Acta Informatica 18, 377–392 (1983)

Acta Informatica

The Complexity of Drawing Trees Nicely*

Kenneth J. Supowit[1] and Edward M. Reingold

Department of Computer Science University of Illinois at Urbana-Champaign
Urbana, Illinois 61801, USA

Summary. We investigate the complexity of producing aesthetically pleasing drawings of binary trees, drawings that are as narrow as possible. The notion of what is aesthetically pleasing is embodied in several constraints on the placement of nodes, relative to other nodes. Among the results we give are: (1) There is no obvious "principle of optimality" that can be applied, since globally narrow, aesthetic placements of trees may require wider than necessary subtrees. (2) A previously suggested heuristic can produce drawings on n-node trees that are $\Theta(n)$ times as wide as necessary. (3) The problem can be reduced in polynomial time to linear programming; hence, if the coordinates assigned to the nodes are continuous variables, then the problem can be solved in polynomial time. (4) If the placement is restricted to the integral lattice then the problem is NP-hard, as is its approximation to within a factor of about 4 per cent.

From a tree no proof can be brought.
Talmud: Baba Metziah 59b

I. Introduction

Recent papers [7, 10, 9], have studied the problem of producing narrow, well-shaped drawings of tress. The notion of "well-shaped" is incorporated by several aesthetics [7, 10] designed to capture various aspects of shapeliness. The basic task is to assign a pair of coordinates (x, y) to each node of the tree; after such an assignment of coordinates, the tree is easily printed or drawn on some output device. In this paper, we will examine the complexity of determining the assignment of coordinates that gives the narrowest possible drawing while satisfying the aesthetics.

Let T be a (rooted, binary) tree. A *placement* of T is a function

$$\pi\colon \{\text{nodes of } T\} \to R^2.$$

* This research was supported in part by the National Science Foundation, grant numbers NSF MCS 77-22830, NSF MCS 79-04897, and NSF MCS 81-17364

[1] Present address: Hewlett-Packard, Computer Research Lab, Palo Alto, CA94304, USA

0001-5903/83/0018/0377/$03.20

The mapping π specifies, for each node p in T, the coordinates $\pi(p)=(x, y)$ of the point in the Euclidean plane where p is to be placed when the tree is drawn. Drawing a tree T means drawing, for each edge (p, q) in T, a straight line segment joining point $\pi(p)$ to point $\pi(q)$. We will use $\pi_x(p)$ and $\pi_y(p)$ to denote the x and y coordinates of $\pi(p)$, respectively. A placement π of T is *eumorphous* (from the Greek $\varepsilon\upsilon\mu o\rho\phi o\varsigma$ meaning "well-shaped") if it satisfies the following six aesthetic constraints:

Aesthetic 1. For each $i \geqq 0$, there is a straight line ℓ_i such that for each node p which has level i, $\pi(p)$ lies on ℓ_i. (The *level* of a node is the number of branches between it and the root; that is, the root has level 0, and each other node has level one greater than that of its father). Furthermore, the lines ℓ_i are all mutually parallel and evenly spaced. Without loss of generality, we require that the lines ℓ_i are all parallel to the x-axis.

Aesthetic 2. Each right son is placed strictly to the right of its father, and each left son strictly to the left of its father. In particular, for each node p that has a right son,

$$\pi_x(\text{rightson}(p)) - \pi_x(p) \geqq 1,$$

and for each node p that has a left son,

$$\pi_x(p) - \pi_x(\text{leftson}(p)) \geqq 1.$$

Aesthetic 3. For each $i \geqq 0$, for each two nodes p, q having level i, p and q must be placed at least 2 units apart, that is,

$$|\pi_x(p) - \pi_x(q)| \geqq 2.$$

Note that Aesthetic 3 is not implied by Aesthetic 2, since p and q need not have the same father.

Aesthetic 4. Fathers must be centered over their sons. That is, for each node p that has a left son and a right son,

$$\pi_x(\text{rightson}(p)) - \pi_x(p) = \pi_x(p) - \pi_x(\text{leftson}(p)).$$

Aesthetic 5. No two tree edges cross each other when the tree edges are drawn as straight line segments. That is, if two tree edges are drawn so as to intersect, then they share a common endpoint.

Aesthetic 6. If T_1 and T_2 are isomorphic subtrees of T, then π must place T_1 and T_2 identically, up to a translation. We use the term *isomorphic* to denote what Knuth calls *similar* ([5], p. 325): Two binary trees T_1, T_2 are *isomorphic* if either (1) they are both empty, or (2) they are both non-empty and their left and right subtrees are respectively isomorphic. By a *translation*, we mean that there are two real numbers Δ_x and Δ_y such that for each node p_1 of T_1,

$$\pi_x(p_1) = \pi_x(p_2) + \Delta_x,$$

and

$$\pi_y(p_1) = \pi_y(p_2) + \Delta_y,$$

where p_2 is the node of T_2 which corresponds to p_1 under the isomorphism.

Some justification for the first of these aesthetics is given in [10], but they are rather self-evident. We note, however, that in drawing parse trees, one might want all the leaves to lie on one horizontal line; for that application Aesthetic 1 is not desirable. In this case, though, the width (as defined below) of the placement is fixed and so the minimum width placement problem for such parse trees is not interesting. We therefore restrict our attention to the wide class of applications for which Aesthetic 1 is desirable.

