4-21-1996

A Note on Parallel Algorithms for Optional h-v Drawings of Binary Trees

P. Takis Metaxas
Wellesley College, pmetaxas@wellesley.edu

Grammati E. Pantziou
Computer Technology Institute (Greece)

Antonis Symvonis
University of Sydney

Recommended Citation
A Note on Parallel Algorithms for h-v Drawing of Binary Trees, with G.E. Pantziou and A. Symvonis. In Computational Geometry: Theory and Applications, 9 145-158 (1998).

A Note on Parallel Algorithms for Optimal h-v Drawings of Binary Trees

Panagiotis T. Metaxas
Department of Computer Science
Wellesley College
Wellesley, MA 02181-8289, USA
pmetaxas@lucy.wellesley.edu

Grammati E. Pantziou*
Computer Technology Institute
P.O. Box 1122, 26110 Patras, Greece
pantziou@cti.gr

Antonis Symvonis
Basser Department of Computer Science
University of Sydney,
N.S.W. 2006, Australia
symvonis@cs.su.oz.au

Revised: 21 April 1996

Abstract

In this paper we present a method to obtain optimal h-v 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 number of processors reduces substantially when we study minimum area drawings. Our work places the problem of obtaining optimal size h-v drawings in NC, presenting the first algorithm with polylogarithmic time complexity.

Keywords: h-v drawing, parallel algorithm, parallel tree contraction, tree layout.

*The work of this author was partially supported by the ESPRIT Basic Research Project GEPPCOM (contract no. 9072).

1 Introduction

Drawing trees in a way that facilitates the understanding of the properties of the object being drawn is part of extensive research in the areas of visualisation, computational geometry and documentation systems. In particular, rooted trees have been used to represent family trees, hierarchical structures and search trees. (For a survey of graph drawing algorithms, including algorithms for drawing trees, see [5].) In this paper we study h-v drawings of binary trees. h-v drawings were previously examined by Eades, Lin and Lin [7] and Crescenzi, Di Battista and Piperno [4]. Our results extend to inclusion drawings [6], and to slicing floorplanning [10, 3].

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 (same Y-coordinate) or immediately below (same X-coordinate) 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.

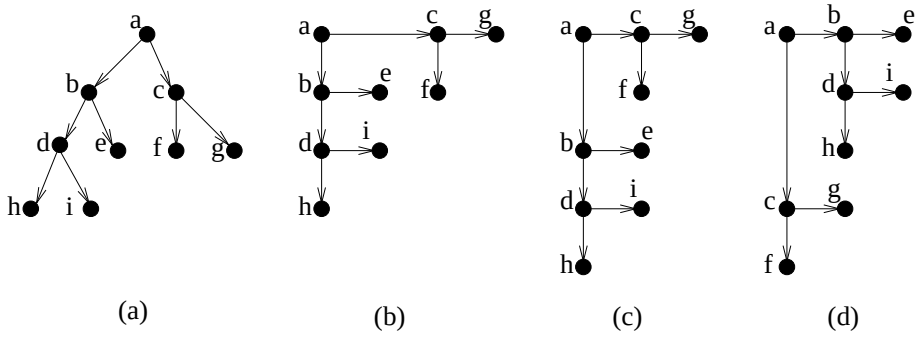

Figure 1: A binary tree and three of its h-v drawings.

As it is evident, 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. Other cost functions can be defined. 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.

In this paper, we present a parallel method that derives optimal h-v drawings of binary trees. 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 high, our method places the problem of obtaining optimal h-v drawings as well as other related problems in NC. By realizing that h-v drawings can be easily converted to *upward* drawings, our method can be used for deriving (non-optimal) upward drawings of binary trees in parallel. In the case that we want to minimise the area of the drawing, by using the fact that for a tree with n nodes there exist upwards layouts of area $n(\log n + 1)$ [4], the

required number of processors reduces to $O(n^4 \log n)$.

The rest of the paper is organised as follows: In Section 2, we give the necessary terminology. In Section 3, we present our parallel algorithm for deriving optimal drawings for binary trees using the h-v convention. Minimum area h-v drawings are also considered. We conclude in Section 4 with open problems and by discussing the application of our method to related problems. A preliminary version of this paper has appeared in [8].

2 Preliminaries

Consider a binary tree $T = (V, E)$ (i.e., a weakly connected directed graph in which all nodes but the root are of in-degree 1 and of out-degree 0, 1 or 2). We 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 connected subgraph of a rooted tree. Note the difference between a subtree and a partial tree.

A *drawing* Δ of a graph $G = (V, E)$ maps each node $v \in V$ to a point $P_v = (x_v, y_v)$ on the plane and each edge $(u, v) \in E$ to a simple Jordan curve with endpoints P_u and P_v. If all edges are mapped to straight-line segments, we have a *straight-line drawing*. Moreover, if all edges are line segments parallel to the X or Y axis, the drawing is called an *orthogonal straight-line drawing* (or a *rectilinear drawing*). If all the nodes of G are mapped to points with integer coordinates, we have a *grid* drawing. When edges intersect only at common endpoints, the drawing is called *planar*. In this paper, we study orthogonal straight-line planar grid drawings of rooted binary trees (for simplicity, referred 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*. 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 $X_{\min} \leq i \leq X_{\max}$ there exists node $v \in V$ with $x_v = i$, and
2. for all integers i such that $Y_{\min} \leq i \leq Y_{\max}$ there exists node $v \in V$ with $y_v = i$.

