Parallel h-v Drawings of Binary Trees*

PANAGIOTIS T. METAXAS[1], GRAMMATI E. PANTZIOU[2,3]
and ANTONIS SYMVONIS[4]

[1] Department of Computer Science, Wellesley College, Wellesley MA 02181, USA
[2] Department of Mathematics and Computer Science, Dartmouth College
[3] Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece
[4] Department of Computer Science, University of Sydney, N.S.W. 2006, Australia

Abstract. In this paper we present a method to obtain optimal h-v and inclusion drawings in parallel. Based on parallel tree contraction, our method computes optimal (with respect to a class of cost functions of the enclosing rectangle) drawings in $O(\log^2 n)$ parallel time by using a polynomial number of EREW processors. The method can be extended to compute optimal inclusion layouts in the case where each leaf l of the tree is represented by rectangle $l_x \times l_y$. Our method also yields an NC algorithm for the slicing floorplanning problem. Whether this problem was in NC was an open question [2].

1 Introduction

In this paper we examine drawings of rooted binary trees. We study the h-v drawing convention studied by Crescenzi, Di Battista and Piperno [3] and Eades, Lin and Lin [7]. Our results extend to the inclusion convention [6], and to slicing floorplanning [10, 2].

The drawing of a rooted binary tree using the *h-v drawing convention* is a planar grid drawing in which tree nodes are represented as points (of integer coordinates) in the plane and tree edges as non-overlapping vertical or horizontal line segments. Moreover, each node is placed immediately to the right or immediately below its parent and the drawings of subtrees rooted at nodes with the same parent are non-overlapping. Figure 1 shows three different h-v drawings of the same tree. Different h-v drawings of the same tree can be of different quality. The *quality* (or *cost*) is a function of the drawing. The most commonly used cost function is the area of the enclosing rectangle of the drawing.

Eades et al. [7] showed how to compute in $O(n^2)$ time an optimal h-v drawing of a tree with n nodes with respect to a cost function $\psi(w, h)$ which is nondecreasing in both parameters w and h, where w and h are the width and the height of the enclosing rectangle of the drawing, respectively. The same method can be used to develop optimal drawings (with respect to some cost function ψ) when the inclusion convention is adopted. In the *inclusion* convention, a node is represented by a rectangle and the parent-child relation by enclosing the rectangle which represents the child within that of the parent. Moreover, rectangles of

* The work of the second author is partially supported by the EEC ESPRIT Basic Research Action No. 7141 (ALCOM II) and by the NSF grant No. CDA-9211155. Email: pmetaxas@lucy.wellesley.edu, pantziou@cs.dartmouth.edu, symvonis@cs.su.oz.au

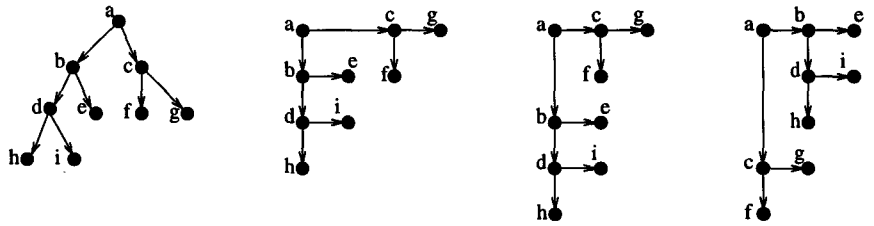

Fig. 1. Examples of h-v drawings.

nodes with the same parent are non-overlapping, next to each other (same X or Y coordinate of the top left corner) and in at least distance δ from each other. The rectangles representing the leaves are assumed to have sidelengths which are multiples of the "unit" of length.

A closely related problem is that of slicing floorplanning. In *slicing floorplanning* we are given a regular binary tree (called *slicing tree*) in which leaves represent rectangular modules that are to be placed in the plane. Each internal node is an H or V node. In the final drawing, the modules are drawn as rectangles of the corresponding size but with a choice on their orientation (i.e., $x \times y$ or $y \times x$) and internal nodes by horizontal or vertical line segments. The drawing of the subtrees rooted at the children of a V-node (H-node) are drawn next to (on top of) each other. Alternatively, the drawing of the subtree rooted at a V-node (H-node) consists of two vertical (horizontal) slices, each next to (on top of) each other and containing the drawings of the subtrees rooted at its children. An $O(n^2)$-time sequential algorithm for the slicing floorplaning problem was given by Stockmeyer [10]. Chen and Tollis [2] derived a parallel algorithm that needs $O(n)$ time and $O(n)$ processors, where n is the number of leaves of the slicing tree.

