PARALLEL h-v DRAWINGS OF BINARY TREES

Panagiotis T. Metaxas
Grammati E. Pantziou
Antonis Symvonis

Technical Report PCS-TR93-202

12/93
Parallel h-v Drawings of Binary Trees

Panagiotis T. Metaxas 1 Grammati E. Pantziou 2
Annis Symvonis 3

(1) Department of Computer Science, Wellesley College, Wellesley MA 02181, USA
(2) Department of Mathematics and Computer Science, Dartmouth College,
Hanover NH 03755, USA
(3) Department of Computer Science, University of Sydney, Sydney, Australia

Abstract

Drawing trees in a way that facilitates the understanding of the properties of the object depicted is part of extensive research in the areas of visualization, computational geometry and documentation systems [DET93]. Usually, the drawing has to satisfy several constraints like limited drawing space and placement of the nodes in positions governed by some predefined relations. In addition, among all the possible drawings, the preferable has to minimize some cost function, which makes it an interesting combinatorial problem.

Of particular interest is the so-called h-v drawings of binary trees problem [ELL92a] which is closely related to the slicing floorplan design problem [St83]. In this paper, we present the first fast parallel algorithm for the h-v drawing problem. Our solution runs in $O(\log^2 n)$ parallel time using $n^4$ EREW PRAM processors. Even though the number of processors involved is high, our work places the problem in NC, presenting the first algorithm with polylogarithmic time complexity. Whether these problems were in NC was an open problem [CT90].

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 visualization, computational geometry and documentation systems. Usually, the drawing has to satisfy several constraints like limited drawing space and placement of the nodes in positions governed by some predefined relations.

Of particular interest are the so-called h-v drawings of binary trees defined below. Let $T = (V, E_T)$ be a rooted binary tree on $|V| = n$ nodes and $|E_T|$ edges. A grid drawing $\pi$ of $T$ is the assignment of locations $(x_v, y_v) \in N^2$ for each node $v \in V$; this effectively assigns the line segment that connects the grid locations $(x_v, y_v)$ and $(x_w, y_w)$ on the grid $N^2$ to each tree edge $(v, w) \in E_T$. ($N$ is the set of positive integers.) The enclosing rectangle of a subtree $T_u$ is the minimum grid rectangle that contains all the nodes of $T_u$. A drawing is planar if it can be drawn on the paper so that no two of its edges intersect.

*The work of the second author is partially supported by the EEC ESPRIT Basic Reasearch Acetion No. 7141 (ALCOM II) and by the NSF postdoctoral fellowship No. CDA-9211155. Email: pmetaxas@lucy.wellesley.edu, pantziou@cs.dartmouth.edu, symvonis@basser.cs.sun.OZ.AU
An h-v drawing of a tree $T = (V, A)$ is defined to be a planar grid drawing for which each tree edge is restricted to be either a horizontal or a vertical straight line, and, for any tree node $u$ with children $v$ and $w$, the enclosing rectangles of the $u$'s children subtrees are disjoint.

The problem of minimum size h-v drawing of a binary tree $T$ is the problem of, given some cost function $\psi$, creating the smallest, with respect to $\psi$, instance of all the possible h-v drawings of $T$. Typical cost functions include: the minimization of the area ($\psi(x, y) = xy$) of the tree's enclosing rectangle, the minimization of the perimeter ($\psi(x, y) = x + y$) of the enclosing rectangle and teh minimization of the enclosing rectangle ($\psi(x, y) = \max(x, y)$).

It is an interesting fact that this tree drawing problem is actually closely related to the slicing floorplan design problem, a VLSI layout problem [St83]. Floorplan design is the problem of allocating space to a set of modules in the plane in order to minimize the area of the bounding rectangle. The plane (floor rectangle) is partitioned using vertical and horizontal line segments called slices. A floorplan is slicing if either it is a basic rectangle or there is a slice that partitions the enclosing rectangle into slicing floorplans. Stockmeyer in [St83] gave the first $O(n^2)$ time algorithm for finding optimal slicing floorplans. In [CT90], another $O(n^2)$ sequential time and an $O(n)$ parallel time using $O(n)$ processors is presented.

