf. Section 3).

The paper is organized as follows: Section 2 provides a unified view on the existing work for tree layout; Section 3 introduces our reformulation approach, presents new analysis results, and discusses implications.

2 Classification of Existing Work

The development of algorithms for the drawing of trees has a long tradition. In 1979 Wetherell and Shannon presented a linear time algorithm for the drawing of binary trees of uniform-sized nodes. An improvement of this algorithm was given by Reingold and Tilford in 1981, whose algorithm draws isomorphic subtrees identically up to translation. 1990 Walker introduced an algorithm for the aesthetic layout of n-ary trees, allegedly running in linear time. 2002 Buchheim et al. demonstrated a runtime complexity of $O(n^2)$ for Walker's algorithm, while improving it towards $O(n)$ at the same time. Other and little known approaches can be found in [Brüggemann-Klein and Wood 1989; Eades 1992; Eades et al. 1992; Gibbons 1996; Kennedy 1996; Radack 1988].

The generalized box-drawing problem has achieved less attention. Vaucher gave a relevant algorithm with a runtime complexity of $O(nh)$, where h denotes the tree's drawing height. His work served as starting point for Bloesch, who developed both a variant with improved space requirements, $O(h)$, and a version that fulfilled all constraints claimed by Reingold and Tilford [Bloesch 1993; Vaucher 1980]. Miyadera et al. presented an $O(n^2)$-runtime algorithm, and Hasan et al. presented the first linear-time algorithm for an acceptable number of layout constraints.

In the following we have organized the most important and commonly referred boolean layout constraints for the drawing of trees. The compilation considers all of the mentioned papers and shall serve as a "unification" of the language usage; moreover, it classifies the constraints with respect to a layout problem class. Table 1 relates the constraints to algorithms while Figure 2 illustrates selected constraints.

Boolean layout constraints for binary trees with uniform-sized nodes:

(1) The vertical coordinate of a node corresponds to its level in the tree [see 2].
(2) A left child is positioned to the left of its parent and a right child to the right.

[2] W.l.o.g. it is assumed that the layout direction from root node to leaf nodes is top down. The constraints in [Hasan et al. 2002] presume a layout direction from left (root) to right (leafs).

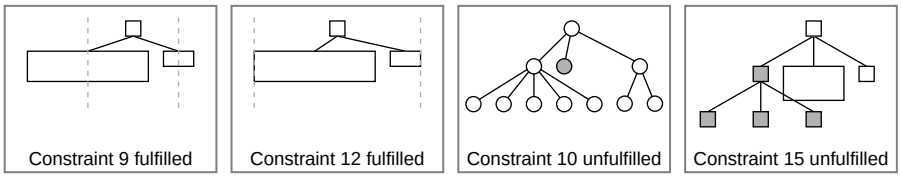

Figure 2: Illustration of selected layout constraints from our compilation.

(3) A parent node is centered over its child nodes.
(4) Nodes at the same level have a minimum horizontal distance.
(5) Edges do not intersect.
(6) Isomorphic subtrees are layouted identically up to translation.
(7) The order of the children of a node is retained in the drawing.
(8) If for each node the order of its children is reversed, the original and the resulting tree produce drawings that are reflections of each other.

Additional boolean layout constraints for n-ary trees with uniform-sized nodes:

(9) A parent node is centered over the center of its leftmost and rightmost children.
(10) Horizontal distances between nodes at the same level have minimum variance while obeying Constraints (8) and (9).

Additional boolean layout constraints for n-ary trees with variable-sized boxes:

(11) The box of each node is separated vertically from its parent by a minimum predefined distance.
(12) A parent node is centered between the left border of its leftmost child and the right border of its rightmost child.
(13) The top borders of the boxes corresponding to a parent's children align horizontally.
(14) Each edge connects the center of the bottom of some parent node with the center of the top of some child node.
(15) Edges and nodes do not intersect.

