数理解析研究所講究録
943 巻 1996 年 79-87

Efficient Drawing Algorithms on the Minimum Area for Tree-Structured Diagrams

Kensei Tsuchida * (土田 賢省)
Takeo Yaku † (夜久 竹夫)

Abstract

In this paper, we deal with a treelike diagram which we call a "tree structured diagram"(TSD for short). A TSD is a generalization of program diagrams. We firstly define the problem of drawing TSDs and introduce constraints for beautiful drawings of TSDS. Then we present efficient $O(n)$ and $O(n^2)$ algorithms which produces minimum width drawing unders certain sets of constraints. These algorithms will be applied to practical uses such as visual programming and others.

1 Introduction

Recently a number of algorithms for drawing various graphs and diagrams such as planar graphs[3, 4, 7, 8, 11, 1, 12], undirected graphs[6, 9], hierarchic graphs[5, 13, 19], data-flow diagrams[2, 17], program diagrams[10, 14, 15, 16] and entity-relationship diagrams[20] have been proposed.

Among them, drawing trees is a basic and important problem. It has various applications such as visual programming, data presentations and others. For example , in visual programming program diagrams generally have a tree structure in the sub-diagrams. In order that a processing system of program diagrams is practical use and useful, program generators which based on efficient algorithms of nicely drawing trees is needed. Thus the tidy drawing problem of trees has become an important theme.

Several authors have studied the problem of producing tidy drawings of binary trees, i.e. the problem of producing drawings that are aesthetically pleasing and of minimum width. C.Wetherell and A.Shannon formalized the constraints for the tidy drawing of binary trees and proposed a linear time algorithm to draw binary trees under the constraints[24]. M.Reingold and S.Tilford presented a linear time algorithm which gives narrower drawings of binary trees than Wetherell and Shannon while satisfying the Wetherell-Shannon's constraints [18]. Tsuchida also presented two efficient algorithms for drawing n-ary trees nicely[21, 22]. One is the $O(n)$ time algorithm drawing optimal trees under the constraint in which adjoining two sub-trees must be apart from at least one unit each other. Another is the $O(n^2)$ time algorithm drawing optimal trees under the constraint in which two sub-trees are allowed to intersect each other. These algorithms are modified and applied to the processing system for program diagrams.

In this paper, we deal with a treelike diagram which we call a "tree structured diagram"(TSD for short). A TSD is a generalization of program diagrams[25]. The TSD is a tree structure whose node is a rectangular box (which is called a *cell*). We define the problem of drawing TSDs and introduce constraints for beautiful drawings of TSDS. There are some differences between TSDS and trees with respect to drawings. However, Tsuchida proved that problems of minimum width drawings of TSDS are NP-complete under certain sets of constraints[23]. In this paper, we present efficient $O(n)$ and $O(n^2)$ algorithms which produces minimum width drawing unders certain sets of constraints.

In Section 2, we formalize the problem of drawing TSDs and introduced constraints for drawing TSDs.

*Department of Information and Computer Sciences, Toyo University, Email: kensei@krc.eng.toyo.ac.jp
†Department of Applied Mathematics, Nihon University, Email: yaku@chs.nihon-u.ac.jp

In Section 3, we present an $O(n)$ algorithm which produces the narrowest drawing of a given TSD under certain sets of constraints.

In Section 4, we present an $O(n^2)$ time algorithm which produces the narrowest drawing of a given TSD under certain sets of constraints.

In Section 5, we summarize our results.

2 Preliminary Definitions

We denote by Z the set of integers.

Definition 1. A *tree structured diagram* T is defined by

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

where V is a set of *cells*, E is a set of *edges*, (V, E) is a *directed ordered tree* with the *root*, r is the *root cell* in V, $W : V \to Z$ is the *width function* of cells and $D : V \to Z$ is the *depth function* of cells.

We assume that, for each edge $(p, q) \in E$, p is the *father* of q. In this thesis, the term width(depth) is the horizontal(vertical) length of a cell. A TSD T can be considered as a rooted tree in which each node p is associated with two attributes $W(p)$ and $D(p)$. We take the coordinate system as shown in Fig. 1. Each vertex of a cell is placed on the integral lattice Z^2. A *placement* of a TSD T is a function $\pi : V \to Z^2$ (the integral lattice), where V is the set of cells of T. A placement π maps the left upper corner of a cell to a point in Z^2. If $\pi(p) = (x, y)$ then we define $\pi_x(p) = x$ and $\pi_y(p) = y$.