Even though there are several $O(n^2)$ sequential solutions to these problems, [ELL92a, ELL92b, St83, CT90], to the best of our knowledge, there is no parallel algorithm with polylogarithmic running time for them. In fact, [CT90] include it as an open problem.

In this paper, we present the first fast parallel algorithm for the minimum size h-v drawing problem. The algorithm finds the minimum area enclosing rectangle, but with small modifications, can work for other cost functions too. Our solution runs in $O(\log^2 n)$ parallel time using $n^4$ EREW PRAM processors. Even though the number of processors involved is quite high, it is the first algorithm for these problems that achieves polylogarithmic parallel time, placing the problem in NC. It was not known before if this problem was in NC.

2 The Parallel h-v Drawing Algorithm

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 [ADKP89]. 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 $T(i) = (V(i), E(i)), i = 0, 1, \ldots, k$, with $k = O(\log |T|)$. Note that the algorithm starts off with $T(0) = T$, and proceeds by contracting trees $|T(i)| \leq \varepsilon|T(i - 1)|$, with $0 < \varepsilon < 1$. At the end, $T(k)$ contains only one node.

During the $i$th 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 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 $\varepsilon$ is roughly 1/2. To use the tree contraction technique, one has to describe the updates that take place during the shunt operation. We do that in the next section.

We now give some notation and define the information we need associated with each node of the tree. We also define some useful operations and prove basic lemmas.
2.1 Data Structures and Useful Operations

Let $l$ be a leaf of a tree, $s$ its sibling, $f$ their parent and $p$ the parent of $f$. If we apply the shunt operation to $l$, then the nodes $l$ and $f$ are contracted to $s$ and $p$ is the new parent of $s$.

If $v \in V(i)$ then let $T_v$ be the subtree of $T$ rooted at $v$ and $T_v^c$ be the part 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 subtree $T_v^c$ that has been contracted to $u$, and a set $R_u$ that keeps information for all useful drawings of $T_v^c$ (the notion of a useful drawing is defined in the sequel). Each element of $R_u$ corresponds to a specific drawing $\pi$ 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 $\pi$. 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 drawing $\pi$ such that the enclosing rectangle $(W_u, H_u)$ of $\pi$ remains unchanged and $\pi$ is still a valid h-v drawing of $T_v^c$. 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 $\pi$ where $(0,0)$ is the coordinate of the top left corner of any drawing. Note that, for each $u \in V = V_0$, we initialize $L_u = < u; ((0,0), (0,0), (0,0)) >$.

Suppose that at some phase of the parallel tree-contraction algorithm we want to include the drawing $\pi$ of a subtree rooted at a node $v$, to a drawing $\pi'$ of another subtree whose $v$ is a leaf. Then, we need to know the position of $v$ in $\pi'$ as well as how much the inclusion of $\pi$ to $\pi'$ will change the rectangle corresponding to $\pi'$. 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, we shall show that the information kept by those parameters is all that is needed for the algorithm to work.

Let $\pi^1 = ((W^1, H^1), (A^1_u, B^1_u), (x^1_u, y^1_u))$ and $\pi^2 = ((W^2, H^2), (A^2_u, B^2_u), (x^2_u, y^2_u))$ be two drawings of $R_u$.

We say that rectangle $(W^1, H^1)$ dominates (or fits in) rectangle $(W^2, H^2)$ if

$$W^1 \leq W^2 \quad \text{and} \quad H^1 \leq H^2.$$ 

We say that drawing $\pi^1$ dominates (or fits in) drawing $\pi^2$ if the enclosing rectangle of $\pi^1$ fits in the enclosing rectangle of $\pi^2$.

**Definition 2.1** A drawing of $R_u$ that is dominated by no other drawing of $R_u$ with different enclosing rectangle is called an atom.