Remarks. (a) The algorithm of Reingold and Tilford is for binary trees only. (b) Constraint (1) is not reasonable for the generalized box-drawing-problem, and Constraint (11) will serve as an equivalent generalization. (c) Constraint (10) is an improvement introduced by Walker: it makes not sense for the binary tree layout problem and tightens Constraint (4), which states no restriction for the horizontal position of inner subtrees. (d) Together Constraint (4) and (11) guarantee that no two boxes overlap. (e) Constraint (9) implies Constraint (3), and (f) Constraint (15) implies Constraint (5).

3 A Meta Algorithm for the Generalized Box-Drawing Problem

Algorithms for the drawing of trees with uniform-sized nodes have been constantly improved, they are technically mature and efficient. Prevalent principle is to draw a tree recursively in a bottom up sweep, where subtrees emanating from a parent's children are drawn independently and then shifted to the right of the left subtree. Bloesch argues that this principle requires to partition the drawing into horizontal slices, and, that it is difficult to extend the principle toward trees with variable-sized nodes [Bloesch 1993].

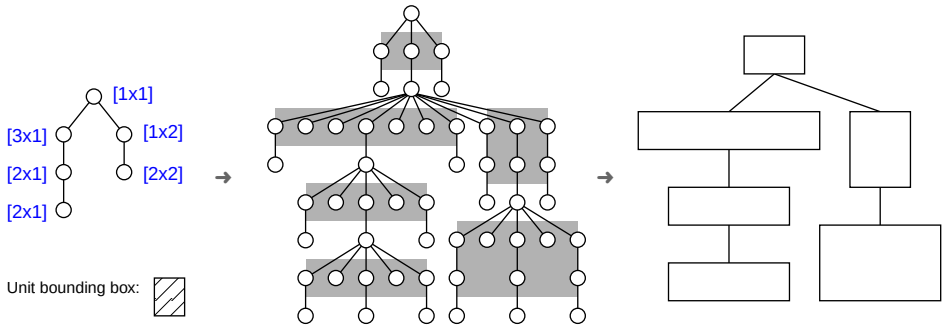

Figure 3: The algorithm illustrated at a small graph and a non-quadratic unit bounding box bb_1.

However, things look different if the information about the bounding boxes, β, are encoded as subtree layout problems with uniform-sized nodes. Then, a standard drawing algorithm can be applied directly.

Let $\Pi = \langle T, \beta, \Gamma \rangle$ and $\Pi' = \langle T', \Gamma' \rangle$ designate the original and the reformulated drawing problem respectively; the transformation $\Pi \longrightarrow \Pi'$ is accomplished with the following meta algorithm (see Figure 3 for a pictorial illustration):

(1) *Definition of Unit Bounding Box.* The nodes in T' are of uniform size, represented by the unit bounding box bb_1. Its actual size is subject to the content of the original boxes and the proportion of β's extreme values. E. g., if the original bounding boxes contain text, bb_1 may not be smaller than the outline of a capital letter.

(2) *Topology Reformulation.* For each bounding box $\beta(v)$ of a node v in T a subtree T_v is constructed. T_v has $w_v + 3 \cdot h_v$ nodes to model the top border of $\beta(v)$, its left and right side as well as a centered bottom node. In particular, $w_v = \lceil width(\beta(v))/width(bb_1) \rceil + 1$ and $h_v = \lceil height(\beta(v))/height(bb_1) \rceil$. T' follows the topology of T: If a node v in T has children $v_1, \ldots, v_k$, the root nodes of the respective subtrees T_{v_i} are unified with the bottom node of T_v.

(3) *Layout.* Application of the fast Walker algorithm [Buchheim et al. 2002] to T'.

(4) *Topology Reverse Mapping.* Substitution of the bounding boxes, β, for the subtrees introduced in Step (2). The top of a bounding box $\beta(v)$, v in T, is centered and aligned with respect to the root node of T_v.

(5) *Rendering.* Rendering of the bounding boxes' contents with a graphics API.