Definition 2. The *width* $Wt(T, \pi)$ of a TSD T placed by π is defined by
$Wt(T, \pi) = \max\{\pi_x(p) + W(p) - \pi_x(q) \mid p \text{ and } q \text{ are cells of } T\}$.

For example, $Wt(T, \pi) = 7$ in the case of Fig. 1.

The *level* of a cell p in a TSD T is defined as the number of edges between p and the root cell of T. The function *Index* is defined as follows : if p is the root cell then $Index(p) = 0$, else if p is the i-*th* son of p's father then $Index(p) = i$.

Definition 3. The *area* of a cell p with respect to π is defined by

$$Area(p, \pi) = \{(x, y) \mid \pi_x(p) \le x \le \pi_x(p) + W(p),$$

$$\pi_y(p) \le y \le \pi_y(p) + D(p)\}.$$

Definition 4. *Drawing* a TSD T placed by π is drawing the boundary of $Area(p, \pi)$ for each cell p in T and drawing, for each edge (p, q) in T, a straight line segment joining the point $(\pi_x(p) + \frac{1}{2}W(p)$, $\pi_y(p) + D(p))$ to the point $(\pi_x(q) + \frac{1}{2}W(q)$, $\pi_y(q))$.

Definition 5 A function *VP(Vertical Position)* mapping a cell p of a TSD T to a non-negative integer is defined as

$$VP(p) = D(v_0) + \sum_{i=1}^{i=k}(1 + D(v_i)),$$

where $(v_0, .., v_k)$ is the path from the root v_0 to the cell $p(= v_k)$.

Definition 6 A function *Intersect* from the set of TSDs to integers is defined as;

$$Intersect(T,\pi) = \max\{\pi_x(p) + W(p) - \pi_x(q) + 1 \mid p\ and\ q\ are\ any\ cells\ of\ subtrees\ T_1\ and\ T_2\ respectively\ such\ that\ the\ roots\ of\ T_1\ and\ T_2\ are\ brothers\ and\ Index(the\ root\ of\ T_1) < Index(the\ root\ of\ T_2)\}$$

The function *Intersect* indicates intersecting degrees of adjoining two cells.

Now we introduce several constraints for the drawings of tree structured diagrams. We denote by $\pi(T)$ a TSD T placed by π. Let p and q be arbitrary cells in a TSD T placed by π.

$\mathbf{B_d1(a)}$. If a cell p is the father of a cell q, then $\pi_y(q) = \pi_y(p) + D(p) + 1$.

$\mathbf{B_d1(b)}$. If levels of cells p and q are the same then $\pi_y(p) = \pi_y(q)$.

$\mathbf{B_d2}$. If a cell p has k sons q_1, ..., q_k, where $Index(q_i) = i$, then

$$\pi_x(p) = \pi_x(q_{\lceil (k+1)/2 \rceil}).$$

$\mathbf{B_d3}$. If a cell p has $k(\geq 2)$ sons $q_1, ..., q_k$, where $Index(q_i) = i$, then

$$\pi_x(q_i) + W(q_i) < \pi_x(q_{i+1}).$$

$\mathbf{B_d4}$. For two cells p and q, if $VP(p) = VP(q)$ and $\pi_x(p) < \pi_x(q)$, then $\max\{\pi_x(s) + W(s) \mid s\ is\ a\ son\ of\ p\} < \min\{\pi_x(s) \mid s\ is\ a\ son\ of\ q\}$.

$\mathbf{B_d5}$. If T_1 and T_2 are isomorphic sub-TSDs (i.e., they have the same tree structure and each corresponding cell has the same width and depth) then π must place T_1 and T_2 identically up to a translation.

$\mathbf{B_d6}$. If p and q are different cells, then $d(Area(p,\pi), Area(q,\pi)) \geq 1$, where d is the Euclidean distance and $d(A,B)$ is the minimum distance between a point in A and a point in B.