In this paper, we present a method that derives optimal drawings of rooted binary trees in parallel for both the h-v and the inclusion conventions. We choose to present our method for the h-v drawing convention since the drawings are still trees in their familiar "conventional" form. Our method determines an optimal h-v drawing of a tree of n nodes in $O(\log^2 n)$ parallel time using $O(n^6/\log n)$ EREW processors. Even though the number of processors is too high for the method to be of practical interest, our work places the problem in the class NC. In the case that we want to minimize the area of the drawing, we can reduce the number of processors to $O(n^4 \log n)$.

Applying our method to the slicing floorplanning problem yields an $O(\log^2 n)$ PRAM algorithm which uses $O(L^6/\log n)$ processors, where L is the sidelength of the floorplan of maximum enclosing rectangle. This proves that when $L = O(n^c)$, for some constant $c \geq 1$, the slicing floorplanning problem is in NC. The assumption $L = O(n^c)$ is a reasonable one since, otherwise, the layout would be of superpolynomial size.

Because of space limitations, all proofs are omitted and can be found in the full paper [9].

2 Preliminaries

For definitions related to *drawings* (or *layouts*) the reader is referred to [5]. In this paper, we will study orthogonal straight-line planar grid drawings of rooted binary trees. Unless otherwise specified we will refer to them simply as drawings.

Given a graph $G = (V, E)$ and a drawing Δ of G on the plane, the drawing of any subgraph H of G resulting from Δ is called a *partial drawing* of H (with respect to Δ). The *enclosing rectangle* of a drawing is the smallest rectangle with sides parallel to the axes which contains all points of the drawing. Let $X_{\max} = \max_{v \in V}\{x_v\}$, $X_{\min} = \min_{v \in V}\{x_v\}$, $Y_{\max} = \max_{v \in V}\{y_v\}$, $Y_{\min} = \min_{v \in V}\{y_v\}$. $X_{\max}$, $X_{\min}$, $Y_{\max}$ and $Y_{\min}$ completely define the enclosing rectangle of a drawing. Two rectangles are *overlapping* if they share at least a point of the plane. Otherwise, they are *non-overlapping* or *disjoint*. The *width* of a drawing is equal to $X_{\max} - X_{\min}$ while its *height* is equal to $Y_{\max} - Y_{\min}$. A drawing is *reduced* if (1) for all integers i such that $X_{\min} \leq i \leq X_{\max}$ theres exists node $v \in V$ with $x_v = i$; and (2) for all integers i such that $Y_{\min} \leq i \leq Y_{\max}$ theres exists node $v \in V$ with $y_v = i$. In the rest of the paper we will assume only reduced drawings.

A *rooted tree* $T = (V, E)$ is a weakly connected directed graph in which all nodes but the root are of indegree 1. We will use the notation $|T|$ to denote the number of nodes of tree T. The *subtree rooted at* v, denoted T_v, consists of v, all of v's descendants and the edges between them. A *partial tree* is a weakly connected subgraph of a rooted tree.

The enclosing rectangle of a drawing can be completely described by its width, height and the coordinates of one of its corners, say the left-top one. Most times, during the description of our algorithm, we will assume that the left-top corner of the enclosing rectangle has coordinates $(0, 0)$. By fixing a point of reference, it is sufficient to describe a rectangle R by a pair of two integers, its width and height, i.e., $R = (w, h)$, $w \geq 0$, $h \geq 0$. Two rectangles are called *equal* if they have identical width and height.

Given two rectangles $R_1 = (w_1, h_1)$ and $R_2 = (w_2, h_2)$ we say that rectangle R_1 *dominates* (or *fits in*) R_2 if $w_1 \leq w_2$ and $h_1 \leq h_2$. Given a set S of rectangles, an *atom* is an element of S which dominates no other rectangle in S. Any set of atoms that are sorted in increasing order with respect to their widths, are also sorted in decreasing order with respect to their heights.