Unlike the first five aesthetics, Aesthetic 6 perhaps does not immediately come to mind as a desirable property of tree drawings. However, we have included it for two reasons:

(1) In [7] an example is given of an undesirable placement of a tree by the algorithm of [10]. That placement satisfies Aesthetics 1 through 5, but not Aesthetic 6. The undesirable features of this placement are precluded by Aesthetic 6. (In fact, this is why the authors of [7] proposed Aesthetic 6.)

(2) In some applications, one wishes to examine large trees to find repeated patterns; the search for patterns is facilitated by having isomorphic subtrees drawn isomorphically. For example, Bitner (see [6]) was looking for a way to prune a certain class of backtrack search trees. He discovered a recurrent pattern by visually examining a tree with 348 nodes. Such examinations are facilitated by Aesthetic 6 since it can aid in human pattern recognition.

Our problem is: Given a tree T, find a eumorphous placement π of T of minimum width, where the *width* of π is defined as

$$\text{width}(\pi(T)) = \max\{\pi_x(p) - \pi_x(q) \colon p, q \text{ are nodes of } T\}.$$

Because of the Aesthetic 1, the assignment of y-values to the nodes does not affect the width of the placement. Therefore we need consider only those eumorphous placements π such that for each node p other than the root, $\pi_y(p) = \pi_y(\text{father}(p)) + 1$; in other words, the Euclidean distance from ℓ_i to ℓ_{i+1} is 1, for each $i \geqq 0$.

The Wetherell-Shannon algorithm, given in [10], produces placements satisfying the first five Aesthetics. In [7], Reingold and Tilford give examples for which the Wheterell-Shannon algorithm produces placements wider than necessary. They also present a heuristic algorithm that produces placements of narrow width satisfying all six aesthetics (i.e., eumorphous placements). Both the Wetherall-Shannon and the Reingold-Tilford algorithms can be implemented in time $\Theta(n)$, where n is the number of nodes in the tree being drawn.

There are four main results of the present paper. In Sect. II, we show that there is no obvious "principle of optimality" that could lead to a dynamic programming solution, since there exist trees whose minimum width placement cannot be achieved without making some subtrees wider than necessary. Also, we show that for an infinite class of integers n, there are n-node trees T such that

$$\frac{\text{width}(\pi(T))}{\text{width}(\pi'(T))} = \frac{n+2}{6},$$

where π' is a minimum-width eumorphous placement of T and π is the placement of T produced by the Reingold-Tilford algorithm. In Sect. III, we show that

the problem of finding a minimum width placement can be reduced in polynomial time to linear programming, and hence can be solved in polynomial time. However, as shown in Sect. IV, if we restrict placements to map nodes onto points in the integral lattice, then the problem is NP-hard; in fact, we show that if $P \neq NP$, then there does not exist a polynomial time algorithm A which produces eumorphous placements such that for each tree T,

$$\frac{\text{width}(\pi(T))}{\text{width}(\pi'(T))} < \frac{25}{24},$$

where π' is the minimum width eumorphous placement and π is the placement produced by algorithm A.

II. Suboptimality of the Reingold-Tilford algorithm

The Reingold-Tilford (RT) algorithm for producing eumorphous placements works on a tree T essentially as follows:

1. If T has a nonempty left subtree, then apply the algorithm recursively to that subtree.

2. If T has a nonempty right subtree, then apply the algorithm recursively to that subtree.

3. (a) If root (T) has only one nonempty subtree, then place that subtree, as positioned by (1) or (2), so that the root of the subtree is horizontal distance 1 from its father, on the appropriate side.

(b) If T has two nonempty subtrees then place them, as positioned by (1) or (2), as close together as possible without violating the aesthetics, and then place the root of T midway between the sons in accordance with Aesthetic 4.

It is simple to show by induction on the height of T that the algorithm gives a eumorphous placement of T.

A plausible principle of optimality would be: if a eumorphous placement π of a tree T has minimum width then for each subtree T' of T, the placement $\pi(T')$

