<table>
<thead>
<tr>
<th>Title</th>
<th>Title of the document in Japanese</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Tsuchida, Kensei</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (1991), 754: 228-236</td>
</tr>
<tr>
<td>Issue Date</td>
<td>1991-06</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/82099">http://hdl.handle.net/2433/82099</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>

京都大学
Layout Problems of Tree Structured Diagrams

Kensei Tsuchida

Department of Industrial Engineering and Management
Kanagawa University
3-27-1 Rokkakubashi, Kanagawa-ku, Yokohama 221, JAPAN

1 Introduction

Visualizing information has been one of the main subjects in computer field. Representing objects by diagrams is one approach of visualizing information and its effectiveness is widely recognized. Recently many graph drawing algorithms have been proposed. Several authors have studied the problem of producing tidy drawings of binary trees - drawings that are aesthetically pleasing and of minimum width. A number of aesthetics have been proposed.

In this paper, we extend this problem to that of general tree structured diagrams. We formalize tree structured diagrams and modify the aesthetics to apply the problem to the layout problems of that. Moreover we investigate the complexity of producing drawings of minimum width under certain constraints which we introduce.

In Section 2 we give preliminary definitions and introduce constraints. In Section 3 we show the problem is NP-hard under certain constraints.
2 Preliminary Definitions

Throughout this paper, we deal with a treelike diagram which we called a tree structured diagram. The tree structured diagram is a general form of various flow diagrams such as data-flow diagrams, hierarchical program diagrams and entity-relationship diagrams. In a tree structured diagram each rectangular box(cell) is placed tree-like on the integral lattice.

Definition 1 The tree structure $T$ is defined as follows.

$$T = (V, E, r, W, D)$$

Where $V$ is a set of cells, $E$ is a set of edges, $(V, E)$ is a ordered tree, $r$ is a rooted cell in $V$, $W$ is a width function of cell: $V \rightarrow Z$ and $D$ is a depth function of cell: $V \rightarrow Z$.

Terms width and depth mean lengths in x-direction and y-direction in this paper.

Definition 2 The placement of the tree structure $T$ is expressed as a function $L: V \rightarrow Z^2$ (the integral lattice).

Where if $L(p) = (x, y)$ then $L_x(p)$ and $L_y(p)$ are $x$ and $y$ coordinates of $L(p)$. The pair $(T, L)$ is called the tree structured diagram.

The tree structure $T$ can be viewed as a rooted tree in which each node(cell) $p$ is assigned to two attributes $W(p)$ and $D(p)$. The tree structured diagram $(T, L)$ can be viewed as a rooted tree in which each node(cell) $p$ is assigned to four attributes $W(p)$, $D(p)$, $L_x(p)$ and $L_y(p)$.

Definition 3 The width of the tree structured diagram $W_t(T, L)$ is defined as follows.

$$W_t(T, L) = \max \{L_y(p) + W(p) - L_y(q)\},$$

where $p$ and $q$ are cells of $T$.

Definition 4 The level of the cell $p$ is defined as the number of edges between $p$ and the rooted cell.

Definition 5 The function $Index: cells \rightarrow integers$ is defined as follows. If the cell $p$ is a rooted cell then $Index(p) = 0$. Else if $p$ is the $i$-th son of $p$'s father then $Index(p) = i$.

Definition 6 The box area of the cell $p$ is defined by the set $Area(p, L)$.

$$Area(p, L) = \{(x, y) \mid L_x(p) \leq x \leq L_x(p) + \text{depth}(p),$$

$$L_y(p) \leq y \leq L_y(p) + \text{width}(p)\}$$
**Definition 7** Drawing a tree structured diagram \((T, L)\) is the drawing of a straight line segment joining a point \((L_x(p) + W(p), L_y(p) + \frac{1}{2}D(p))\) to a point \((L_x(q), L_y(q) + \frac{1}{2}D(q))\) each edge \((p, q)\) (from \(p\) to \(q\)) in \(T\).

Now we introduce several constraints for drawings of tree structured diagrams. For a tree structure \(T\), we consider the placement \(L\) such that the tree structure diagram \((T, L)\) has the minimum width under certain constraints. A constraint is a condition for drawing tree structured diagrams nicely. The following constraints are modifications of binary trees or added new.

**B1** If the level of cell \(p\) is equal to that of the cell \(q\) and \(L_y(p) < L_y(q)\) then \(L_y(p)\) (the eldest son of \(q\)) > \(L_y\) (the youngest son of \(q\))

**B2** If \(p\) and \(q\) are different cells then there is no common part of \(Area(p, L)\) and \(Area(q, L)\).

**B3** If \(T_1\) and \(T_2\) are isomorphic subtrees (each corresponding cells have the same attributes(sizes)) then \(L\) must place \(T_1\) and \(T_2\) identically up to a translation.

**B4(a)** In the tree structured diagram \((T, L)\) if cells \(p\) and \(q\) are brothers then \(L_x(p) = L_x(q)\).

**B4(b)** If levels of cells \(p\) and \(q\) are the same then \(L_x(p) = L_x(q)\).

**B4(c)** If the cell \(p\) is the father of the cell \(q\) then \(L_x(q) = L_x(p) + W(p) + 1\)

Note that satisfying B4(c) implies satisfying B4(a), and B4(b) implies B4(a).