With each drawing we associate a cost. Our objective is to derive drawings of minimum cost. In this paper, the cost function will be a function of the enclosing rectangle of the drawing, i.e., a function $\psi : \Re^2 \longrightarrow \Re$. Our results will hold for any function that is nondecreasing in both parameters, i.e., $\psi(x_1, y_1) \geq \psi(x_2, y_2)$ whenever $x_1 \geq x_2$ and $y_1 \geq y_2$. Let $R = (width, height)$. The following are commonly used cost functions that are non-decreasing in both parameters: $area(R) = width \cdot height$, $minimum\_enclosing\_square(R) = \max(width, height)$, $perimeter(R) = 2(width + height)$.

The problem of *minimum size h-v drawing of a binary rooted tree* T is the problem of determining an h-v drawing of T of minimum cost with respect to some cost function ψ.

3 The Parallel Algorithm for h-v Drawings

The algorithm for finding a minimum size h-v drawing of a binary tree $T = (V, E)$ is based on the parallel tree contraction technique [1]. Within a logarithmic number of phases, the parallel tree-contraction algorithm contracts a tree T to its root by processing a logarithmic number of intermediate binary trees $T(i) = (V(i), E(i))$, $i = 0, 1, \cdots k$, with $k = O(\log |T|)$. Note that the algorithm starts off with $T(0) = T$, and proceeds by contracting tree $T(i-1)$ to tree $T(i)$ of $|T(i)| \leq \varepsilon |T(i-1)|$ nodes, $0 < \varepsilon < 1$. At the end, $T(k)$ contains only one node. During the ith phase of the algorithm, the tree $T(i)$ is obtained from $T(i-1)$ by applying a local operation, called *shunt*, to a subset of the leaves of $T(i-1)$. The shunt operation is composed of two steps: In the first step, a subset of the tree leaves are removed (an operation called *pruning*) and their parent (or sibling) nodes are updated to reflect this fact. In the second step, each of these parents is removed (by an operation called *shortcutting*), and its child is updated. The shunt operation removes roughly half of the tree nodes, so ε is roughly $1/2$. To use the tree contraction technique, one has to describe the updates that take place during the shunt operation. Before we do that, we give some notation and describe the information associated with each node of the tree.

3.1 Data Structures and Useful Operations

If $v \in V(i)$ then let $T_v^{c_i}$ be the partial tree (of T) contracted to v after applying the shunt operation to siblings of v during the first i phases of the parallel tree-contraction algorithm.

With each node u in $V(i)$ we associate a tuple L_u containing the following information: The root r_u of the partial tree $T_u^{c_i}$ that has been contracted to u, and a set R_u that keeps information for all *useful* drawings of $T_u^{c_i}$ (the notion of a useful drawing will be defined shortly). Each element of R_u corresponds to a specific reduced *partial drawing* π and consists of 3 tuples, i.e., $\pi = ((W_u, H_u), (A_u, B_u), (x_u, y_u))$. The first tuple, i.e., (W_u, H_u), describes the width and the height of the enclosing rectangle of π. The second tuple, i.e., (A_u, B_u) describes the width A_u and the height B_u of the largest rectangle (having u at its top left corner) that can be included in the partial drawing π such that the enclosing rectangle (W_u, H_u) of π remains unchanged and π is still a valid h-v drawing of $T_u^{c_i}$. We shall refer to (A_u, B_u) as the *empty rectangle* corresponding to R_u. Finally, the third tuple, i.e., (x_u, y_u), is the location of u in π, where $(0, 0)$ is the coordinate of the top left corner of any partial drawing. Note that, u is a leaf in the partial tree $T_u^{c_i}$. For each $u \in V = V_0$, we initialize $L_u = \langle u;\ ((0,0),(0,0),(0,0))\rangle$.

Suppose that at some phase of the parallel tree-contraction algorithm we want to include the partial drawing π of a partial tree rooted at a node v in a partial drawing π' of another partial tree whose v is a leaf. Then, we need to know the position of v in π' as well as how much the inclusion of π in π' will change the rectangle corresponding to π'. Thus, all the parameters associated above with each node of each $T(i)$, $i = 0, \cdots, \log |T|$, are necessary for the

parallel tree-contraction approach to work. In Section 3.3, we shall show that the information kept by those parameters is all that is needed for the algorithm to work.

Let $\pi^1 = ((W_u^1, H_u^1), (A_u^1, B_u^1), (x_u^1, y_u^1))$ and $\pi^2 = ((W_u^2, H_u^2), (A_u^2, B_u^2), (x_u^2, y_u^2))$ be two partial drawings of $T_u^{c_i}$.

Definition 1. We say that (partial) drawing π^1 *dominates* (or *fits in*) (partial) drawing π^2 if the enclosing rectangle of π^1 *fits in* the enclosing rectangle of π^2.