We say that drawing $\pi^1$ prevails drawing $\pi^2$ with respect to the size $|T_u|$ of a subtree $T_u$ if $\pi^1$ fits in $\pi^2$ and at least one of the following conditions is satisfied.

a) $(A^2_u, B^2_u)$ fits in $(A^1_u, B^1_u)$

b) $A^2_u \leq A^1_u$ and $B^2_u > B^1_u \geq |T_u|$

c) $A^2_u > A^1_u \geq |T_u|$ and $B^2_u \leq B^1_u$

d) $A^2_u > A^1_u \geq |T_u|$ and $B^2_u > B^1_u \geq |T_u|$.

The situation of each of the above conditions is demonstrated in Figures 1, 2, 3 and 4, respectively.
Definition 2.2 A drawing of $R_u$ that is prevailed with respect to the size $|T_u|$ of $T_u$ by no other drawing of $R_u$ with different enclosing rectangle is called useful.

Lemma 2.1 Let $u \in V(i)$, $T_u^{ci}$ be the subtree 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 drawings in $L_u$ is at most $O(\min(|T_u|, |T_u^{ci}|)|T_u^{ci}|^2)$.

Proof: The proof is based on the definition of the prevail operation. From the definition of the prevail operation, if $\pi^1 = ((W^1, H^1), (A^1, B^1), (x^1, y^1))$ and $\pi^2 = ((W^2, H^2), (A^2, B^2), (x^2, y^2))$ are two drawings in $R_u$ such that $\pi^1$ fits in $\pi^2$ and $A^2 > A^1 \geq |T_u|$ and $B^2 > B^1 \geq |T_u|$, then $\pi^1$ prevails $\pi^2$ and $\pi^2$ is eliminated. $\pi^2$ is also eliminated in the case that $A^2 \leq A^1$ and $B^2 \leq B^1$. This means that for drawings that have empty rectangles with width and/or height bigger than $|T_u|$, the width and/or the height of the empty rectangle is irrelevant for the prevail operation. Thus, given a drawing $\pi = ((W, H), (A, B), (x, y))$ that has $A$ and/or $B$ bigger than $|T_u|$, we may replace it in $R_u$ by a drawing $\pi'$ whose $A$ and/or $B$ parameters are equal to $|T_u|$. Let $R'_u$ be the new set of drawings.

Notice that in $R'_u$, we cannot have two drawings with the same enclosing rectangle whose empty rectangles dominate each other. The maximum size of any one of the sides of an empty rectangle can be at most $|T_u^{ci}|$, 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 drawings are the ones with empty rectangles whose side size is at most $\min(|T_u|, |T_u^{ci}|)$. Note that the maximum number of different empty rectangles such that no empty rectangle is dominated by another one is at most $\min(|T_u|, |T_u^{ci}|)$ and the number of different enclosing rectangles is $O(|T_u^{ci}|^2)$. This results to a total of at most $O(\min(|T_u|, |T_u^{ci}|)|T_u^{ci}|^2)$ useful drawings. □

2.2 The Shunt Updates

Let $l, s$ be the children of $f$ and $p$ be $f$'s parent in the tree $T(i - 1)$. Let also $L_l = \langle r_l; R_l \rangle$, where

$$R_l = \{(W_i^1, H_i^1), (A_i^1, B_i^1), (x_i^1, y_i^1)), \cdots, ((W_i^j, H_i^j), (A_i^j, B_i^j), (x_i^j, y_i^j))\}.$$
Figure 2: The situation described in Condition b) of the definition of \textit{prevail}.

Figure 3: The situation described in Condition c) of the definition of \textit{prevail}.
Figure 4: The situation described in Condition d) of the definition of prevail.

\[ L_f = \langle r_f; R_f \rangle, \text{ where} \]
\[ R_f = \{((W^j_1, H^j_1), (A^j_1, B^j_1), (x^j_1, y^j_1)), \ldots, ((W^j_j, H^j_j), (A^j_j, B^j_j), (x^j_j, y^j_j))\} \]

and \( L_s = \langle r_s; R_s \rangle, \text{ where} \)
\[ R_s = \{((W^1_s, H^1_s), (A^1_s, B^1_s), (x^1_s, y^1_s)), \ldots, ((W^k_s, H^k_s), (A^k_s, B^k_s), (x^k_s, y^k_s))\} \]

be the information associated with \( l \), \( s \) and \( f \) respectively. Recall that \( R_u, R_f \) and \( R_s \) keep information for all useful 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 \( T = T(0) \).