$\mathbf{B_d7}$. If T_1 and T_2 are sub-TSDs whose roots are brothers and $Index(the\ root\ of\ T_2) = Index(the\ root\ of\ T_2) + 1$, then $\max\{\pi_x(s) + W(s) \mid s \in T_1\} \leq \pi_x(the\ root\ of\ T_2)$and $\pi_x(the\ root\ of\ T_1) \leq \min\{\pi_x(s) + W(s) \mid s \in T_2\}$.

$\mathbf{B_d8(k)}$. For given a non-negative integer k, the placement $\pi(T)$ satisfies the inequality; $Intersect(T,\pi) \leq k$.

$\mathbf{B_d\#}$. If a cell p has $k(\geq 3)$ sons $q_1, ..., q_k$, where $Index(q_i) = i$, then for each $j (1 \leq j \leq k-2)$

$$\pi_x(q_{j+2}) - \pi_x(q_{j+1}) = \pi_x(q_{j+1}) - \pi_x(q_j).$$

Here we consider the following sets of constraints $C_d{}^a, C_d^b, C_d^a\#, C_d^b\#, C_d^{a+}$ and $C_d^{a+}(k)$ by combining the above constraints.

$C_d^a = B_d1(a) \wedge B_d2 \wedge B_d3 \wedge B_d4 \wedge B_d5 \wedge B_d6,$

$C_d^b = B_d1(b) \wedge B_d2 \wedge B_d3 \wedge B_d4 \wedge B_d5 \wedge B_d6,$

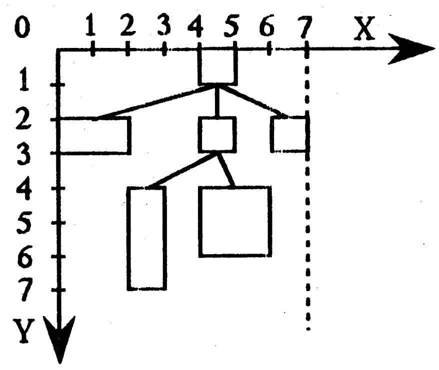

Figure 1: A tree structured diagram T.

$C_d^a\# = C_d^a \wedge B_d\#$, $C_d^b\# = C_d^b \wedge B_d\#$,

$C_d^{a+} = B1(a) \wedge B2 \wedge B3 \wedge B4 \wedge B5 \wedge B6 \wedge B7$,

$C_d^{a+}(k) = C_d^{a+} \wedge B8(k)$.

In this paper, for a TSD T we consider the placement π such that $\pi(T)$ has the minimum width under certain set of constraints.

3 $O(n)$ time algorithm

In this section, we construct an $O(n)$ time algorithm which produces the minimum width drawings of TSDs while satisfying the constraint $C_d^{a+}(0)$, where n is the number of cells in a given TSD. This algorithm traverses a given TSD in postorder and evaluates three values $L(p)$, $R(p)$ and $DI(p)$ for each cell p. Next it places each cell p in preorder with referring to the value $DI(p)$.

The values $L(p)$ and $R(p)$ for a cell p represent how far the sub-TSD, whose root cell is p, spreads out left-hand side and right-hand side respectively. Given a TSD T and its placement $\pi(T)$, $L(p)$ and $R(p)$ for a cell p of T are defined by

$L(p) = \pi_x(p) - \min\{\pi_x(q) \mid q \in T'\}$,

$R(p) = \max\{\pi_x(q) + W(q) \mid q \in T'\} - \pi_x(p)$,

where T' is the sub-TSD whose root cell is p.

The value $DI(p)$ is the distance in the direction of the x-axis between p and p's father, and defined by

$DI(p) = \pi_x(p) - \pi_x(p's\ father)$.

First, we denote the properties that hold among L, R, DI and constraints stated above.

Lemma 1 *For a TSD T and its placement $\pi(T)$, if L, R and DI satisfy the following (i), (ii) and the constraint B_d1 then $\pi(T)$ satisfies constraints B_d2, B_d3, B_d4, B_d6, B_d7 and $B_d8(0)$.*

(i) For a cell p which is an only son of its father, $DI(p) = 0$.