In the rest of the paper we assume only reduced drawings.

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. During the description of our algorithm, we 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:

$$w_1 \leq w_2 \quad \text{and} \quad h_1 \leq h_2.$$

Given a set S of rectangles, an *atom* is an element of S which is dominated by 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 $\psi : N^2 \longrightarrow N$ is defined on the enclosing rectangle of the drawing. Our results 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:

1. $area(R) = width \cdot height$
2. $perimeter(R) = 2(width + height)$
3. $minimum\_enclosing\_square(R) = \max(width, height)$
4. $height\_for\_a\_given\_width(R, w) = \begin{cases} height & \text{if } width \leq w \\ \infty & \text{otherwise} \end{cases}$

An *h-v drawing* of a binary tree is an orthogonal straight-line planar grid drawing which also satisfies the following restrictions:

1. Any tree node v is drawn at the left-top corner of the enclosing rectangle in the partial drawing of the subtree rooted at v.
2. The enclosing rectangles of the partial drawings of the subtrees rooted at sibling nodes are non-overlapping.

The problem of *minimum size h-v drawing of a binary tree* T is the problem of determining an h-v drawing of T of minimum cost with respect to some cost function ψ. Figure 2 shows two h-v drawings of the same tree. The left drawing of Figure 2 is of minimum area while the right drawing is of minimum enclosing square. Both drawings are of minimum perimeter.

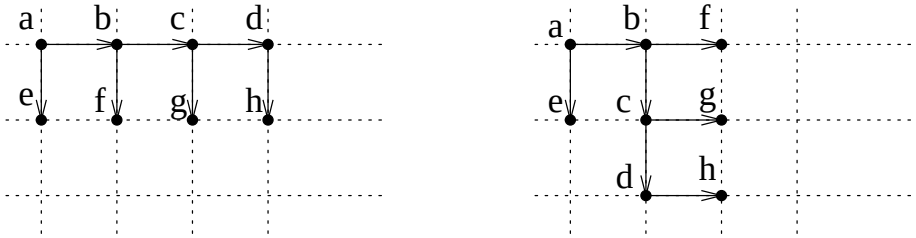

Figure 2: Examples of minimum size h-v drawings.

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 (see, e.g., [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*, see Figure 3) 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 simultaneously 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

If $u \in V(i)$ then let T_u^i be the partial tree of T contracted to u after applying the shunt operation to siblings of u 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^i that has been contracted to u, and a set R_u that keeps information for all drawings of T_u^i. Each element of R_u corresponds to a specific reduced *partial drawing* π and consists of 3 tuples, $\pi = ((W_u, H_u),\ (A_u, B_u),\ (x_u, y_u))$. The first tuple, (W_u, H_u), describes the width and the height of the enclosing rectangle of π. The second tuple, (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^i. We refer to (A_u, B_u) as the *empty rectangle* corresponding to R_u. Finally, the third tuple, (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^i. For each $u \in V = V_0$, we initialise $L_u =< u;\ ((0,0),(0,0),(0,0)) >$.

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.

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. Recall that R_l, R_f and R_s keep information for all partial drawings of T_l^{i-1}, T_s^{i-1}, and T_f^{i-1} respectively. Note that r_l and r_s are the children of f in the tree $T = T(0)$, and that $T_{r_l} = T_l^{i-1}$ because l is a leaf of $T(i-1)$. Note also that since we deal with the drawing of subtree T_{r_l}, we do not have to record any information about an empty rectangle. Thus, the notation $((W_l, H_l), (\cdot, \cdot), (\cdot, \cdot))$.

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 (Figure 3).

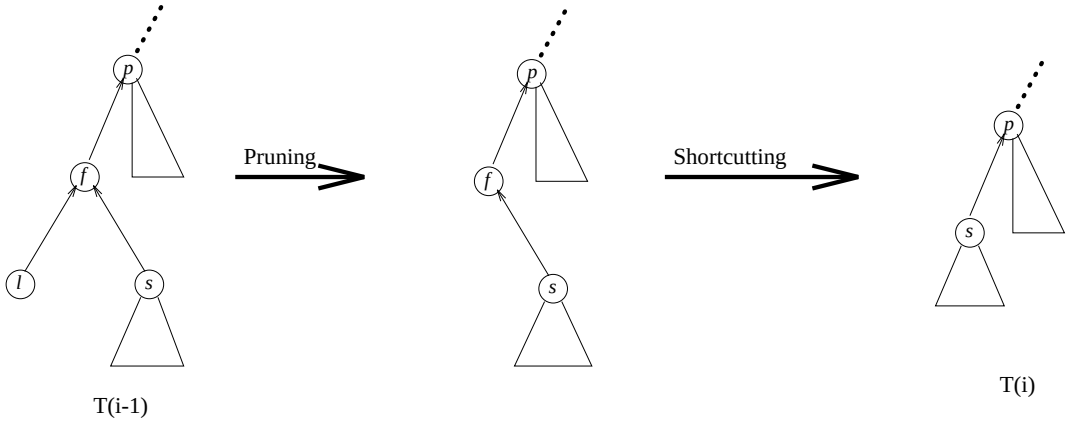

Figure 3: Application of the shunt operation to leaf l.