The \( \ell \)th phase of the shunt operation to a leaf \( l \) of \( V(i-1) \) consists of two stages.

**Stage 1:**

In the first 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 \( T_s \)) drawings of the subtree rooted at \( f \) and including the subtrees \( T_{r_l} \) and \( T_{s}^{c_{i-1}} \).

Let \( \pi_s = ((W_s, H_s), (A_s, B_s), (x_s, y_s)) \) be an element of the set \( R_s \) and \( \pi_l = ((W_l, H_l), (A_l, B_l), (x_l, y_l)) \) be an element of the set \( R_l \).

There are essentially 4 ways to arrange the subtrees \( T_{r_l} \) and \( T_{u}^{c_{i-1}} \). For each one we compute the new drawing. In what follows, the superscripts in \( (x^{f''}_s, y^{f''}_s) \) are used to avoid confusion with \( (x_s, y_s) \). \( (x^{f''}_s, y^{f''}_s) \) are the coordinates of \( s \) in the drawing of the subtree rooted at \( f \) and including the subtrees \( T_{r_l} \) and \( T_{s}^{c_{i-1}} \) while, \( (x_s, y_s) \) are the coordinates of \( s \) in the subtree \( T_{s}^{c_{i-1}} \).

**Case 1:** It is described in Figure 5. The produced drawing \( ((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x^{f''}_s, y^{f''}_s)) \) is defined by:

\[
W_{f'} := W_l + W_s + 1 \\
H_{f'} := \max(H_l + 1, H_s) \\
A_{f'} := A_s
\]
\[ B'_{f'} := B_s + \max(0, H_l + 1 - H_s) \]
\[ x'_{s'} := x_s + W_l + 1 \]
\[ y'_{s'} := y_s \]

The correctness of the computed values for \( W_{f'}, H_{f'}, A_{f'}, x'_{s'}, y'_{s'} \) is obvious from the figure 5. For \( B_{f'} \), note that we simply extent the height of the empty rectangle up to the bounds of the enclosing rectangle.

**Case 2:** It is described in Figure 6. The produced drawing \(((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x'_{s'}, y'_{s'}))\) is defined by:

\[ W_{f'} := \max(W_l + 1, W_s) \]
\[ H_{f'} := H_l + H_s + 1 \]
\[ A_{f'} := A_s + \max(0, W_l + 1 - W_s) \]  
\[ B_{f'} := B_s \]
\[ x'_{s'} := x_s \]
\[ y'_{s'} := y_s + H_l + 1 \]  (II)

**Case 3:** It is described in Figure 7. The produced drawing \(((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x'_{s'}, y'_{s'}))\) is defined by:
Figure 6: The way of combining two subtrees described in case 2.

Figure 7: The way of combining two subtrees described in case 3.
Figure 8: The way of combining two subtrees described in case 4.

\[ W_{f'} := W_l + W_s + 1 \]
\[ H_{f'} := \max(H_s + 1, H_l) \]
\[ A_{f'} := A_s \]
\[ B_{f'} := B_s + \max(0, H_l - H_s - 1) \]
\[ x_{s'}^{f'} := x_s \]
\[ y_{s'}^{f'} := y_s + 1 \]  \hspace{1cm} \text{(III)}

**Case 4:** It is described in Figure 8. The produced drawing \(((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_{s'}^{f'}, y_{s'}^{f'}))\) is defined by:

\[ W_{f'} := \max(W_s + 1, W_l) \]
\[ H_{f'} := H_s + H_l + 1 \]
\[ A_{f'} := A_s + \max(0, W_l - W_s - 1) \]  \hspace{1cm} \text{(IV)}
\[ B_{f'} := B_s \]
\[ x_{s'}^{f'} := x_s + 1 \]
\[ y_{s'}^{f'} := y_s \]

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

After computing all the possible 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.