Definition 2. Partial drawing π^1 *prevails* partial drawing π^2 with respect to integer $\lambda > 0$ if π^1 *fits in* π^2 and at least one of the following conditions is satisfied: a) (A_u^2, B_u^2) fits in (A_u^1, B_u^1); b) $A_u^2 \leq A_u^1$ *and* $B_u^2 > B_u^1 \geq \lambda$; c) $A_u^2 > A_u^1 \geq \lambda$ *and* $B_u^2 \leq B_u^1$; d) $A_u^2 > A_u^1 \geq \lambda$ *and* $B_u^2 > B_u^1 \geq \lambda$.

In this paper, when using the notion of "prevail", λ will usually be the size (i.e., the number of nodes) of a partial tree.

Definition 3. Let $u \in V(i)$, $T_u^{c_i}$ be the partial tree (of T) contracted to u during the first i phases of the parallel tree contraction algorithm, T_u be the subtree of T rooted at u and R a set of partial drawings of $T_u^{c_i}$. A partial drawing π in R is called *useful* if π is prevailed, with respect to the size $|T_u|$ of tree T_u, by no other partial drawing in R.

Lemma 4. *Let $u \in V(i)$, $T_u^{c_i}$ be the partial tree (of T) contracted to u during the first i phases of the parallel tree contraction algorithm and T_u be the subtree of T rooted at u. Then the number of useful partial drawings in L_u is at most $O(min(|T_u|, |T_u^{c_i}|) \cdot |T_u^{c_i}|^2)$.*

3.2 The Shunt Updates

Let l be a leaf in tree $T(i-1)$, s be l's sibling, f be l's parent and p be f's parent. Let also $L_f = \langle r_f; R_f \rangle$, where $R_f = \{((W_f^1, H_f^1), (A_f^1, B_f^1), (x_f^1, y_f^1)), \cdots, ((W_f^i, H_f^i), (A_f^i, B_f^i), (x_f^i, y_f^i))\}$, $L_s = \langle r_s; R_s \rangle$, where $R_s = \{((W_s^1, H_s^1), (A_s^1, B_s^1), (x_s^1, y_s^1)), \cdots, ((W_s^j, H_s^j), (A_s^j, B_s^j), (x_s^j, y_s^j))\}$ and $L_l = \langle r_l; R_l \rangle$, where $R_l = \{((W_l^1, H_l^1), (\cdot,\cdot), (\cdot,\cdot)), \cdots, ((W_l^k, H_l^k), (\cdot,\cdot), (\cdot,\cdot))\}$, be the information associated with f, s and l respectively[5].

Recall that R_l, R_f and R_s keep information for all useful partial drawings of $T_l^{c_{i-1}}$, $T_s^{c_{i-1}}$, and $T_f^{c_{i-1}}$ respectively. Note that since l is a leaf of T, $T_{r_l} = T_l^{c_{i-1}}$. Note also that r_l and r_s are the children of f in the tree $T = T(0)$.

During the ith phase of the tree contraction algorithm, we apply the shunt operation to a set of leaves of $V(i-1)$. The shunt operation to a leaf l of $V(i-1)$ consists of two stages, namely, a *pruning* and a *shortcutting* stage.

[5] Because we deal with the drawing of subtree T_{r_l} rather than a partial tree, we do not have to record any information about an empty rectangle. Thus, the notation $((W_l, H_l), (\cdot,\cdot), (\cdot,\cdot))$.

The Pruning Stage. In the pruning stage, we use the tuples L_l and L_s to construct the tuple $L_{f'} = \langle f; R_{f'} \rangle$ containing information about all useful (with respect to the size $|T_s|$ of T_s) partial drawings of the partial tree rooted at f and including the subtree T_{r_l} and the partial tree $T_s^{c_{i-1}}$.

Let $\pi_s = ((W_s, H_s), (A_s, B_s), (x_s, y_s))$ be an element of the set R_s and $\pi_l = ((W_l, H_l), (\cdot, \cdot), (\cdot, \cdot))$ be an element of the set R_l. There are essentially 4 ways to arrange T_{r_l} and $T_u^{c_{i-1}}$. For each one we compute the new drawing. In what follows, the superscripts in $(x_s^{f'}, y_s^{f'})$ are used to avoid confusion with (x_s, y_s). $(x_s^{f'}, y_s^{f'})$ are the coordinates of s in the drawing of the partial tree rooted at f and including the subtree T_{r_l} and $T_s^{c_{i-1}}$ while, (x_s, y_s) are the coordinates of s in the partial drawing of $T_s^{c_{i-1}}$.