(ii) For more than 2 brothers $q_1, ..., q_k (2 \leq k, Index(q_i) = i, 1 \leq i \leq k)$, $DI(q_{j+1}) - DI(q_j) \geq R(q_j) + L(q_{j+1}) + 1 (1 \leq j \leq k-1)$ and $DI(q_m) = 0$, where $m = \lceil (k+1)/2 \rceil$. □

Lemma 2 *For a TSD T and its placement $\pi(T)$, if DI satisfies the following (i) and the constraint B_d1 then $\pi(T)$ satisfies constraint B_d5.*

(i) If T_1 and T_2 are isomorphic sub-TSDs of T and a cell p_1 of T_1 corresponds to a cell p_2 of T_2, then $DI(p_1) = DI(p_2)$. □

Here we state the algorithm which constructs the placement $\pi(T)$ for a given TSD T.

Algorithm Layout-1
Input. A TSD $T = (V, E, r, W, D)$ whit n cells.
Output. $\pi(T)$: the placement of T.
Method.
(1) For each leaf cell p, let $L(p) = 0, R(p) = W(p), DI(p) = 0$.
(2) Traversing the TSD T in postorder, when each cell p is visited, evaluate $DI(p)$, $L(p)$ and $R(p)$ in the following way.
(Case.1) In the case of that a cell p has only one son q, let
$L(p) = L(q), R(p) = \max(W(p), R(q)), DI(q) = 0$.
(Case.2) In the case of that a cell p has exactly two sons
q_1 and $q_2 (Index(q_i) = i, 1 \le i \le 2)$, let
$DI(q_2) = 0$,
$DI(q_1) = R(q_1) + L(q_2) + 1$,
$L(p) = L(q_1) + DI(q_1)$,
$R(p) = \max(W(p), R(q_2))$.
(Case.3) In the case of that a cell p has $k (k \ge 3)$ sons $q_1, \ldots, q_k$
$(Index(q_i) = i, 1 \le i \le k)$ and $m = \lceil (k+1)/2 \rceil$, let
$DI(q_m) = 0$,
$DI(q_{m-1}) = R(q_{m-1}) + L(q_m) + 1$,
(for $j \leftarrow m-2$ step -1 until 1)
$DI(q_j) = DI(q_{j+1}) + L(q_{j+1}) + R(q_j) + 1$,
$DI(q_{m+1}) = L(q_{m+1}) + R(q_m) + 1$,
(for $j \leftarrow m+2$ step 1 until k)
$DI(q_j) = DI(q_{j-1}) + R(q_{j-1}) + L(q_j) + 1$,
$L(p) = L(q_1) + DI(q_1)$,
$R(p) = \max(W(p), R(q_k) + DI(q_k))$.
(3) Place the root cell r at the origin, this is, let
$\pi_x(r) = 0, \pi_y(r) = 0$.
Next traversing the TSD T in preorder,place each cell p, whose father is q, as follows.
$\pi_y(p) = \pi_y(q) + D(q) + 1$, $\pi_x(p) = \pi_x(q) + DI(p)$.

Lemma 3 *For a given TSD T, the placement $\pi(T)$ which produced by the algorithm Layout-1 satisfies the constraint $C_d^{a+}(k)$.* □

Lemma 4 *For a given TSD T, the placement $\pi(T)$ which produced by the algorithm Layout-1 is a minimum width under the constraint $C_d^{a+}(0)$.* □

Lemma 5 *The algorithm Layout-1 requires O(n) time, where n is the number of cells of a given TSD.* □

We summarize these results as:

Theorem 1 *For a given TSD T whit n cells, there is an O(n) time algorithm which produces the minimum width placement of T under $C_d^{a+}(0)$.* □

4 $O(n^2)$ time algorithm

In this section, we construct an $O(n^2)$ time algorithm which produces the minimum width drawings of TSDs while satisfying the constraint $C_d^{a+}(k)$, where n is the number of cells in a given TSD. In the similar way of the algorithm Layout-1, this algorithm traverses a given TSD in postorder and evaluates two arraies $AL(p)$, $AR(p)$ and the value $DI(p)$ for each cell p. Next, in the same manner of Layout-1, it places each cell p in preorder with referring to the value $DI(p)$.