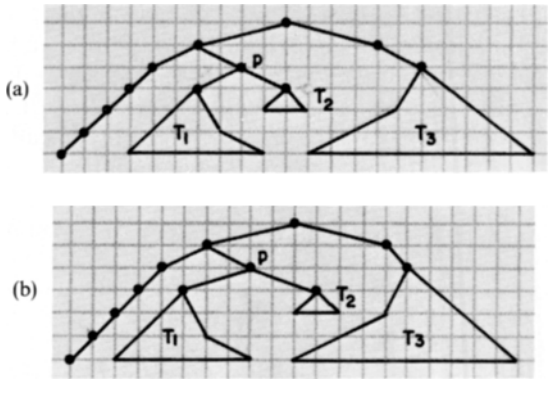


Fig. 1. Schematic view of a tree that violates the principle of optimality

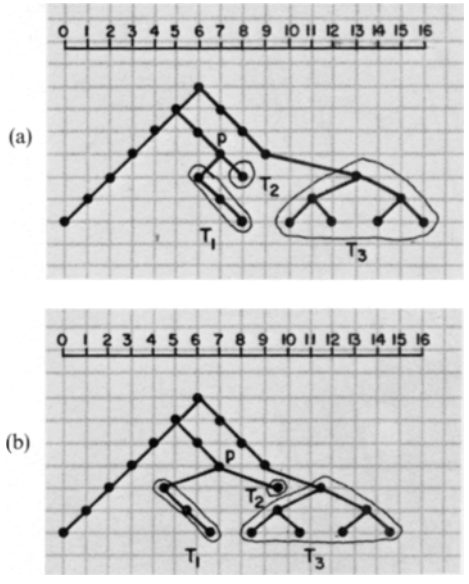


Fig. 2. A tree that violates the principle of optimality

has minimum width. It turns out, however that this is not always true, as we now show. We first informally describe the tree T pictured schematically in Figs. 1(a) and 1(b). If the subtree rooted at node p is placed with a minimum width eumorphous placement (as in Fig. 1(b)), then Aesthetic 3 forces T_3 to be placed rather far to the right, in order that the leftmost node of T_3 does not collide with the rightmost node of T_1. On the other hand, if the root of T_1 is placed farther from the root of T_2, then T_3 can be moved to the left (see Fig. 1(a)); thus making the subtree rooted at p wider than necessary, but giving the entire tree T minimum width.

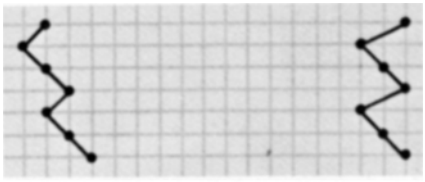
To be more precise, an example of such a tree T is given in Fig. 2(a) and 2(b). Because of Aesthetics 1 through 3, for each eumorphous placement π of T such that width (π(subtree rooted at p))) has minimum width (namely, 2), we have width$(\pi(T)) \geqq 16$; such a placement is shown in Fig. 2(a). However, the eumorphous placement π of T shown in Fig. 2(b) satisfies

$$\text{width}(\pi(\text{subtree rooted at } p)) = 5,$$

and

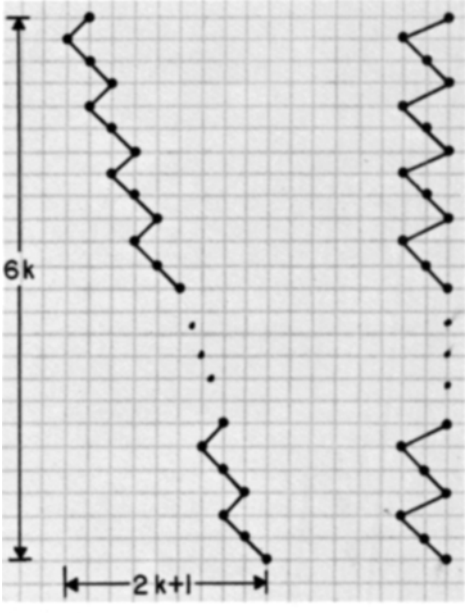
$$\text{width}(\pi(T)) = 14.5.$$

Thus, T is a counterexample to the principle of optimality stated above. Because Algorithm RT works recursively on each nonempty subtree of its input tree, the falseness of the principle of optimality proves that Algorithm RT does not always produce minimum width eumorphous placements. In fact, we now show that algorithm RT does not always even approximate the minimum width eumorphous placement.



(a) (b)

Fig. 3(a). Algorithm RT's placement of T_1 **(b).** The minimum width eumorphous placement of T_1



(a) (b)

Fig. 4(a). Algorithms RT's placement of T_k **(b).** The minimum width eumorphous placement of T_k

Figure 3(a) shows the placement π produced by Algorithm RT for a tree T_1; Figure 3(b) shows a minimum width eumorphous placement of T_1. Note that

$$\text{width}(\pi(T_1))=3 \quad \text{and} \quad \text{width}(\pi'(T_1))=2.$$

The tree T_k in Figs. 4(a) and 4(b) consists of k copies of the tree T_1 of Fig. 3, linked together vertically. Figure 4(a) shows the placement π of T_k produced by Algorithm RT; Figure 4(b) shows the minimum width eumorphous placement π'. The number of nodes in T_k is

$$k((\text{the number of nodes in } T_1)-1)+1=6k+1.$$

Also,

$$\text{width}(\pi(T_k))=k(\text{width}(\pi(T_1))-1)+1=2k+1=\frac{n+2}{3},$$

where $n=6k+1$ is the number of nodes of T_k. Thus, for an infinite class of n,

there exist trees T such that

$$\frac{\text{width}(\pi(T))}{\text{width}(\pi'(T))} = \left(\frac{n+2}{3}\right)/2 = \frac{n+2}{6},$$

where π' is the minimum width placement of T.

III. A Polynomial-Time Algorithm for the Continuous Case

In this section, we prove that the problem of computing a minimum width eumorphous placement of a tree T can be done in deterministic polynomial time. The proof is a polynomial time reduction of the problem to a linear program, which can be solved in deterministic polynomial time by Khachian's algorithm [3]. We stress that this result is not meant to be of practical importance; it serves only to contrast with the NP-hardness results of the next section. For that reason, we have not concentrated much on the efficiency of the resulting polynomial time algorithm, but have been satisfied with somewhat crude bounds on performance.