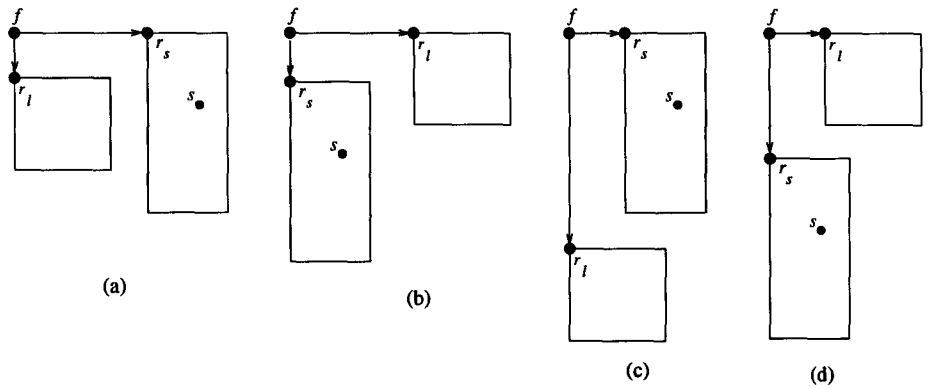

Fig. 2. The cases which occur when combining the subtree T_{r_l} together with the partial tree $T_s^{c_{i-1}}$ during the pruning phase of the shunt operation.

Case 1: The situation in case 1 is described in Figure 2(a). The produced partial drawing $((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_s^{f'}, y_s^{f'}))$ is defined by:

$$W_{f'} = W_l + W_s + 1 \qquad A_{f'} = A_s \qquad x_s^{f'} = x_s + W_l + 1$$
$$H_{f'} = \max(H_l + 1, H_s) \qquad B_{f'} = B_s + \max(0, H_l + 1 - H_s) \qquad y_s^{f'} = y_s$$

The correctness of the computed values for $W_{f'}, H_{f'}, A_{f'}, x_s^{f'}, y_s^{f'}$ is obvious from Figure 2(a). For $B_{f'}$, note that we simply extent the height of the empty rectangle up to the bounds of the enclosing rectangle.

Case 2: The situation in case 2 is described in Figure 2(b). The produced partial drawing $((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_s^{f'}, y_s^{f'}))$ is defined by:

$$W_{f'} = W_l + W_s + 1 \qquad A_{f'} = A_s \qquad x_s^{f'} = x_s$$
$$H_{f'} = \max(H_s + 1, H_l) \qquad B_{f'} = B_s + \max(0, H_l - H_s - 1) \qquad y_s^{f'} = y_s + 1$$

Case 3: The situation in case 3 is described in Figure 2(c). The produced partial drawing $((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_s^{f'}, y_s^{f'}))$ is defined by:

$$W_{f'} = \max(W_s + 1, W_l) \qquad A_{f'} = A_s + \max(0, W_l - W_s - 1) \qquad x_s^{f'} = x_s + 1$$
$$H_{f'} = H_s + H_l + 1 \qquad B_{f'} = B_s \qquad y_s^{f'} = y_s$$

Case 4: The situation in case 4 is described in Figure 2(d). The produced partial drawing $((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_s^{f'}, y_s^{f'}))$ is defined by:

$W_{f'} = \max(W_l + 1, W_s)$ $\quad A_{f'} = A_s + \max(0, W_l + 1 - W_s)$ $\quad x_s^{f'} = x_s$

$H_{f'} = H_l + H_s + 1$ $\quad B_{f'} = B_s$ $\quad y_s^{f'} = y_s + H_l + 1$

Note that in all of the above cases the size of the empty rectangle is extended up to the bounds of the enclosing rectangle.

It is possible that the sibling s of leaf l does not exist. This can happen when tree T is not a regular binary tree. This leads to the following additional cases:

Case 5: The produced drawing $((W_{f'}, H_{f'}), (\cdot,\cdot), (\cdot,\cdot))$ of T_{r_f} is defined by:

$$W_{f'} = W_l \qquad H_{f'} = H_l + 1$$

Case 6: The produced drawing $((W_{f'}, H_{f'}), (\cdot,\cdot), (\cdot,\cdot))$ of T_{r_f} is defined by:

$$W_{f'} = W_s + 1 \qquad H_{f'} = H_s$$

After computing all the possible partial drawings, the prevail operation (with respect to the size $|T_s|$ of T_s) is applied to them and the set $R_{f'}$ is obtained. I.e., all the drawings that are prevailed by other drawings are eliminated. (For details regarding the parallel implementation of the prevail operation see the full paper [9].)