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 \(B = |T_l|\). In the second step, basic parallel algorithmic techniques are employed (e.g., sorting, pointer doubling and list ranking) and the drawings that are prevailed by other drawings are eliminated. i.e., given drawings \(\pi^1\) and \(\pi^2\), \(\pi^2\) is eliminated if and only if \(\pi^1\) fits in \(\pi^2\) and the empty rectangle of \(\pi^2\) fits in the empty rectangle of \(\pi^1\).
Stage 2:
In the second 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 $\pi$ of the new set $R_s$ we combine drawings $\pi'' = ((W_{f'}, H_{f'}), (A_{f'}, B_{f'}), (x_{s'}, y_{s'}))$ of $R_1'$ with drawing $\pi' = ((W_f, H_f), (A_f, B_f), (x_f, y_f))$ of $R_f$. In simple words, we embed $\pi''$ into $\pi'$.

The new element $\pi = ((W_s, H_s), (A_s, B_s), (x_s, y_s))$ of $R_s$ produced by embedding $\pi''$ into $\pi'$ is computed as follows:

$$
W_s := W_f + \max(0, W_{f'} - A_f) \\
H_s := H_f + \max(0, H_{f'} - B_f) \\
A_s := A_{f'} + \max(0, A_f - W_{f'}) \\
B_s := B_{f'} + \max(0, B_f - H_{f'}) \\
x_s := x_f + x_{f'} \\
y_s := y_f + y_{f'}
$$

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 $\pi$ is extended to be as large as the difference between the sizes of the enclosing rectangle of $\pi''$ and the empty rectangle of $\pi'$ allow.

Finally, the prevail operation (with respect to the size $|T_i|$ of $T_s$) is applied to the computing drawings and the new set $R_s$ is obtained. The implementation of the prevail is done in the same way as in the stage 1.

3 Correctness

To prove the correctness of the algorithm we have to establish that during the course of the algorithm we keep all the possibly useful information. To do this we have to prove two things: Firstly, that we can safely use the prevail operation to shorten our $R$-sets after each stage and, secondly, 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.

**Lemma 3.1** Consider two drawings $\pi_1^{i-1} = ((W_s, H_s), (A_s, B_s), (x_{1s}, y_{1s}))$ and $\pi_2^{i-1} = ((W_s, H_s), (A_s, B_s), (x_{2s}, y_{2s}))$ of $R_s$ that differ only in the coordinates of $s$ in the drawing. Also assume a drawing $\pi_i^{i-1} = ((W_i, H_i), (A_i, B_i), (x_i, y_i))$ of $R_i$ and a drawing $\pi_f^{i-1} = ((W_f, H_f), (A_f, B_f), (x_f, y_f))$ of $R_f$. Let $\pi_1^i$ be the drawing obtained by combining $\pi_1^{i-1}$, $\pi_i^{i-1}$ and $\pi_f^{i-1}$ during phase $i$. Let $\pi_2^i$ be the drawing obtained by combining $\pi_2^{i-1}$, $\pi_i^{i-1}$ and $\pi_f^{i-1}$ during phase $i$. Then, $\pi_1^i$ and $\pi_2^i$ have the same enclosing and “empty” rectangles.

**Proof:** Easy. Simply observe that the rules for computing the enclosing and “empty” rectangles during stages 1 and 2 of each phase, do not use the coordinates of $s$. ■

**Lemma 3.2** We can “safely” eliminate prevailed elements from $R_s$ after the end of each shunt phase.
**Proof:** Consider two drawings \( \pi_1 = ((W_1^1, H_1^1), (A_1^1, B_1^1), (\ldots)) \) and \( \pi_2 = ((W_2^2, H_2^2), (A_2^2, B_2^2), (\ldots)) \) of \( R \) such that \( \pi_1 \) prevails \( \pi_2 \). Then \( \pi_1 \) fits in \( \pi_2 \) and \((A_2^2 \leq A_1^1 \text{ and } B_2^2 \leq B_1^1)\) or \((A_2^2 \leq A_1^1 \text{ and } B_2^2 \geq |T_w|)\) or \((A_2^2 > A_1^1 \geq |T_w| \text{ and } B_2^2 \leq B_1^1)\) or \((A_2^2 > A_1^1 \geq |T_w| \text{ and } B_2^2 > B_1^1 \geq |T_w|)\).