The linear program that we will construct has variables $\pi_x(p)$ for each node p in the tree, and two auxiliary variables X and x. In addition to the inequalities that we describe below, we have, for all nodes p,

$$X \geqq \pi_x(p),$$

and

$$x \leqq \pi_x(p).$$

The objective function to be minimized in the linear program is $X - x$, the width of the mapping π. Thus we will have a minimum width placement, subject to the constraints of the inequalities, as constructed from the tree and the aesthetics.

As we have stated above, the first aesthetic does not concern the $\pi_x(p)$. For the second aesthetic, we introduce the inequalities

$$\pi_x(\text{rightson}(p)) - \pi_x(p) \geqq 1, \tag{1}$$

for all nodes p for which rightson(p) exists, and

$$\pi_x(p) - \pi_x(\text{leftson}(p)) \geqq 1, \tag{2}$$

for all nodes p for which leftson(p) exists. These constraints can be computed in a preorder traversal of the tree [5]. For the third aesthetic, we have the inequalities

$$\pi_x(p_2) - \pi_x(p_1) \geqq 2, \tag{3}$$

for all nodes p_1, p_2 such that p_1 and p_2 are on the same level of the tree and p_2 is the level order successor of p_1. These constraints can be computed on a level order traversal of the tree [6]. Constraints (1), (2) and (3) also force the placement π to satisfy the fifth aesthetic, as a simple inductive argument shows. To force compliance with the fourth aesthetic, we have the equalities

$$\pi_x(\text{rightson}(p)) - \pi_x(p) = \pi_x(p) - \pi_x(\text{leftson}(p)), \tag{4}$$

for all nodes for which both leftson(p) and rightson(p) exist.

Aesthetic 6 is the most troublesome, since it requires the determination of all pairs of isomorphic subtrees in the tree. We can do this in polynomial time by labelling each node in the tree with a label consisting of

(1) the number of nodes in the subtree rooted at that node, and

(2) the rank of that subtree, where the *rank* is as defined by Knott in [4].

The computation of the rank at the root requires (recursively) the computation of the rank for each of the subtrees, and so the labelling requires a single application of an appropriate form of the rank procedure given in [4]. In addition, we may encode the number of nodes in the subtree with the rank, as Knott suggests. The rank of a subtree, together with its size, uniquely determines the subtree, up to isomorphism. Thus by sorting the nodes in increasing order by label, we can partition them into equivalence classes based on isomorphism. For each non-singleton equivalence class of subtrees other than the leaves, we add constraints as follows. Let $\{p_1, p_2, \ldots, p_k\}$ be a set of roots of subtrees in some equivalence class. We include inequalities

$$\pi_x(\text{rightson}(p_i)) - \pi_x(p_i) = \pi_x(\text{rightson}(p_{i+1})) - \pi_x(p_{i+1}) \tag{4a}$$

for all i, $l \leqq i < k$ such that right subtrees exist for trees in the equivalence class, and

$$\pi_x(p_i) - \pi_x(\text{leftson}(p_i)) = \pi_x(p_{i+1}) - \pi_x(\text{leftson}(p_{i+1})) \tag{4b}$$

for all i, $l \leqq i < k$, such that right subtrees do not exist for trees in the equivalence class.

Because corresponding subtrees of isomorphic trees are isomorphic, constraints (4a) and (4b) are sufficient to force compliance with Aesthetic 6. The time required to compute (4a) or (4b) depends on the time to compute the rank function but that is clearly polynomial, as follows from a straightforward analysis of Knott's formulae.

The total number of constraints (1), (2), (3) and (4) is $O(n)$ for a tree with n nodes.

Thus the reduction to a linear program can be done in polynomial time and by [3] can be solved in polynomial time.

IV. The NP-Hardness of the Discrete Case

In this section, we restrict placements to the integral lattice; that is, we allow only placements π such that for each node p, $\pi_x(p)$ and $\pi_y(p)$ are integers. This case is of far more practical significance than the continuous case discussed in Sect. III, because in actually drawing trees the resolution is finite and very crude on line printers, and better but still finite on plotting devices.

We will show the following discrete optimization problem to be NP-hard: Given a tree T, find a minimum width eumorphous placement of the nodes of T on the integral lattice. We prove this by showing that the corresponding decision problem is NP-complete: Given a tree T and an integer $W \geqq 1$, is there a eumorphous placement π of T on the integral lattice so that $\text{width}(\pi(T)) \leqq W$? Because integer programming is in NP [1], the results of Sect. III imply that this decision problem is in NP.

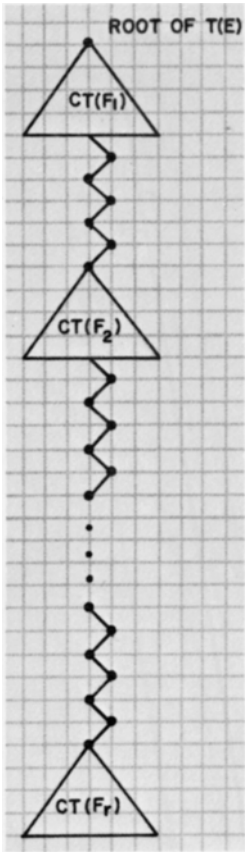


Fig. 5. Schematic view of $T(E)$, where $E=(F_1 \wedge F_2 \wedge \ldots \wedge F_r)$