3.2.1 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 partial drawings of the partial tree which is rooted at f and includes the subtree T_{r_l} and the partial tree T_s^{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_s^{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^{i-1} while, (x_s, y_s) are the coordinates of s in the partial drawing of T_s^{i-1}.

Case 1: The situation in this case is described in Figure 4(a). The produced partial drawing $((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_s^{f'}, y_s^{f'}))$ is defined by:

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

The correctness of the computed values for $W_{f'}, H_{f'}, A_{f'}, x_s^{f'}, y_s^{f'}$ can be readily verified from Figure 4(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 this case is described in Figure 4(b). The produced partial drawing $((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_s^{f'}, y_s^{f'}))$ is defined by:

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

Case 3: The situation in this case is described in Figure 4(c). The produced partial drawing $((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_s^{f'}, y_s^{f'}))$ is defined by:

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

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

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

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 situation in this case is described in Figure 4(e). The produced drawing $((W_{f'}, H_{f'}), (\cdot, \cdot), (\cdot, \cdot))$ of subtree T_{r_f} is defined by:

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

Case 6: The situation in this case is described in Figure 4(f). The produced drawing $((W_{f'}, H_{f'}), (\cdot, \cdot), (\cdot, \cdot))$ of subtree T_{r_f} is defined by:

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

3.2.2 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 $\pi^{f'}$

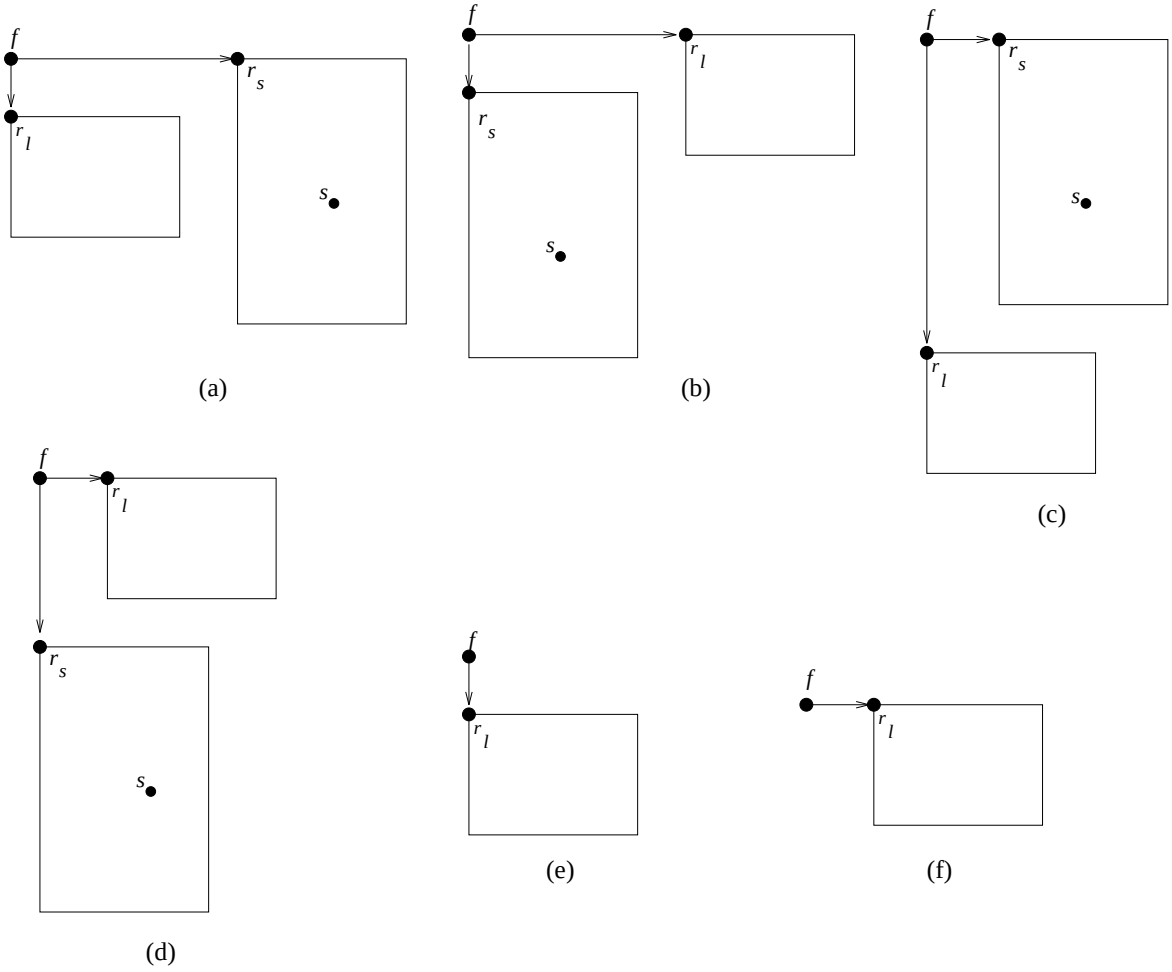

Figure 4: The cases which occur when combining the subtree T_{r_l} together with the partial tree T_s^{i-1} during the pruning phase of the shunt operation.

into π^f is computed as follows:

$$\begin{array}{ll} W_s := W_f + \max(0, W_{f'} - A_f) & H_s := H_f + \max(0, H_{f'} - B_f) \\ A_s := A_{f'} + \max(0, A_f - W_{f'}) & B_s := B_{f'} + \max(0, B_f - H_{f'}) \\ x_s := x_f + x_s^{f'} & y_s := y_f + y_s^{f'} \end{array} \quad (1)$$

The correctness of the computed values for W_s and H_s can be readily verified. For A_s and B_s, note that in the case where $A_f > W_{f'}$ 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 allows.