We distinguish among the following cases.

a) \( \pi_1 \) fits in \( \pi_2 \) and \((A_2^2 \leq A_1^1 \text{ and } B_2^2 \leq B_1^1)\).

Then, according to the definition of prevail, \( \pi_2 \) is eliminated. Assume that it is not safe to do so. That means that there exists a layout (atom) of \( T_s \) in \( \pi_1 \) such that we only get by using the drawing \( \pi_2 \) that 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 \( \pi_{T_s} \), of dimensions \((W, H)\).

Then, we must have that:

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

But, consider the drawing obtained by using \( \pi_{T_s} \) and \( \pi_1 \). We must have that:

\[
W_{r_s}^1 = W_s^1 + \max(0, W - A_s^1) \\
H_{r_s}^1 = H_s^1 + \max(0, H - B_s^1)
\]  

Note that:

\[
\max(0, W - A_s^1) \leq \max(0, W - A_s^2) \\
\max(0, H - B_s^1) \leq \max(0, H - B_s^2) \\
W_s^1 \leq W_s^2 \\
H_s^1 \leq H_s^2
\]

From (1), (2) and (3) we get that

\[
W_{r_s}^1 \leq W_{r_s}^2 \text{ and } H_{r_s}^1 \leq H_{r_s}^2.
\]

Thus \( \pi_2 \) is not an atom, a contradiction.

b) \( \pi_1 \) fits in \( \pi_2 \) and \((A_2^2 \leq A_1^1 \text{ and } B_2^2 > B_1^1 \geq |T_w|)\).

Again, according to the definition of prevail, \( \pi_2 \) 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 with \( \pi_1 \) and \( \pi_2 \) respectively, but their empty rectangles are \( (A_{u_1}^1, |T_w|) \) and \( (A_{u_2}^2, |T_w|) \), respectively. Now, because of case a), \( \pi_2' \) can be safely eliminated.

c) \( \pi_1 \) fits in \( \pi_2 \) and \((A_2^2 > A_1^1 \geq |T_w| \text{ and } B_2^2 \leq B_1^1)\).

In this case, the interesting drawings \( \pi_1' \) and \( \pi_2' \) have as empty rectangles the \((|T_w|, B_{u_1}^1)\) and \((|T_w|, B_{u_2}^2)\) respectively. Thus, \( \pi_2' \) can be safely eliminated.

d) \( \pi_1 \) fits in \( \pi_2 \) and \((A_2^2 > A_1^1 \geq |T_w| \text{ and } B_2^2 > B_1^1 \geq |T_w|)\).

Obviously, the drawings \( \pi_1' \) and \( \pi_2' \) have the same empty rectangle \((|T_w|, |T_w|)\). Thus, \( \pi_2' \) can be safely eliminated. ■

11
Lemma 3.3 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 $O(n^6)$ EREW processors.

Proof (sketch): In stage 1, there are four possible ways to arrange the subtrees $T_l$ and $T_s$. All of them have been considered and the corresponding drawings have been obtained. For stage 2, there is only one way to embed $\pi_f'$ into $\pi_f$. In addition, by the previous lemmas, the preval operation does not eliminate useful 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. In stage 1, since $l$ is a leaf, the tree $T_l$ includes only $l$. Thus, we may have only $O(|T_l|)$ elements in the set $R_l$. Also, from lemma 2.1, the list $R_s$ contains at most $O(min(|T_s|, |T_s^{c_i-1}|) \cdot |T_s^{c_i-1}|^2)$ drawings. This means that at most $O(|T_l| \cdot min(|T_s|, |T_s^{c_i-1}|) \cdot |T_s^{c_i-1}|^2)$ drawings are computed, on which we apply the preval operation to get $R_f'$. The number of drawings in $R_f'$ is at most $O(min(|T_l| + |T_s^{c_i-1}|, |T_s|) \cdot (|T_l| + |T_s^{c_i-1}|)^2)$.