To show NP-hardness of the decision problem, we will give a reduction from 3-SAT (see [2]) to the specific decision problem in which $W=24$. Let

$$E=F_1 \wedge F_2 \wedge \ldots \wedge F_r$$

be a Boolean expression over the variables $x_1, x_2, \ldots, x_n$, with clauses

$$F_i=(y_{i,1}+y_{i,2}+y_{i,3}),$$

for each i, $1 \leqq i \leqq r$, where the y_{ij} are literals. We will construct a tree $T(E)$ for which there is a eumorphous placement of width $\leqq 24$ if and only if E is satisfiable. Conceptually, T is of the form shown in Fig. 5. That is, $T(E)$ can be thought of as consisting of r *clause trees*, each corresponding to a clause F_i. Denote the clause tree for F_i by $CT(F_i)$. If E is satisfiable then there is a eumorphous placement π such that $\text{width}(\pi(CT(F_i))) \leqq 24$, for each i, $1 \leqq i \leqq r$. On the other hand, if E is not satisfiable, then for each eumorphous placement π, at least one clause tree F_j satisfies $\text{width}(\pi(CT(F_j))) \geqq 25$; hence $\text{width}(\pi(T(E))) \geqq 25$.

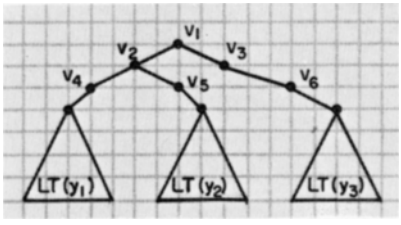


Fig. 6. Schematic view of a clause tree $CT(F)$, where $F=(y_1+y_2+y_3)$

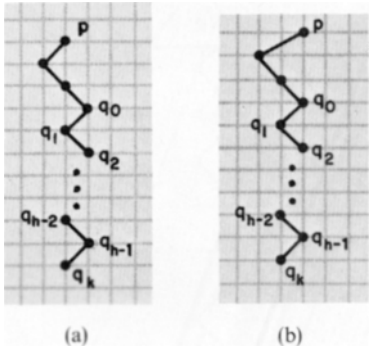


Fig. 7. The variable tree $VT(x_k)$

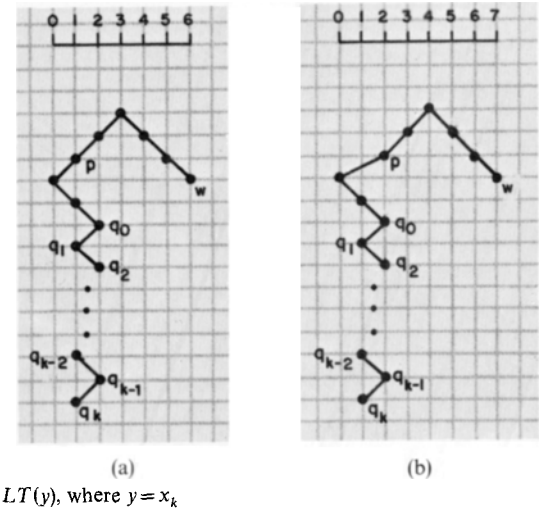


Fig. 8. The literal tree $LT(y)$, where $y=x_k$

Construction of a Clause Tree

We now describe $CT(F)$, for a clause $F=(y_1+y_2+y_3)$. $CT(F)$ contains as subtrees three *literal trees* (denoted $LT(y_1)$, $LT(y_2)$, and $LT(y_3)$), one corresponding to each literal of F, as shown in Fig. 6: v_1 is the root of $CT(F)$. It has a left son v_2 and a right son v_3. v_2 has a left son v_4 and a right son v_5. v_3 has a right son v_6. v_4 has as a left son the root of $LT(y_{i,1})$. v_5 has as a right son the root of $LT(y_{i,2})$.

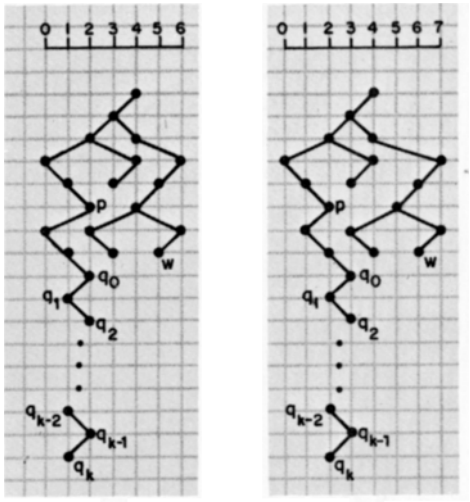


Fig. 9. The literal tree $LT(y)$, where $y=\bar{x}_k$

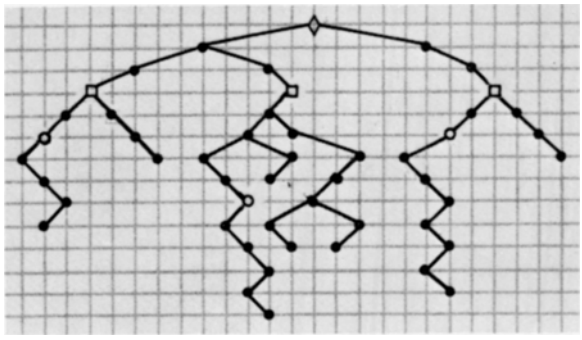


Fig. 10. The clause tree $CT(F)$, where $F=(x_1+\bar{x}_2+x_4)$. The diamond is the root of $CT(F)$; the squares are the roots of the three literal trees; the circles are the roots of the three variable trees

v_6 has as a right son the root of $LT(y_{i,3})$. Notice that the roots of the three literal trees all have the same level (namely, 3) in $CT(F)$.