3.3 Analysis

In the pruning stage, we use the tuples $L_l = \langle r_l; R_l \rangle$ and $L_s = \langle r_s; R_s \rangle$ to construct tuple $L_{f'} = \langle f; R_{f'} \rangle$. Since each element of R_l and R_s is of the form $((W, H), (A, B), (x, y))$,

where $0 \leq W, H, A, B, x, y \leq n$, each of R_l and R_s contains $O(n^6)$ partial drawings. In order to consider all of their combinations in constant time, we need $O(n^{12})$ processors. Of course, not all of these combinations result into distinct partial drawings. This is because the resulting set R'_f also contains $O(n^6)$ partial drawings. So, we do have to eliminate duplicates. Eliminating duplicates in $O(\log n)$ time can be achieved by using standard techniques such as sorting, pointer doubling and list ranking.

The analysis of the shortcutting stage is identical. By noting that $O(n)$ leaves of $T(i-1)$ can participate in a shunt update, we conclude that each phase of the parallel tree contraction algorithm requires $O(n^{13})$ processors and terminates after $O(\log n)$ time.

By realising, as the next lemma shows, that it is enough to keep only one partial drawing out of those that differ only in the coordinates of the *interface to below*, i.e., they have the same enclosing and empty rectangle, the required number of processors reduces to $O(n^9)$.

Lemma 3.1 *Consider two drawings $\pi_s^1 = ((W_s, H_s), (A_s, B_s), (x_s^1, y_s^1))$ and $\pi_s^2 = ((W_s, H_s), (A_s, B_s), (x_s^2, y_s^2))$ of R_s that differ only in the coordinates of s in the drawing. Also assume a drawing $\pi_l = ((W_l, H_l), (\cdot, \cdot), (\cdot\cdot))$ of R_l and a drawing $\pi_f = ((W_f, H_f), (A_f, B_f), (x_f, y_f))$ of R_f. Let $\pi_s^{1'}$ be the drawing obtained by combining π_s^1, π_l and π_f during phase i. Let $\pi_s^{2'}$ be the drawing obtained by combining π_s^2, π_l and π_f during phase i. Then, $\pi_s^{1'}$ and $\pi_s^{2'}$ have the same enclosing and empty rectangles.*

Proof: Simply observe that the rules for computing the enclosing and empty rectangles during the pruning and shortcutting stages of each phase, do not use the coordinates of s. ∎

Theorem 3.1 *A minimum size h-v drawing of a binary tree with n nodes can be computed in $O(\log^2 n)$ parallel time using $O(n^9/\log n)$ EREW processors.*

Proof: In the pruning stage , there are at most four possible ways to arrange the subtrees T_{r_l} and T_s^{i-1}. All of them have been considered and the corresponding partial drawings have been obtained. For the shortcutting stage, there is only one way to embed $\pi^{f'}$ into π^f. At the end of the tree contraction, the root of the tree has a list of n drawings (atoms). We evaluate the cost function ψ for each drawing and we select the one of minimum cost (size). Thus, the algorithm correctly computes a minimum size h-v drawing.

We have to execute $O(\log n)$ stages of the tree contraction algorithm each requiring $O(\log n)$ time and $O(n^9)$ processors. By employing Brent's Lemma [2], it is possible to reduce the number of required processors to $O(n^9/\log n)$. ∎

3.3.1 Reducing the Number of Processors

In this section we show how to further reduce the number of processors required for the parallel algorithm to $O(n^6/\log n)$. This is achieved by providing a better upper bound for the number of partial drawings which must be maintained at each node of the tree during

the course of the tree contraction algorithm. In doing so, we introduce the *prevail* operation which might find applications to other layout algorithms.

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^i.

Definition 3.1 *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 3.2 *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 $B_u^2 > B_u^1 \geq \lambda$

The situations described by the above conditions are described graphically in Figure 5. In this paper, when using the notion of *prevail*, λ is the size (i.e., the number of nodes) of a partial tree.

Definition 3.3 *Let* $u \in V(i)$, T_u^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 *be a set of partial drawings of* T_u^i. *A partial drawing* π *in* R *is called* useful *if no other partial drawing in* R *prevails* π *with respect to the size* $|T_u|$ *of tree* T_u.

Lemma 3.2 *Let* $u \in V(i)$, T_u^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* $O(min(|T_u|, |T_u^i|) \cdot |T_u^i|^2)$.