The Shortcutting Stage. In the shortcutting stage, the tuples $L_{f'}$ and L_f are used to construct the new tuple for L_s. The root of the new L_s will be r_f. To determine an element π of the new set R_s we combine drawings $\pi^{f'} = ((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_s^{f'}, y_s^{f'}))$ of $R_{f'}$ with drawing $\pi^f = ((W_f, H_f), (A_f, B_f), (x_f, y_f))$ of R_f. In simple words, we embed $\pi^{f'}$ into π^f. The new element $\pi = ((W_s, H_s), (A_s, B_s), (x_s, y_s))$ of R_s produced by embedding $\pi^{f'}$ into π^f is computed as follows:

$W_s = W_f + \max(0, W_{f'} - A_f)$ $\quad A_s = A_{f'} + \max(0, A_f - W_{f'})$ $\quad x_s = x_f + x_s^{f'}$

$H_s = H_f + \max(0, H_{f'} - B_f)$ $\quad B_s = B_{f'} + \max(0, B_f - H_{f'})$ $\quad y_s = y_f + y_s^{f'}$

The correctness of the computed values for W_s and H_s is obvious. For A_s and B_s, note that in the case that $A_f > W_{f'}$ and/or $B_f > H_{f'}$, A_s equals $A_{f'} + A_f - W_{f'}$ and/or $B_s = B_{f'} + B_f - H_{f'}$. I.e., the empty rectangle of π is extended to be as large as the difference between the sizes of the enclosing rectangle of $\pi^{f'}$ and the empty rectangle of π^f allow.

Finally, the prevail operation (with respect to the size $|T_s|$ of T_s) is applied to the computing drawings and the new set R_s is obtained.

3.3 Correctness and Analysis

To prove the correctness of the algorithm we have to establish that during the course of the algorithm we keep all the necessary information. To do that we have to prove two things: firstly, that it is enough to keep only one drawing out of those that differ only in the coordinates of the "interface to bellow", i.e., they have the same enclosing and "empty" rectangles and, secondly, that we can safely use the prevail operation to shorten our R-sets after each stage.

Lemma 5. *Consider two partial drawings* $\pi 1_s^{i-1} = ((W_s, H_s), (A_s, B_s), (x1_s, y1_s))$ *and* $\pi 2_s^{i-1} = ((W_s, H_s), (A_s, B_s), (x2_s, y2_s))$ *of* R_s *that differ only in the coordinates of* s *in the drawing. Also assume a drawing* $\pi_l^{i-1} = ((W_l, H_l), (\cdot, \cdot), (\cdot\cdot))$ *of* R_l *and a partial drawing* $\pi_f^{i-1} = ((W_f, H_f), (A_f, B_f), (x_f, y_f))$ *of* R_f. *Let* $\pi 1_s^i$ *be the partial drawing obtained by combining* $\pi 1_s^{i-1}$, π_l^{i-1} *and* π_f^{i-1} *during phase* i. *Let* $\pi 2_s^i$ *be the partial drawing obtained by combining* $\pi 2_s^{i-1}$, π_l^{i-1} *and* π_f^{i-1} *during phase* i. *Then,* $\pi 1_s^i$ *and* $\pi 2_s^i$ *have the same enclosing and empty rectangles.*

Lemma 6. *We can "safely" eliminate prevailed elements from* R_s *after the end of each shortcutting stage as well as after the end of each pruning stage.*

Theorem 7. *A minimum size h-v drawing of a binary tree with* n *nodes can be correctly found in* $O(\log^2 n)$ *parallel time using* $O(n^6 / \log n)$ *EREW processors.*

3.4 Minimum Area h-v Drawings

When we are after minimum area h-v drawings, the number of processors required by the parallel algorithm can be substantially reduced. To achieve that, we used the following result due to Crescenzi, Di Battista and Piperno: *For any binary tree* T *of* n *nodes, there exists an h-v drawing of* T *with at most* $n(\log n + 1)$ *area. Moreover, the width of the layout is at most* $\log n + 1$ *while its height is at most* n. Based on this result we can show:

Theorem 8. *A minimum area h-v drawing of a binary tree with* n *nodes can be correctly found in* $O(\log^2 n)$ *parallel time using* $O(n^4 \log n)$ *EREW processors.*

Theorem 9. *A h-v drawing of area at most* $n(\log n + 1)$ *of a binary tree with* n *nodes can be correctly found in* $O(\log^2 n)$ *time using* $O(n^2 \log^3 n)$ *EREW processors.*

4 Related Problems

Inclusion Drawings of Binary Trees. The method used to compute minimum size h-v drawings can be also used to compute inclusion drawings.