**B5** If the cell \(p\) has \(k\) sons \(q_1, \ldots, q_k (\text{Index}(q_i) = i)\) then
\[
L_y(p) = L_y(q_{\lfloor k/2 \rfloor})
\]

**B6** If the cell \(p\) has \(k \geq 3\) sons \(q_1, \ldots, q_k (\text{Index}(q_i) = i)\) then
\[
L_y(q_{j+2}) - L_y(q_{j+1}) = L_y(q_{j+1}) - L_y(q_j)
\]
\[
(1 \leq j \leq k - 2)
\]
Here we define constraints $C_1$, ..., $C_6$ by composing these above constraints.

\[
C_1 = B1 \land B2 \land B3 \land B4(a) \land B5 \land B6 \\
C_2 = B1 \land B2 \land B3 \land B4(b) \land B5 \land B6 \\
C_3 = B1 \land B2 \land B3 \land B4(c) \land B5 \\
C_4 = B1 \land B2 \land B3 \land B4(a) \land B5 \\
C_5 = B1 \land B2 \land B3 \land B4(b) \land B5 \\
C_6 = B1 \land B2 \land B3 \land B4(c) \land B5
\]

3 NP-Hardness

In this section, we prove that the complexity of determining the minimum width under constraint $C_1$ is NP-hard. Next we can also show that if we omit $B6$ the problem is still NP-hard. We can show this analogously to Supowit-Reingold’s method. Our decision problem is: For a given tree structure $T$ and a positive integer $M$, is there a placement $(T,L)$ satisfying $C_1$ such that $\text{Wt}(T,L) \leq M$?

To show NP-hardness of this decision problem, we will a reduction from 3-SAT to the specific decision problem with $M = 81$. Let

\[E = F_1 \land F_2 \land ... \land F_r\]

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

\[F_i = (y_{i,1} + y_{i,2} + y_{i,3})\]

for each $i$ where $1 \leq i \leq r$ and $y_{i,j}$ are literals.

3.1 Construction of a Tree Structure

Let $E$ be a Boolean expression as above. We construct a tree structure $T(E)$ for which there exists a placement $L$ satisfying $C_1$ such that $\text{Wt}(T(E), L) \leq 81$ iff $E$ is satisfiable. $T(E)$ is the form shown in Fig.1. We denote the clause tree structure for $F_i$ by $CT(F_i)$, where $F_i = (y_{i,1} + y_{i,2} + y_{i,3})$. $CT(F_i)$ contains subtree structures $LT(y_{i,1})$, $LT(y_{i,2})$ and $LT(y_{i,3})$ which corresponding to each literal of $F_i$, as shown Fig.2. Notice that all distances of $v_1$ and $v_2$, $v_3$ and $v_4$ and $p_1$ and $p_2$ can be expended without violating $C_1$. $LT(y)$ contains variable tree structure $VT(x_k)$ for $x_k$. $VT(x_k)$ is shown in Fig.3. In Fig.3 if $k$ is even $VT(x_k)$ is (a.1) or (b.1) otherwise (a.2) or (b.2). If $y = x_k$, $LT(y)$ is as shown in Fig.4. If $y = \overline{x}_k$, ...
LT(y) is as shown in Fig.5. CT(F_i) is connected to CT(F_{i+1}) as follows. Consider the cell of LT(y_{i,3}) labelled w in Fig.4 and Fig.5. We make a straight tail of (4n + 10) cells coming down from w and link them so that the (4n + 10)'th cell in the tail is the root of CT(F_{i+1}).

### 3.2 Proof of Satisfiability

We can prove that E is satisfiable iff T(E) can be placed in Wt(T(E),L) ≤ 81 by the similar argument of Supowit-Reingold' proof [1]. Here we omit the detail of the proof.

Hence we obtain the following theorem.

**Theorem 1** For a given tree structure T and a positive integer M, the problem of determining the existence of a placement L such that Wt(T(E),L) is at most M while satisfying C_1 is NP-hard. In fact, the specific sub-problem with M = 81 is NP-hard.

We can also show that the complexity of determining the minimum width under constraint C_4 is NP-hard. To show this, we have only to modify the proof of the case of C_1 so that if y = x_k, LT(y) is as shown in Fig.6 instead of Fig.5. Similarly we obtain the following theorem.

**Theorem 2** For a given tree structure T and a positive integer M, the problem of determining the existence of a placement L such that Wt(T(E),L) is at most M while satisfying C_4 is NP-hard. In fact, the specific sub-problem with M = 81 is NP-hard.

Furthermore we can obtain following corollaries by definitions of other constraints.

**Corollary 1** For a given tree structure T and a positive integer M, the problem of determining the existence of a placement L such that Wt(T(E),L) is at most M while satisfying C_2 is NP-hard. In fact, the specific sub-problem with M = 81 is NP-hard.

**Corollary 2** For a given tree structure T and a positive integer M, the problem of determining the existence of a placement L such that Wt(T(E),L) is at most M while satisfying C_3 is NP-hard. In fact, the specific sub-problem with M = 81 is NP-hard.

**Corollary 3** For a given tree structure T and a positive integer M, the problem of determining the existence of a placement L such that Wt(T(E),L) is at most M while satisfying C_5 is NP-hard. In fact, the specific sub-problem with M = 81 is NP-hard.

**Corollary 4** For a given tree structure T and a positive integer M, the problem of determining the existence of a placement L such that Wt(T(E),L) is at most M while satisfying C_6 is NP-hard. In fact, the specific sub-problem with M = 81 is NP-hard.
Fig. 1 Schematic view of $T(E)$, where $E = F_1 \land F_2 \land \ldots \land F_r$.

Fig. 2 Schematic view of a clause tree structure $CT(F_i)$, where $F_i = (y_{i,1} + y_{i,2} + y_{i,3})$. 
Fig. 3 The variable tree structure $VT(x_k)$, where if $k$ is even it is (a.1) or (b.1) otherwise (a.2) or (b.2).

Fig. 4 The literal tree structure $LT(y)$, where $y = x_k$. 

7
Fig. 5 The literal tree structure $LT(y)$, where $y = \bar{x}_k$.

Fig. 6 The literal tree structure $LT(y)$, where $y = \bar{x}_k$. 
References