Proof: 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 in R_u such that π^1 fits in π^2 and $A_u^2 > A_u^1 \geq |T_u|$ and $B_u^2 > B_u^1 \geq |T_u|$. Then, based on the definition of the prevail operation, π^1 prevails π^2 and π^2 is eliminated. π^2 is also eliminated in the case where $A_u^2 \leq A_u^1$ and $B_u^2 \leq B_u^1$. This means that for partial drawings that have empty rectangles with width and/or height greater than $|T_u|$, the width and/or the height of the empty rectangle is irrelevant for the prevail operation. Thus, given a partial drawing $\pi = ((W_u, H_u), (A_u, B_u), (x_u, y_u))$ that has A_u and/or B_u greater than $|T_u|$, we may replace it in R_u by a partial drawing π' whose A_u and/or B_u parameters are equal to $|T_u|$. Let R_u' be the new set of partial drawings.

Notice that in R_u', we cannot have two partial drawings with the same enclosing rectangle such that the empty rectangle of the one dominates that of the other. The maximum size of each side of an empty rectangle can be at most $|T_u^i|$. But, we only need rectangles big enough to fit the drawing of T_u, i.e., we need empty rectangles with side size at most $|T_u|$. Thus, the interesting partial drawings are the ones with empty rectangles whose side size is

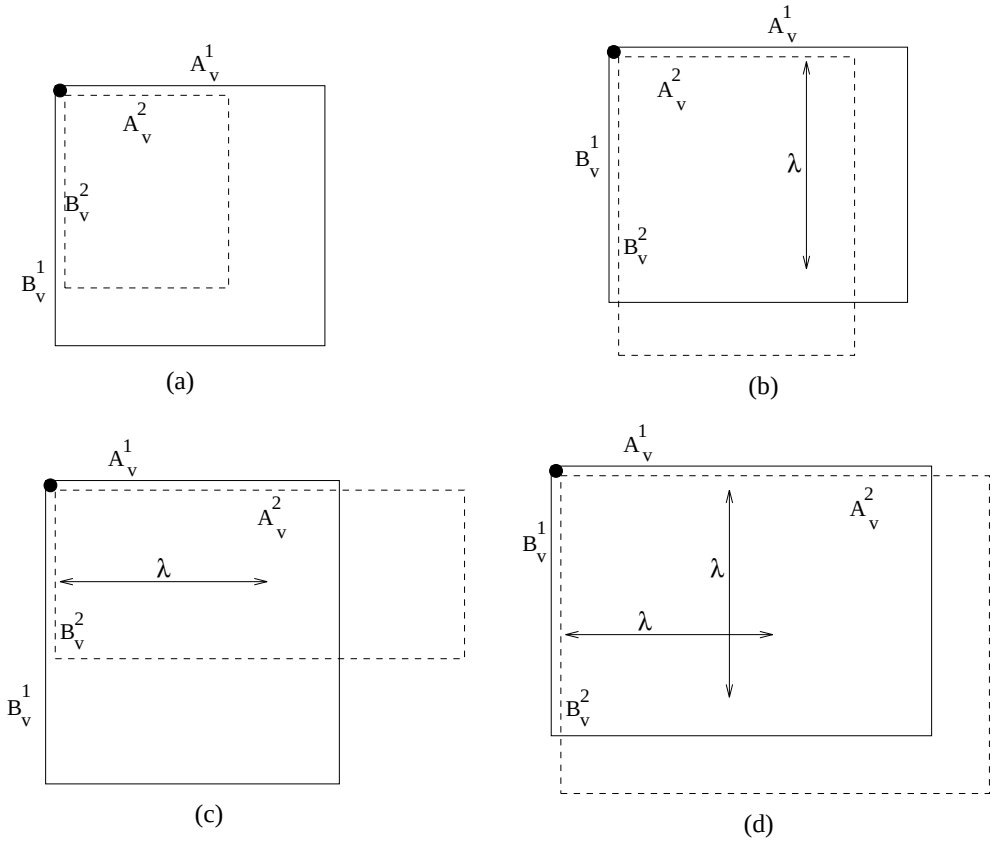

Figure 5: The situations described by Conditions a, b, c and d in the definition of *prevail*.

at most $min(|T_u|, |T_u^i|)$. Note that the maximum number of different empty rectangles such that no empty rectangle is dominated by another one is $O(min(|T_u|, |T_u^i|))$ and the number of different enclosing rectangles is $O(|T_u^i|^2)$. This results to a total of $O(min(|T_u|, |T_u^i|) \cdot |T_u^i|^2)$ useful partial drawings. ∎

In order to reduce the number of processors required for the parallel algorithm, we apply the prevail operation after each pruning and shortcutting stage. The parallel implementation of the prevail operation consists of two major steps. In the first step, each drawing $((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_s^{f'}, y_s^{f'}))$ with $A_{f'} > |T_s|$ and/or $B_{f'} > |T_s|$ is substituted by the drawing $((W_{f'}, H_{f'}), (A, B), (x_s^{f'}, y_s^{f'}))$ where $A = |T_s|$ and/or $B = |T_s|$. In the second step, basic parallel algorithmic techniques are employed (e.g., sorting, pointer doubling and list ranking) and the drawings which are prevailed by other drawings are eliminated. Both steps terminate after $O(\log n)$ time.