In stage 2, the number of computed drawings is $|R_f| \cdot |R_f'| = \min(|T_l| + |T_s^{c_i-1}|, |T_s|) \cdot (|T_l| + |T_s^{c_i-1}|)^2 \cdot \min(|T_f|, |T_f'|) \cdot (|T_f| + |T_s|)^2$. Again, the preval 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 drawings depend on the number of nodes that have already 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 preval operation a total number of $O(n^6)$ processors is employed. $O(\log n)$ time is needed for the implementation of the preval. This gives a total of $O(\log^2 n)$ parallel time and $O(n^6/\log n)$ processors for the parallel tree contraction algorithm.

4 Reducing the Number of Processors

From the proof of lemma 3.3 we have that stage 1 needs roughly $O(n^4)$ processors while stage 2 requires $O(n^6)$ processors. An implementation of the second stage of the shunt operation that needs a smaller number of processors would give a better bound on the total number of processors that the algorithm requires. In this section, we present a procedure that given the tuples $L_f'$ and $L_f$ at the beginning of the second stage of the shunt operation, computes the tuple $L_s$ and needs at most $O(|T_f^{c_i-1}| \cdot \min(|T_f^{c_i-1}|, |T_f|) \cdot (|T_l| + |T_s^{c_i-1}| \cdot \min(|T_l| + |T_s^{c_i-1}|, |T_s|))$ processors. This gives a total number of $O(n^4)$ processors for the parallel tree contraction algorithm, matching the requirements of stage 1.

We observe that the assignments that compute the tuples of $R_s$ (equations (V) on page 9), once can see that $W_s$ and $A_s$ are functions of $W_f, A_f, W_f'$ and $A_f'$. Similarly, $H_s$ and $B_s$ are functions of $H_f, H_f, H_f'$ or $B_f'$. This observation suggests that one does not need to combine the full $R_f$ and $R_f'$ lists (producing a new list of length $O(n^6)$) to compute $R_s$, but only parts of them. In fact one can work with lists of length $O(n^4)$.

The problem can be formulated as a database query evaluation problem as follows: Given two relations $R_f(W_f, H_f, A_f, B_f)$ and $R_f'(W_f', H_f', A_f', B_f')$, produce a new relation $R_s(W_s, H_s, A_s, B_s)$ whose attributes obey the restrictions (V) of page 9.

We solve this problem by computing the products

$$\pi_{W_f,A_f}(R_f) \times \pi_{W_f,A_f}(R_f') \times \pi_{H_f,B_f}(R_f) \times \pi_{H_f,B_f}(R_f')$$

12
where $\pi$ is the projection and $\times$ is the cartesian product operations. From the first product we can compute the relations with attributes $W_s, A_s$, and from the second the relations with attributes $H_s, B_s$. We then combine the relations with natural join operation to compute $R_s$. Each intermediate relation has length at most $O(|T^{\pi_1-1}_f| \cdot \min(|T^{\pi_2-1}_f|, |T_f|) \cdot (|T_{r_1} + T_{s}^{\pi_2-1}| \cdot \min(|T_{r_1} + T_{s}^{\pi_2-1}|, |T_s|))$, therefore it can be computed in parallel in constant time using the same amount of processors. This means that the parallel tree contraction algorithm needs at most $O(n^3)$ processors.

**Theorem 4.1** A minimum size $h$-$v$ drawing of a binary tree with $n$ nodes can be found in $O(\log^2 n)$ parallel time using $O(n^4)$ EREW processors.

**Proof:** From the above discussion. ■

**Acknowledgements.** We would like to thank Panos Chryssanthi for valuable discussions on the query evaluation problem.

**References**