Theorem 10. *A minimum size inclusion drawing of a binary tree with* n *nodes can be found in* $O(\log^2 n)$ *parallel time using* $O(((\delta + \max(a, b))n)^6 / \log n)$ *EREW processors.*

In the case that the rectangle associated with leaf l is of dimensions $l_x \times l_y$ rather than $a \times b$, the number of required processors increases[6]. Let $L = \sum_{\text{leaves } l} \max(l_x, l_y)$. Then, as a corollary of Theorem 10 we get:

[6] Again, l_x and l_y are supposed to be multiples of the "unit" of length.

Corollary 11. *A minimum size inclusion drawing of a binary tree with n nodes in which each leaf l is associated with a rectangle of size $l_x \times l_y$ can be found in $O(\log^2 n)$ parallel time using $O(\delta n + L)^6 / \log n)$ EREW processors, where $L = \sum_{leaves\ l} \max(l_x, l_y)$*

Slicing Floorplanning. We can apply our method to get a parallel solution for the slicing floorplanning problem. We set δ to 0 while, for the pruning stage, we compute partial floorplans based on whether the parent of the leaf is an H or V node. Initially, the list of partial floorplans of each leaf of the slicing tree contains two partial floorplans, one for each orientation of the leaf's module.

Let the module associated with leaf l of the slicing tree have dimensions $l_x \times l_y$, $l_x \leq l_y$. Let $L = \sum_{\text{leaves } l} l_y$. From Corollary 11 we get:

Theorem 12. *A minimum size slicing floorplan of a slicing tree with n leaves can be found in $O(\log^2 n)$ parallel time using $O(L^6 / logn)$ EREW processors.*

Thus, for slicing floorplans which can be drawn in polynomial area, the problem of determining an optimal area slicing floorplan is placed in the class NC.

References

1. Abrahamson, K., Dadoun, N., Kirkpatrick, D. and Przytycka, T. A Simple Parallel Tree Contraction Algorithm. J. of Algorithms, 10(1989), pp. 287–302.
2. Chen, C-H and Tollis, I. Parallel Algorithms for Slicing Floorplan Designs. In Proc. of SPDP '90, Dec. 1990, pp. 279–282.
3. Crescenzi, P., Di Battista, G. and Piperno, A. A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees. Computational Geometry: Theory and Applications, Vol. 2, pp. 187–200, 1992.
4. Di Battista, G. and Tamassia, R. Algorithms for Plane Representation of Acyclic Digraphs. Theoretical Computer Science, Vol. 61, pp. 175–198, 1988.
5. Di Battista, G., Eades, P., Tamassia, R. and Tollis, I.G. Algorithms for Drawing Graphs: an Annotated Bibliography. Tech. Report, Brown Univ., June 1993.
6. Eades, P., Lin, T., and Lin, X. Two Tree Drawing Conventions. TR 174, Key Centre for Software Technology, Dept. of Computer Science, The University of Queensland, 1990. (To appear in Computational Geometry and Applications)
7. Eades, P., Lin, T., and Lin, X. Minimum Size h-v Drawings, Advanced Visual Interfaces (Proceedings of AVI 92), World Series in Computer Science Vol. 36, pp. 386–394, 1992.
8. Garg, A., Goodrich, M.T., and Tamassia, R. Area-Efficient Upward Tree Drawings, 9th Annual Symposium on Computational Geometry, pp. 359–368, 1992.
9. Metaxas, P., Pantziou, G., Symvonis A. Parallel h-v Drawings of Binary Trees. Technical Report 480, Basser Dept. of Computer Science, Uninversity of Sydney, March 1994.
10. Stockmeyer, L. Optimal Orientations of Cells in Slicing Floorplan Designs. Information and Control **57**, pp. 91–101, 1983.