The following two lemmata show that we can safely eliminate prevailed partial drawings after each pruning and each shortcutting stage. By "safely", we mean that there is no valid drawing which can be created only through the involvement of an eliminated partial drawing.

Lemma 3.3 *We can safely eliminate prevailed elements from R_s after the end of each short-cutting stage.*

Proof: Let $\pi^1 = ((W_s^1, H_s^1), (A_s^1, B_s^1), (.,.))$ and $\pi^2 = ((W_s^2, H_s^2), (A_s^2, B_s^2), (.,.))$ be two partial drawings of R_s such that π^1 prevails π^2. Then, π^1 fits in π^2 and at least one of the following conditions is true:

a) $A_s^2 \leq A_s^1$ and $B_s^2 \leq B_s^1$
b) $A_s^2 \leq A_s^1$ and $B_s^2 > B_s^1 \geq \min(|T_s|, |T_s^i|)$
c) $A_s^2 > A_s^1 \geq \min(|T_s|, |T_s^i|)$ and $B_s^2 \leq B_s^1$
d) $A_s^2 > A_s^1 \geq \min(|T_s|, |T_s^i|)$ and $B_s^2 > B_s^1 \geq \min(|T_s|, |T_s^i|)$.

We distinguish among the following cases:

i) (π^1 fits in π^2) and ($A_s^2 \leq A_s^1$ and $B_s^2 \leq B_s^1$). Then, according to the definition of *prevail*, π^2 is eliminated. Assume that it is not safe to do so. That means that there exists a drawing (atom) of T_{r_s} that we only get by using the partial drawing π^2 which we want to eliminate. Say that this atom has dimensions $(W_{r_s}^2, H_{r_s}^2)$. Also assume that the drawing of T_s in this atom is π_{T_s} of dimensions (W, H). Then, we must have that:

$$W_{r_s}^2 = W_s^2 + \max(0, W - A_s^2) \qquad H_{r_s}^2 = H_s^2 + \max(0, H - B_s^2) \tag{2}$$

But, consider the drawing obtained by using π_{T_s} and π^1. We must have that:

$$W_{r_s}^1 = W_s^1 + \max(0, W - A_s^1) \qquad H_{r_s}^1 = H_s^1 + \max(0, H - B_s^1) \tag{3}$$

Note that:

$$\begin{array}{l} \max(0, W - A_s^1) \leq \max(0, W - A_s^2) \\ \max(0, H - B_s^1) \leq \max(0, H - B_s^2) \\ W_s^1 \leq W_s^2 \\ H_s^1 \leq H_s^2 \end{array} \tag{4}$$

From (2),(3) and (4) we get:

$$W_{r_s}^1 \leq W_{r_s}^2 \quad \text{and} \quad H_{r_s}^1 \leq H_{r_s}^2.$$

Thus π^2 is not an atom, a contradiction.

ii) (π^1 fits in π^2) and ($A_s^2 \leq A_s^1$ and $B_s^2 > B_s^1 \geq \min(|T_s|, |T_s^i|)$). Again, according to the definition of prevail, π^2 is eliminated. This is safe, since in this case the interesting drawings are the drawings $\pi^{1'}$ and $\pi^{2'}$ that have the same enclosing rectangles as π^1 and π^2 respectively, but their empty rectangles are $(A_s^1, \min(|T_s|, |T_s^i|))$ and $(A_s^2, \min(|T_s|, |T_s^i|))$, respectively. Now, because of case i), $\pi^{2'}$ can be safely eliminated.

iii) (π^1 fits in π^2) and ($A_s^2 > A_s^1 \geq \min(|T_s|, |T_s^i|)$ and $B_s^2 \leq B_s^1$). In this case, the interesting drawings $\pi^{1'}$ and $\pi^{2'}$ have as empty rectangles $(\min(|T_s|, |T_s^i|), B_s^1)$ and $(\min(|T_s|, |T_s^i|), B_s^2)$ respectively. Thus, $\pi^{2'}$ can be safely eliminated.

iv) (π^1 fits in π^2) and ($A_s^2 > A_s^1 \geq \min(|T_s|, |T_s^i|)$ and $B_s^2 > B_s^1 \geq \min(|T_s|, |T_s^i|)$). Obviously, the drawings $\pi^{1'}$ and $\pi^{2'}$ have the same empty rectangle $(\min(|T_s|, |T_s^i|), \min(|T_s|, |T_s^i|))$. Thus, $\pi^{2'}$ can be safely eliminated. ∎

Lemma 3.4 *We can safely eliminate prevailed elements from R_s after the end of each pruning stage.*

Proof: Let $T_{f'}$ be the partial tree rooted at f and containing T_l and T_s^{i-1} as f's children. Let $\pi_{f'}^1 = ((W_{f'}^1, H_{f'}^1), (A_{f'}^1, B_{f'}^1), (.,.))$ and $\pi_{f'}^2 = ((W_{f'}^2, H_{f'}^2), (A_{f'}^2, B_{f'}^2), (.,.))$ be two partial drawings of $R_{f'}$ such that $\pi_{f'}^1$ prevails $\pi_{f'}^2$. Then $\pi_{f'}^1$ fits in $\pi_{f'}^2$ and at least one of the following conditions is true:

a) $A_{f'}^2 \leq A_{f'}^1$ and $B_{f'}^2 \leq B_{f'}^1$
b) $A_{f'}^2 \leq A_{f'}^1$ and $B_{f'}^2 > B_{f'}^1 \geq |T_{f'}|$
c) $A_{f'}^2 > A_{f'}^1 \geq |T_{f'}|$ and $B_{f'}^2 \leq B_{f'}^1$
d) $A_{f'}^2 > A_{f'}^1 \geq |T_{f'}|$ and $B_{f'}^2 > B_{f'}^1 \geq |T_{f'}|$.