We now describe $LT(y)$, for a literal y in F. The variable in the literal y is x_k for some k, $1 \leqq k \leqq n$; that is, y is x_k or $\bar{x}_k$. $LT(y)$ contains, as a subtree, the *variable tree* (denoted $VT(x_k)$) for x_k. $VT(x_k)$ is shown in Fig. 7(a) and 7(b). p is the root of $VT(x_k)$. There is a zigzagging tail of $k+1$ nodes $q_0, q_1, \ldots, q_k$. That is, for all m, $1 \leqq m \leqq k-1$, q_m has exactly one son q_{m+1}. If m is even then q_{m+1} is a left son, otherwise it is a right son.

If y is uncomplemented, i.e., $y=x_k$, then $LT(y)$ is as shown in Fig. 8(a) and 8(b). Otherwise, when $y=\bar{x}_k$, $LT(y)$ is as shown in Fig. 9(a) and 9(b). In either case, notice that $VT(x_k)$ is a subtree of $LT(y)$.

This completes the description of $CT(F)$ (later, we will say why we distinguish the node w in Figs. 8 and 9). The example $CT(F)$, where $F=(x_1+\bar{x}_2+x_3)$, is shown in Fig. 10. In Fig. 10, the root of $CT(F)$ is shown as a diamond, the three roots of literal trees are shown as squares, and the three roots of variable trees are shown as circles.

Linking the Clause Trees Together

The root of $CT(F_1)$ is also the root of $T(E)$. The only remaining detail of $T(E)$ to be specified is how the clause trees are connected to each other. Fix some i, $1 \leqq i \leqq r-1$. We will describe how $CT(F_i)$ is connected to $CT(F_{i+1})$. Consider the node of $LT(y_{i,2})$ (i.e., the middle literal of F_i) labelled w in Figs. 8 and 9. There is a zigzagging tail of $n+6$ nodes coming down from w. The $(n+6)^{\text{th}}$ node in the tail is the root of $CT(F_{i+1})$.

As an example, the complete tree $T(E)$ produced by this reduction for the expression

$$E=(x_1+\bar{x}_2+\bar{x}_3)\wedge(\bar{x}_1+x_2+x_4)\wedge(\bar{x}_2+\bar{x}_3+\bar{x}_4)$$

is shown in Fig. 11. The diamonds, squares, and circles are used to distinguish types of nodes exactly as in Fig. 10.

Proof that E is Satisfiable iff T(E) can be Placed in Width $\leqq 24$

Note that each occurrence of each variable tree in $T(E)$ is a subtree of $T(E)$. Therefore, for all k, $1 \leqq k \leqq n$, Aesthetic 6 causes each occurrence of $VT(x_k)$ to be placed identically, up to translations. This is a key point in the proof.

Assume that E is satisfiable with the truth assignment

$$\tau\colon \{x_1, x_2, \ldots, x_n\} \to \{\text{true, false}\}.$$

We will construct a placement π of $T(E)$ such that $\text{width}(\pi(T(E))) \leqq 24$.

For all k, $1 \leqq k \leqq n$, if $\tau(x_k)=$ true then let π place $VT(x_k)$ as shown in Fig. 7(a); that is, each son has the minimum (1 unit) allowed horizontal distance from its father. If $\tau(x_k)=$ false, then π places $VT(x_k)$ as shown in Fig. 7(b); that is, each son has the minimum horizontal distance from its father, except for the son of p, which has horizontal distance 2 from p.

For all i and j such that $1 \leqq i \leqq r$ and $1 \leqq j \leqq 3$, if $y_{ij}=x_k$ is uncomplemented, then let π place it as in Fig. 8(a) if $\tau(x_k)=$ true, and as in Fig. 8(b) if $\tau(x_k)=$ false. If $y_{ij}=\bar{x}_k$ is complemented, then place it as in Fig. 9(b) if $\tau(x_k)=$ true, and as in Fig. 9(a) if $\tau(x_k)=$ false. Note that if y_{ij} is true under τ, then $\text{width}(\pi(LT(y_{ij})))=6$ (Figs. 8(a) and 9(a)). If y_{ij} is false under τ, then $\text{width}(\pi(LT(y_{ij})))=7$ (Figs. 8(b) and 9(b)).