$DI(p)$ is the same in the previous section. The array $AL(p)$ (resp., $AR(p)$) for a cell p represents the left (resp., right)-hand side outline of the sub-TSD whose root cell is p. For given a TSD T, its placement $\pi(T)$ and a cell p of T, the both lengths of $AL(p)$ and $AR(p)$ are equal to $\max\{VP(p,T); p\ is\ a\ leaf\ of\ T\}$ and the i-th values of them are defined as follows.

$AL_i(p) = 0 \quad if\{q; q \in T', i \in [VP(q,T') - D(q), VP(q,T')]\}$
$(= V(i)) = \phi$
$\min\{\pi_x(q); q \in V(i)\} - \pi_x(p) \quad$ otherwise,

$AR_i(p) = 0 \quad$ if $V(i) = \phi$
$\max\{\pi_x(q) + W(q); q \in V(i)\} - \pi_x(p) \quad$ otherwise,
where T' is the sub-TSD whose root cell is p.

First, we denote the properties that hold among AL, AR, DI and constraints stated above. Note that values $AL_i(p)$ and $AR_i(p)$ are not always non-negative.

Lemma 6 *For a given positive integer k, a TSD T and its placement $\pi(T)$, if AL, AR and DI satisfy the following (i), (ii) and the constraint B_d1, then $\pi(T)$ satisfies constraints B_d2, B_d3, B_d4, B_d6, B_d and $B_d8(k)$.*

(i) For a cell p which is an only son of its father,
(for $1 \le j \le M$) $AL_j(p) = 0$,
(for $1 \le j \le 1 + D(p)$) $AR_j(p) = W(p)$,
(for $D(p) + 1 \le j \le M$) $AR_j(p) = 0$,
$DI(p) = 0$, where $M = \max\{VP(s,T); s\ is\ a\ leaf\ of\ T\}$.
(ii) For more than 2 brothers $q_1, \ldots, q_l$
($Index(q_s) = s, 1 \le s \le l$), i ($1 \le i \le l-1$) and j ($1 \le j \le M$),
$DI(q_{i+1}) - DI(q_i) \ge AR_j(q_i) - AL_j(q_{i+1}) + 1 \quad (1)$,
$DI(q_{i+1}) - DI(q_i) \ge -AL_j(q_{i+1}) + 1 \quad (2)$,
$DI(q_{i+1}) - DI(q_i) \ge AR_j(q_i) + 1 \quad (3)$,
$DI(q_{i+1}) - DI(q_i) \ge \max_j\{AR_j(q_i)\} - \min_j\{AL_j(q_{i+1})\} + 1 - k \quad (4)$,
$DI(q_m) = 0 \quad (5)$,
where $m = \lceil (l+1)/2 \rceil$ and $M = \max\{VP(s,T); s\ is\ a\ leaf\ of\ T\}$. □

Here we state a $O(n^2)$-time algorithm which constructs the minimum width placement $\pi(T)$ for a given TSD T under $C_d^{a+}(k)$.

Algorithm Layout-2
Input. A positive integer k and
a TSD $T = (V, E, r, W, D)$ whit n cells
Output. $\pi(T)$: the placement of T.
Method.
(1) Let $M = max\{VP(s,T); s\ is\ a\ leaf\ of\ T\}$.
For each cell p and $j (1 \le j \le M)$, let
$AL_j(p) = 0, AR_j(p) = 0$.
if p is a leaf, then let $DI(p) = 0$.
(2) Traversing the TSD T in postorder, when each cell p is visited,
evaluate $DI(p)$, $AL(p)$ and $AR(p)$ in the following way.