Remarks. (a) Since both the topology reformulation step and the reverse mapping step require constant time for each node, the overall runtime is in $O(n)$. However, the size of bb_1 defines the horizontal and the vertical resolution of the approach, this way determining a multiplicative constant in terms of the runtime complexity. (b) Obviously all layout constraints of Reingold and Tilford as well as of Walker are fulfilled. Moreover, due to construction also all additional layout constraints for variable-sized boxes are fulfilled; this pertains in particular to the constraints (10), (12), and (15), which are not fulfilled by the special purpose algorithms from Bloesch, Miyadera et al., or Hasan et al. (c) Note that the minimum enclosing bounding box of a subtree T_v may be significantly larger than the bounding box $\beta(v)$ of the node v it represents. This is caused by the aesthetic optimization constraint (10) among others and does neither compromise the reverse mapping step nor the fulfillment of some layout constraint.

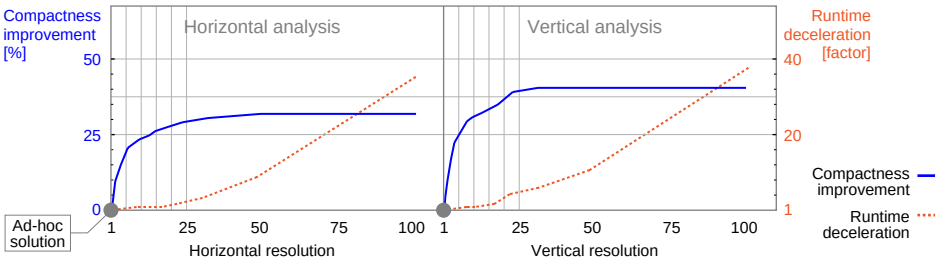

Figure 4: Trade-off between layout compactness and runtime. Due to the nature of the layout algorithm there is less optimization potential in the horizontal direction (solid curve on the left-hand side) than in the vertical direction (solid curve on the right-hand side).

3.1 Qualitative and Quantitative Analysis Results

This subsection presents analysis results to demonstrate the practical applicability of our approach with respect to both layout quality and runtime performance. The determinants of the analyzed graphs are given in Table 2; they resemble problem properties from the applications mentioned at the outset.

number of nodes in the original graph $T = \langle V, E \rangle$	1000 - 30,000
degree of a node $v \in V$	0-10, uniformly distributed
width of the bounding boxes $\beta(v), v \in V$	1-100 units, uniformly distributed
height of the bounding boxes $\beta(v), v \in V$	1-100 units, uniformly distributed

Table 2: Parameters and their distribution of the constructed random graphs.

The following questions shall be answered by the analysis:

(1) How is the resolution, say, the size of bb_1, related to the compactness of the layout?
(2) What is the best trade-off between runtime effort and layout compactness?
(3) Does the nature of the algorithm (bottom up sweep, left-right subtree positioning) prefer horizontal compactness over vertical compactness or vice versa?

Figure 4 compiles several results of our experiments. The curves average over different graphs and show the compactness improvement (solid line) against the bb_1-resolution as well as the runtime deceleration (dashed line) against the bb_1-resolution. The bb_1-resolution defines how many unit bounding boxes fit horizontally (left plot) or vertically (right plot) into a tree's maximum bounding box. Take notice of the different scales: The left scale measures the improvement of the layout compactness as [%]-fraction of the saved place in relation to the ad-hoc solution; the right scale measures the runtime slowdown in multiples of the runtime of the ad-hoc solution. It is very interesting to observe that in fact the layout algorithm of Walker (or Reingold and Tilford) is biased: The horizontal compactness of the ad-hoc solution is significantly closer to the optimum than is the vertical compactness. The results show also that for a substantial compactness improvement the additional runtime effort is pretty small. These insights, which can exactly be quantified, provide valuable information to estimate the size of bb_1 within Step (1) of the meta algorithm.