Let $\pi_f = ((W, H), (A, B), (\cdot, \cdot))$ be a partial drawing of T_f^{i-1} with which $\pi_{f'}^1$ and $\pi_{f'}^2$ will be combined during the shortcutting stage. We will show that it is safe to eliminate $\pi_{f'}^2$. Again, we distinguish four cases, one for each of the four conditions in the definition of the prevail operation:

i) ($\pi_{f'}^1$ fits in $\pi_{f'}^2$) and ($A_{f'}^2 \leq A_{f'}^1$ and $B_{f'}^2 \leq B_{f'}^1$).

This can be equivalently written as:

$$\begin{array}{ll} W_{f'}^1 \leq W_{f'}^2 & H_{f'}^1 \leq H_{f'}^2 \\ A_{f'}^2 \leq A_{f'}^1 & B_{f'}^2 \leq B_{f'}^1 \end{array} \tag{5}$$

Consider the partial drawings π_s^1 and π_s^2 we get when we combine π_f with $\pi_{f'}^1$ and $\pi_{f'}^2$, respectively. For π^1 we have:

$$\begin{array}{ll} W_s^1 = W + \max(0, W_{f'}^1 - A) & H_s^1 = H + \max(0, H_{f'}^1 - B) \\ A_s^1 = A_{f'}^1 + \max(0, A - W_{f'}^1) & B_s^1 = H_{f'}^1 + \max(0, B - H_{f'}^1) \end{array} \tag{6}$$

For π^2 we have:

$$\begin{array}{ll} W_s^2 = W + \max(0, W_{f'}^2 - A) & H_s^2 = H + \max(0, H_{f'}^2 - B) \\ A_s^2 = A_{f'}^1 + \max(0, A - W_{f'}^2) & B_s^2 = H_{f'}^1 + \max(0, B - H_{f'}^2) \end{array} \tag{7}$$

Combining (5), (6) and (7) we get that π_s^1 fits in π_s^2 and also (A_s^2, B_s^2) fits in (A_s^1, B_s^1). Thus π_s^1 prevails π_s^2. So, the use of $\pi_{f'}^2$ resulted to a partial drawing that is eliminated (according to Lemma 3.3) after the shortcutting stage.

The proof for the remaining three cases proceeds on the same lines as in Lemma 3.3. ∎

Theorem 3.2 *A minimum size h-v drawing of a binary tree with n nodes can be computed in $O(\log^2 n)$ parallel time using $O(n^6/\log n)$ EREW processors.*

Proof: By Lemmata 3.3 and 3.4 the prevail operation does not eliminate useful partial drawings. Thus, the algorithm correctly computes a minimum size h-v drawing.

For the time and processor bounds of the algorithm we note the following: During pruning, since l is a leaf, the tree T_l includes only l. Thus, we may have only $O(|T_{r_l}|)$ elements in the set R_l. Also, from Lemma 3.2, the list R_s contains at most $O(min(|T_s|, |T_s^{i-1}|) \cdot |T_s^{i-1}|^2)$ partial drawings. This means that at most $O(|T_{r_l}| \cdot min(|T_s|, |T_s^{i-1}|) \cdot |T_s^{i-1}|^2)$ partial drawings are computed, on which we apply the prevail operation to get $R_{f'}$. The number of partial drawings in $R_{f'}$ is at most $O(min(|T_{r_l}| + |T_s^{i-1}|, |T_s|) \cdot (|T_{r_l}| + |T_s^{i-1}|)^2)$.

After shortcutting, the number of computed partial drawings is $|R_f| \cdot |R_{f'}| = \min(|T_f^{i-1}|, |T_f|) \cdot |T_f^{i-1}|^2 \cdot \min(|T_{r_l}| + |T_s^{i-1}|, |T_s|) \cdot (|T_{r_l}| + |T_s^{i-1}|)^2$. Again, the prevail operation is applied on them. Note that, at each phase of the parallel tree contraction algorithm, each tree node contributes to the complexity of only one shunt operation (applied to a leaf l). This comes from the fact that the number of computed partial drawings depends on the number of nodes that have already been contracted to the nodes f, l and s. Thus, on each phase of the parallel tree contraction algorithm, for both the computation of the elements and the implementation of the prevail operation, a total number of $O(n^6)$ processors is employed. $O(\log n)$ time is needed for the implementation of the prevail. This gives a total of $O(\log^2 n)$ parallel time and $O(n^6/\log n)$ processors for the parallel tree contraction algorithm. ∎

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 a result due to Crescenzi, Di Battista and Piperno which was developed in the context of upward drawings of binary trees [4]. An *upward drawing* of a binary tree is quite similar to an h-v drawing. The only difference is that in an upward drawing of a tree the enclosing rectangles which correspond to partial drawing of subtrees rooted at sibling nodes are allowed to overlap. Related results regarding minimum area upward drawings of general trees were obtained by Garg, Goodrich and Tamassia [9].