(Case.1) In the case of that a cell p has only one son q, let
$AL_1(p) = \cdots = AL_{D(p)+1}(p) = 0$,
$AL_{D(p)+2} = AL_1(q), AL_{D(p)+3} = AL_2(q), \ldots, AL_M(p) = AL_{M-D(p)-1}(q)$,
$AR_1(p) = \cdots = AR_{D(p)+1}(p) = W(p)$,
$AR_{D(p)+2} = AR_1(q), AR_{D(p)+3} = AR_2(q), \ldots, AR_M(p) = AR_{M-D(p)-1}(q)$,
and $DI(q) = 0$.
(Case.2) In the case of that a cell p has $l (l \geq 2)$ sons $q_1, \ldots, q_l$
$(Index(q_i) = i, 1 \leq i \leq l)$ and $m = \lceil (l+1)/2 \rceil$, firstly let
$DI(q_m) = 0$.
For each i from $i = m-1$ step by -1 until 1, let
$m1_i = \max\{AR_j(q_i) - AL_j(q_{i+1} + 1\}$,
$m2_i = \max\{-AL_j(q_{i+1}) + 1\}$,
$m3_i = \max\{AR_j(q_i) + 1\}$,
$m4_i = m3_i + m2_i - 1 - k$
$(= \max_j\{AR_j(q_i)\} - \min_j\{AL_j(q_{i+1})\} + 1 - k)$, and
$\alpha_i = max\{m1_i, m2_i, m3_i, m4_i\}$, where $j (1 \leq j \leq M)$.
Then let $DI(q_i) = DI(q_{i+1}) - \alpha_i$.
Next for each i from $i = m+1$ step by 1 until l,
computing α_i in the same way, let $DI(q_i) = DI(q_{i-1}) + \alpha_i$.
Finally compute $AL(p)$ and $AR(p)$ as follows.
$AL(p)$ is computed by referring $AL(q_1)$ and $DI(q_1), \ldots, AL(q_l)$ and
$DI(q_l)$ in order so that $AL(p)$ represents the left-hand side outline
of the sub-TSD with the root cell p.
In the similar way,
$AR(p)$ is computed by referring $AR(q_l)$ and $DI(q_l), \ldots, AR(q_1)$
and $DI(q_1)$ in order.
(3) Place the root cell r at the origin, this is, let
$\pi_x(r) = 0, \pi_y(r) = 0$.
Next traversing the TSD T in preorder,
place each cell p, whose father is q, as follows.
$\pi_y(p) = \pi_y(q) + D(q) + 1$,
$\pi_x(p) = \pi_x(q) + DI(p)$.

Lemma 7 *For a given TSD T, the placement $\pi(T)$ which produced by the algorithm Layout-2 satisfies the constraint $C_d^{a+}(k)$.* □

Lemma 8 *For a given TSD T, the placement $\pi(T)$ which produced by the algorithm Layout-2 is a minimum width under the constraint $C_d^{a+}(k)$.* □

Lemma 9 *If the function D is bounded, the algorithm Layout-2 requires $O(n^2)$ time, where n is the number of cells of a given TSD.* □

We summarize these results as:

Theorem 2 *For a given positive integer k, a given TSD T whit n cells and any cell p, if the depth $D(p)$ is bounded, there is an $O(n^2)$ time algorithm which produces the minimum width placement of T under $C_d^{a+}(k)$.* □

If we modify the algorithm Layout-2 by removing the part $m4_i$ of (2), then we have the similar algorithm which satisfies the constraint C_d^{a+}. So we can obtain the following result.

Theorem 3 *For a given TSD T whit n cells and any cell p, if the depth $D(p)$ is bounded, there is an $O(n^2)$ time algorithm which produces the minimum width placement of T under C_d^{a+}.* □

5 Conclusions

We have formalized the drawing problem of tree structured diagrams and introduced several constraints which concern readability of the diagrams. Though the problem of drawing TSD is easier to apply visual programming than that of drawing trees, there are some difficulties such as crossing of a cell and a edge. Problems of minimum width drawings of TSDS are NP-complete under certain sets of constraints[23]. However, we could obtaine the efficient algorithms of minimum width drawings of TSDs under some reasonable sets of constraints. These algorithms will be applied to practical uses such as visual programming and others.

References

[1] Bhasker, J. and Sahni, S., A Linear Algorithm to Find a Rectangular Dual of a Planar Triangulated Graph, Algorithmica, Vol.3, 2(1988), pp.247-278.

[2] Batini, C., Nardelli, E. and Tamassia, R., A layout algorithm for data-flow diagrams, IEEE Trans. Software Eng., Vol.SE-12, No.4, (1986), pp.538-546.

[3] Becker, B. and Hotz, G., On The Optimal Layout of Planar Graphs with Fixed Boundary, SIAM J. Computing, Vol.16, 5(1987), pp.946-972.

[4] Becker, B. and Osthof, H.G., Layout with Wires of Balanced Length, Information and Computation, Vol.73, (1987), pp.45-58.