While Figure 4 is mainly concerned with trade-off analyses, it may also be useful to learn about absolute runtime figures. In this connection the algorithms from Reingold and Tilford as well as from Walker (respectively Buchheim et al.) are of major interest

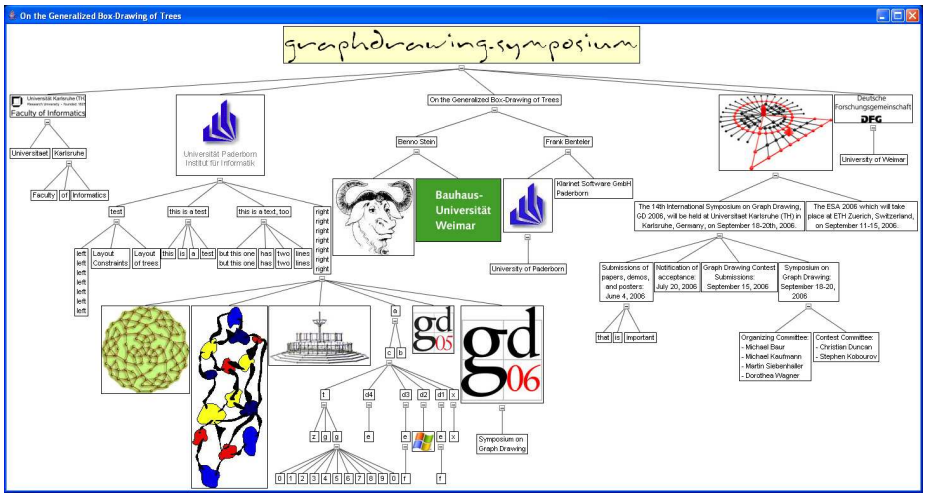

Figure 5: Snapshot of a layout generated with our implementation. Input was a GraphML file that defined a tree with about 100 nodes along with the nodes' contents.

to us. On a standard PC (2Ghz) the layout process of large trees requires approximately 100msec per 10,000 nodes with Reingold and Tilford; Buchheim et al.'s version, which additionally fulfills Constraint (10), is merely less than 1% slower. Underlying is an implementation in Java. As expected, the layout process is not the bottleneck in entire drawing process, which additionally justifies the meta drawing approach.

4 Summary

The paper discussed the generalized box-drawing problem and introduced a scalable solution whose key idea is a topology reformulation in terms of a simplified problem: the layout of trees of uniform-sized nodes. Moreover, we presented a classification of existing tree layout algorithms along with their layout constraints. Our approach fulfills more constraints than the algorithms found in the literature, and qualitative and quantitative analyses show its efficiency in complex layout settings. The advantages of our approach are threefold:

(1) *Conformance.* All well-established layout constraints are fulfilled by virtue of the respective standard algorithm. Moreover, with the topology reformulation powerful extensions of these constraints emerge that are related to the generalized box-drawing problem. Constraint (12) and (15) may serve as examples; their validity can be inferred from the constraints that are fulfilled by the standard algorithm.

(2) *Performance.* If the reformulation step is done in constant time c for each node, the layout step determines the overall runtime, which is linear in the original number of nodes. Our analysis shows that an almost optimum compactness is achieved with pretty small values for c.

(3) *Sustainability.* The approach benefits from improvements in the layout step, be it the fulfillment of additional layout constraints, or a better runtime performance.

The topology reformulation principle is also suited to model new kinds of layout constraints. Examples include the enforcement of bounds for the angle of edges between

parents and children, or bounding-box-dependent distances between a node's parent or siblings. Various tasks require an efficient and high-quality solution of the generalized box-drawing problem. Our application domain is information visualization, and, in particular, the interactive navigation and browsing in large document collections, where performance is of paramount importance.

References

[Bajaj et al. 2003] C. Bajaj, S. Khandelwal, J. Moore, V. Siddavanahalli. Interactive Symbolic Visualization of Semi-automatic Theorem Proving. *IEEE Symp. on Inf. Visualization*, 2003.