We now consider the placement of the clause trees. Let $1 \leqq i \leqq r$. Note that for all j, $1 \leqq j \leqq r$, the root of $LT(y_{ij})$ is the only son of its father (see Fig. 6). Therefore we have the freedom to place those three roots close together without worrying about Aesthetic 4. More specifically, place the root of $LT(y_{i,1})$ one unit left of its father in $CT(F_i)$. Then place the root of $LT(y_{i,2})$ as far left as possible while having each node of $LT(y_{i,2})$ placed at least 2 units to the right of each node of $LT(y_{i,1})$ (so that Aesthetic 3 is not violated). Similarly, place the root of $LT(y_{i,3})$ as far left as possible. Thus,

$$\begin{aligned}\text{width}(\pi(CT(F_i))) \leqq\ & \text{width}(\pi(LT(y_{i,1}))) + 2 \\ & + \text{width}(\pi(LT(y_{i,2}))) + 2 \\ & + \text{width}(\pi(LT(y_{i,3}))).\end{aligned}$$

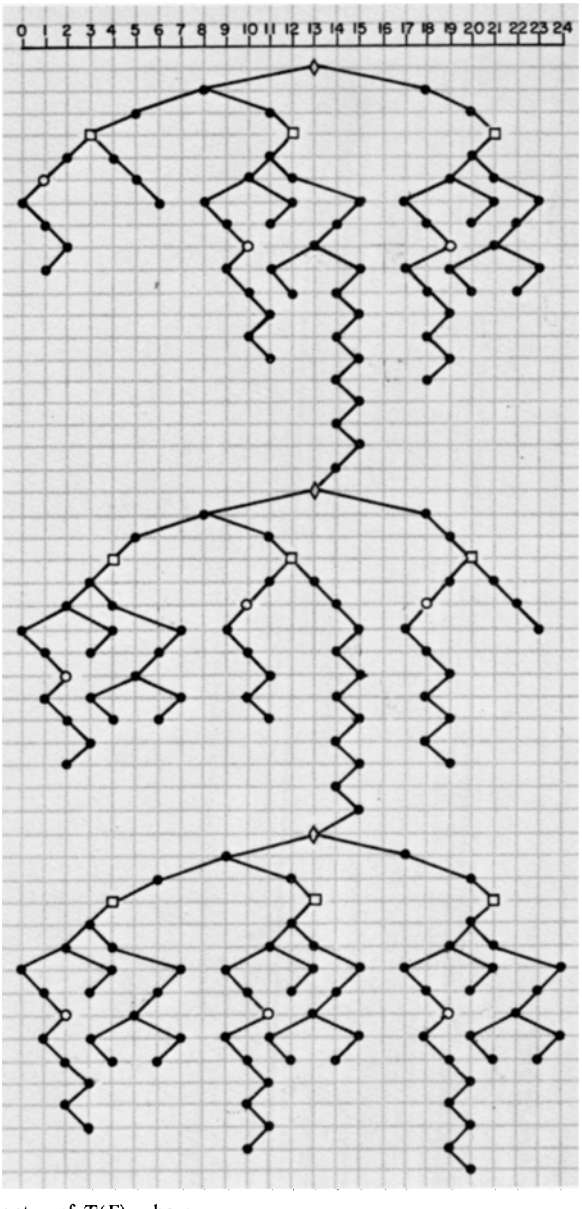


Fig. 11. A placement π of $T(E)$, where

$$E=(x_1+\bar{x}_2+\bar{x}_3)\wedge(\bar{x}_1+x_2+x_4)\wedge(\bar{x}_2+\bar{x}_3+\bar{x}_4).$$

The diamonds are the roots of the clause trees; the squares are the roots of the three literal trees; the circles are the roots of the three variable trees. π corresponds to the truth assignment τ which makes x_1, x_1, and x_4 true, and x_3 false

Now since τ satisfies E, at least one of the three literals in F_i is true under τ. Therefore at least one of $LT(y_{i,1})$, $LT(y_{i,2})$, or $LT(y_{i,3})$ is placed in width 6 by π, and the other two are placed in width 6 or 7. Therefore

$$\text{width}(\pi(CT(F_i)))\leqq 7+7+6+2+2=24.$$

We want to align the clause trees of $\tau(E)$ so that the width of $T(E)$ is equal to the width of its widest clause tree. We are free to do this without violating the Aesthetics because of the length and flexibility of the zigzagging chains that connect the clause trees: their length (i.e., $n+6$ nodes) guarantees that there will be no interference between clauses, while their flexibility allows for proper alignment. Thus, for all i, $1 \leqq i \leqq r$, let π place the root of $CT(F_i)$ so that the leftmost node in $CT(F_i)$ has x-value 0. Thus

$$\text{width}(\pi(T(E))) = \max_{1 \leqq i \leqq r} \{\text{width}(\pi(CT(F_i)))\} \leqq 24.$$

as claimed. Figure 11 illustrates π for a satisfiable E.

We now show the converse, namely that if $T(E)$ can be placed in width $\leqq 24$ then E is satisfiable. Assume that there exists a eumorphous placement π' of $T(E)$ such that $\text{width}(\pi'(T(E))) \leqq 24$. We will construct a truth assignment τ' satisfying E. Define τ' as follows: for all k, $1 \leqq k \leqq n$, let

$$\tau'(x_k) = \begin{cases} \text{true,} & \text{if there is an occurrence of } VT(x_k) \text{ in } T(E) \text{ such that } \pi' \\ & \text{places the root of } VT(x_k) \text{ exactly one unit to the right} \\ & \text{of its son (for example, } \pi' \text{ might place } VT(x_k) \text{ as shown} \\ & \text{in Fig. 7(a)).} \\ \text{false,} & \text{otherwise.} \end{cases}$$

Recall that since each occurrence of $VT(x_k)$ is a subtree of T, Aesthetic 6 tells us that each occurrence of the root of $VT(x_k)$ will have the same distance from its son.