[5] Chiba, N., Onoguchi, K. and Nishizeki, T., Drawing plane graphs nicely, Acta Informatica, Vol.22, No.2, (1985), pp.187-201.

[6] Eades, P., A Heuristic for Graph Drawing, Cogressus Numerantium, Vol.42, (1984), pp.149-160.

[7] Eades, P. and Wormald, N., Fixed Edge Length Graph Drawing is NP-hard, Discrete Applied Mathematics, Vol.28, (1990), pp.111-134.

[8] Fraysseix, H.D., Pach, J. and Pollack, R., How to Draw a Planar Graph on a Grid, Combinatorica, Vol.10, (1990), pp.41-51.

[9] Fruchterman, T. and Reingold, E., Graph Drawing by Force- Directed Placement, Software-Practice and Experience, Vol.21, 11(1991), pp.1129-1164.

[10] Go, N., Kishimoto, M., Miyadera, Y., Okada, N., Tsuchida, K. and Yaku, T., Generation of Hichart Program Diagrams, Transac. IPSJ, Vol. 31 (1990), 1463 -1473 (in Japanese).

[11] Hope, A.K., A Planar Graph Drawing Program, Software Practice and Experience, Vol.1, (1971), pp.83-91.

[12] Jayakumar, R., Thulasiraman, K. and Swamy, M.N., Planar Embedding: Linear-Time Algorithms for Vertex Placement and Edge Ordering, IEEE Trans. on Circuits and Systems, Vol.35, 3(1988), pp.334-344.

[13] Kamada, T. and Kawai, S., An algorithm for drawing general undirected graphs, Information Processing Letters, Vol.31, No.1, (1989), pp.7-15.

[14] Miyadera, Y., Anzai, K. Banba, H. Tsuchida, K and Yaku, T., Method od Drawing Tree-Structured Program Diagrams on the Euclidean Plane, IEEE COMPSAC'93, (1993), pp.193-201.

[15] Miyadera, Y., Tsuchida, K. and Yaku, T., A Tidy Drawing Problem on the Minimum Area for Tree-Structured Diagrams and its Application to Program Diagrams, TECHNOLOGY AND FOUNDATIONS Information Processing '94(Proceedings of the the IFIP 13th World Computer Congress), Vol.1, North-Holland(1994), pp.282-287.

[16] Ogura K., Go N., Kishimoto M., Miyadera Y., Okada N., Tsuchida K., Unno H. and Yaku T., Generation of the Hichart Program Diagrams, J. Inf. Process., Vol.5, (1992), pp.293-300.

[17] Protsko, L.B., Sorenson, P.G., Tremblay, J.P. and Schaefer, D.A., Towards the Automatic Generation of Software Diagrams, IEEE Trans. Software Eng., vol.17, No.1, (1991), pp.10-21.

[18] Reingold, E.M. and Tilford, J.S., Tider drawings of trees, IEEE Trans. Software Eng., 7(1981), pp.223-228.

[19] Sugiyama, K., Tagawa, T. and Toda, M., Methods for visual understanding of hierarchical system structures, IEEE Trans. Syst., Man, Cybern., Vol. SMC-11, No.2, (1981), pp.109-125.

[20] Tamassia, R., Battista, G. and C.Batini, Automatic graph drawing and readability of diagrams, IEEE Trans. Syst., Man, Cybern., Vol. SMC-18, No.1, (1988), pp.61-79.

[21] Tsuchida, K., The complexity of tidy drawings of trees, Topology and Computer Science(S.Suzuki ed.), Kinokuniya, Tokyo(1987), pp.487-520.

[22] Tsuchida, K., $O(n)$ and $O(n^2)$ time algorithms for the drawing problems of trees, IEICE Trans., D-I, Vol.J76-D-I, 6(1993), pp.237-246(in Japanese).

[23] Tsuchida, K., The complexity of drawing tree-structured diagrams, IEICE Trans. Inf. & Syst., Vol. E78-D, 7(1995), pp.901-908.

[24] Wetherell, C. and Shannon, A., Tidy drawings of trees, IEEE Trans. Software Eng., 5(1979), pp.514-520.

[25] Yaku, T. and Futatsugi, K., Tree structured flowcharts, Inst. Electron. Commun. Engin. Japan, Report, AL-78-47(1978), pp.61-66(Japanese with English abstract).