Theorem 3.3 (Crescenzi, Di Battista, Piperno, [4]) *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.* ∎

By using the above result we can reduce the number of partial drawings of T_u^i in R_u during each tree contraction phase. More precisely, we only keep partial drawings with area bounded by $n(\log n + 1)$. As a result, at the end of each pruning stage there are at most $n^2(\log n + 1)$ partial drawings in any R-list while, during the shortcutting stage, a total of at most

$n^4(\log n+1)^2$ partial drawings might be created. Thus, on the same lines with Theorem 3.2, we can prove:

Theorem 3.4 *A minimum area h-v drawing of a binary tree with n nodes can be computed in $O(\log^2 n)$ parallel time using $O(n^4 \log n)$ EREW processors.* ∎

In the case we are interested in some layout of area bounded by $n(\log n + 1)$ rather than a minimum area layout, the number of processors required by our parallel solution can be further reduced to $O(n^2 \log^3 n)$. This is achieved by taking into account that there exist layouts with width at most $\log n + 1$ which satisfy this area requirement.

However, there is a straightforward PRAM algorithm for computing an h-v drawing of area $n(\log n + 1)$ in $O(logn)$ parallel time with $n/logn$ processors. It consists of three steps: Firstly, we determine the size of each subtree, i.e., $|T_v|$ for all nodes v of the tree. Then, we classify the tree edges as *heavy* and *light*. An edge between u and its child v is heavy if $|T_v| > |T_u|/2$, and light otherwise. The heavy edges form a collection of *heavy paths*. Finally, we draw the heavy paths as horizontal chains and the light edges as vertical segments. The computation of the exact coordinates for each node is easy and can be done with standard Euler tour techniques.

4 Conclusions

In this paper, we presented a parallel method which constructs optimal h-v drawings of binary trees. Even though the number of processors involved in our method is high, our work places the problem in NC, presenting the first algorithm with polylogarithmic time complexity. The number of processors involved in the NC algorithm is large. It would be nice to get a solution with a smaller number of processors. Also, the only sequential solution we know is based on the bottom-up approach and takes $O(n^2)$ time. It is open to derive an $o(n^2)$ sequential algorithm.

It is worth mentioning that our method can be applied to produce optimal inclusion drawings of binary trees [6] and minimum area slicing floorplans [10, 3]. For inclusion drawings the time/processor requirements are similar to those of the h-v drawings, thus placing the problem in NC. The slicing floorplanning problem is also placed in NC (with similar time and processor complexities) only in the case that the optimal layout is of polynomial size; the only case of practical interest. For details about these extensions see [8].

Acknowledgement: We would like to thank the anonymous referee for bringing to our attention the optimal parallel algorithm for h-v drawings in $n(\log n + 1)$ area which was sketched at the end of section 3.4.

References

[1] K. Abrahamson, N. Dadoun, D. Kirkpatrick and T. Przytycka, "A Simple Parallel Tree Contraction Algorithm", *Journal of Algorithms*, 10(1989), pp. 287–302.

[2] R. P. Brent, "The Parallel Evaluation of General Arithmetic Expressions", *J. ACM*, 21(1974), pp. 201–206.

[3] C-H. Chen and I. Tollis, "Parallel Algorithms for Slicing Floorplan Designs", In *Proc. of SPDP '90*, (1990), pp. 279–282.

[4] P. Crescenzi, G. Di Battista and A. Piperno, "A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees", *Computational Geometry: Theory and Applications*, Vol. 2, (1992), pp. 187–200.

[5] G. Di Battista, P. Eades, R. Tamassia and I.G. Tollis, "Algorithms for Drawing Graphs: an Annotated Bibliography", *Computational Geometry: Theory and Applications*, Vol. 4, No 5, (1994), pp. 175–198.

[6] P. Eades, T. Lin and X. Lin, "Two Tree Drawing Conventions", *International Journal of Computational Geometry and Applications*, Vol. 3, No. 2 (1993), pp. 133–153.

[7] P. Eades, T. Lin and X. Lin, "Minimum Size h-v Drawings", In *Proc. of Advanced Visual Interfaces (AVI '92)*, World Series in Computer Science, Vol. 36, (1992), pp. 386–394.

[8] P.T. Metaxas, G.E. Pantziou and A. Symvonis, "Parallel h-v drawings of Binary Trees", In *Proc. of the 5th Int. Symp. on Algorithms and Computation (ISAAC '94)*, LNCS 834, (1994), Springer Verlag, pp. 487–496. (Also TR 480, March 1994, Dept of Computer Science, University of Sydney.)

[9] A. Garg, M.T. Goodrich and R. Tamassia, "Area-Efficient Upward Tree Drawings", In *Proc. of 9th Symp. on Computational Geometry*, (1992), pp. 359–368.

[10] L. Stockmeyer, "Optimal Orientations of Cells in Slicing Floorplan Designs", *Information and Control*, Vol. 57, (1993), pp. 91–101.