For convenience, if T is a tree and π a placement of T, then we define $\text{width}_\ell(\pi(T))$ to be the distance between the rightmost and leftmost nodes on level ℓ in T, as placed by π. Now consider some $LT(y)$. It is straightforward to show if y is false under the truth assignment τ', then $\text{width}_3(\pi'(LT(y))) \geqq 7$. We now conclude that τ' satisfies E, since otherwise there would exist an i, $1 \leqq i \leqq r$ such that F_i is false under τ'. But then, by the above remarks,

$$\begin{aligned} \text{width}(\pi'(T(E))) &\geqq \text{width}(\pi'(CT(F_i))) \\ &\geqq \text{width}_6(\pi'(CT(F_i))) \\ &\geqq 7+2+7+2+7 \\ &= 25, \end{aligned}$$

contradicting the choice of π'.

$T(E)$ has $O(nk)$ levels, each containing $O(1)$ nodes. Therefore the construction of $T(E)$ can be performed in time polynomial in the size of E.

We summarize this result as:

Theorem. *Given a rooted binary tree T and a positive integer W, the problem of determining the existence of a eumorphous placement on the integral lattice of T of width at most W is NP-complete. In fact, the specific sub-problem in which $W = 24$ is NP-complete.*

Immediate from the theorem is the following result on the complexity of near-minimum width tree-drawings:

Corollary. *If $P \neq NP$, then there is no polynomial time algorithm A that produces eumorphous placements on the integral lattice such that for all trees T,*

the width of A's placement of T

$$< \frac{25}{24} \text{ (width of the narrowest eumorphous placement of } T\text{)}.$$

Informally, the corollary says that it is NP-hard to approximate minimum width eumorphous placements of trees to within a factor of less than about 4 per cent.

V. Conclusions

We can generalize Aesthetic 2 to state that for each node p that has a right son,

$$\pi_x(\text{rightson}(p)) - \pi_x(p) \geqq \delta,$$

and for each node p that has a leftson

$$\pi_x(p) - \pi_x(\text{leftson}(p)) \geqq \delta;$$

and Aesthetic 3 to state that for each two nodes p, q having the same level,

$$|\pi_x(p) - \pi_x(q)| \geqq 2\delta,$$

where $\delta > 0$ is a real number giving the minimum horizontal separation. Thus, as we have phrased Aesthetics 2 and 3 in Sect. I, $\delta = 1$. The results of Sect. II and III are not changed by this generalization. The NP-hardness reduction in Sect. IV easily generalizes to the case in which $\lceil \delta \rceil$ is odd. The Corollary generalizes to: For all $\delta > 0$ such that $\lceil \delta \rceil$ is odd, $P \neq$ NP implies that there is no polynomial time algorithm A for the discrete optimization problem with minimum horizontal separation δ such that for all trees T,

$$\text{width of } A\text{'s placement of } T < \frac{24\lceil \delta \rceil + 1}{23\lceil \delta \rceil + 1}.$$

We believe that a similar result also holds when $\lceil \delta \rceil$ is even.

The open problems concerning the complexity of tree drawing include:

1. If we drop the Aesthetic 6, is the discrete optimization problem still NP-hard? Note that our reduction depends heavily on Aesthetic 6.

2. By the corollary, it is NP-hard to approximate the discrete optimization problem to within a factor of less than $\frac{25}{24}$. Is there some constant $c > 0$ and some polynomial time heuristic algorithm that always approximates the optimal width to within a factor of c? For the general integer programming problem, it is known that for all constants $c > 0$, $P \neq$ NP implies that no polynomial time algorithm approximates the optimal to within a factor of c [8]. Is this also true for the special case of integer programming that corresponds to our discrete tree-drawing optimization problem?

3. Is there an algorithm for the continuous optimization problem that is faster or easier to implement than Khachian's linear programming algorithm [3]?

4. Is there a simple set of aesthetics that more accurately captures our intuitive notion of nice tree drawings?

References

1. Borosh, I., Treybig, L.B.: Bounds on Positive Integral Solutions of Linear Diophantine Equations. Proc. Amer. Math. Soc. **55**, 299–304 (1976)
2. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., San Francisco, 1979
3. Khachian, L.G.: A Polynomial Algorithm for Linear Programming. Doklady Akad. Nauk SSSR **244**, 191–194 (1979)
4. Knott, G.D.: A Numbering System for Binary Trees. Comm. ACM **20**, 113–115 (1977)
5. Knuth, D.E.: The Art of Computer Programming, Vol.1: Fundamental Algorithms. Addison-Wesley Publishing Co., Reading, MA, 1968
6. Reingold, E.M., Nievergelt, J., Deo, N.: Combinatorial Algorithms: Theory and Practice. Prentice-Hall: Englewood Cliffs, New Jersey, 1977
7. Reingold, E.M., Tilford, J.S.: Tidier Drawings of Trees. IEEE Trans. Software Engineerg. **7** 223–228 (1981)
8. Sahni, S., Gonzalez, T.: *P*-complete Approximation Problems. J. ACM **23**, 555–565 (1976)
9. Vaucher, J.G.: Pretty-Printing of Trees. Software-Practice and Experience **10**, 553–561 (1980)
10. Wetherell, C., Shannon, A.: Tidy Drawings of Trees. IEEE Trans. Software Engineerg. **5**, 514–520 (1979)

Received February 25, 1981/September 20, 1982