[Bloesch 1993] A. Bloesch. Aesthetic Layout of Generalized Trees. *Software Practice and Experience*, 23(8):817–827, 1993.

[Brüggemann-Klein and Wood 1989] A. Brüggemann-Klein and D. Wood. Drawing Trees Nicely with TeX. *Electronic Publishing*, 2(2):101–115, 1989.

[Buchheim et al. 2002] C. Buchheim, M. Jünger, S. Leipert. Improving Walker's Algorithm to Run in Linear Time. *Graph Drawing 2002*, *LNCS* 2528, pp. 344–353. Springer, 2002.

[Cruz and Tamassia 1988] I. F. Cruz and R. Tamassia. Graph Drawing Tutorial, 1988. URL `http://www.cs.brown.edu/people/rt/papers/gd-tutorial/`.

[DiBattista et al. 1994] G. DiBattista, P. Eades, R. Tamassia, I. Tollis. Algorithms for Drawing Graphs: An Annotated Bibliography. *Computational Geometry*, 4, 1994.

[DiBattista et al. 1999] G. DiBattista, P. Eades, R. Tamassia, I. Tollis. *Graph Drawing – Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[Eades 1992] P. Eades. Drawing Free Trees. *Bulletin of the Institute for Combinatorics and its Application*, 5:10–36, 1992.

[Eades et al. 1992] P. Eades, T. Lin, Xuemin. Minimum Size h-v Drawing. *Proc. Advanced Visual Interfaces, AVI'92*, pp. 386–394. World Scientific, 1992.

[Eklund et al. 2002] P. W. Eklund, N. Roberts, S. Green. OntoRama: Browsing RDF Ontologies Using a Hyperbolic-Style Browser. *Proc. 1st Int. Symp. on Cyber Worlds*, pp. 405–411, 2002.

[Gibbons 1996] J. Gibbons. Deriving Tidy Drawings of Trees. *Functional Programming*, 6(3): 535–562, 1996.

[Girba and Lanza 2004] T. Girba and M. Lanza. Visualizing and Characterizing the Evolution of Class Hierarchies. *Proc. 5th ECOOP Workshop on Object-Oriented Reengineering*, 2004.

[Hasan et al. 2002] M. Hasan, M. S. Rahman, T. Nishizeki. A Linear Algorithm For Compact Box-Drawings Of Trees. *CCCG*, pp. 154–157, 2002.

[Kennedy 1996] A. J. Kennedy. Drawing Trees. *Functional Programming*, 6(3):527–534, 1996.

[Lengauer 1990] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley & Sons, New York, 1990.

[Miyadera et al. 1998] Y. Miyadera, K. Anzai, H. Unno, T. Yaku. Depth-First Layout Algorithm for Trees. *Information Processing Letters*, 66(4):187–194, 1998.

[Ogura et al. 1992] K.Ogura, N.Go, M.Kishimoto, Y.Miyadera, Okada, Tsuchida, Unno, Yaku. Generation of the Hichart Program Diagrams. *Information Processing*, 15(2):293–300, 1992.

[Radack 1988] G. M. Radack. Tidy Drawing of M-ary Trees. Report CES-88-24, Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, OH, 1988.

[Reingold and Tilford 1981] E. M. Reingold and J. S. Tilford. Tidier drawing of trees. *IEEE Transactions on Software Engineering*, 7(2):223–228, 1981.

[Supowit and Reingold 1983] K. J. Supowit and E. M. Reingold. The Complexity of Drawing Trees Nicely. *Acta Informatica*, 18:377–392, 1983.

[Vaucher 1980] J. Vaucher. Pretty-Printing of Trees. *Software-Practice and Experience*, 10(7): 553–561, 1980.

[Walker 1990] J. Q. Walker. A Node-positioning Algorithm for General Trees. *Software Practice and Experience*, 20(7):685–705, 1990.

[Wetherell and Shannon 1979] C. Wetherell and A. Shannon. Tidy Drawings of Trees. *IEEE Transactions of Software Engineering*, 5(5):514–520, 1979.