Electronic Colloquium on Computational Complexity, Revision 01 of Report No. 069 (2001)

Optimizing the Layout of a Complete Tree*

Robert A. Legenstein*** and Wolfgang Maass

Institute for Theoretical Computer Science
Technische Universität Graz
Inffeldgasse 16b/1
8010 Graz, Austria
{legi, maass}@igi.tu-graz.ac.at
WWW home page: http://www.igi.TUGraz.at

Abstract. Tree-like organizations are a fundamental parallel processing structure. The layout of trees has so far been studied mostly for trees of degree four or less. In this article, we give tight upper and lower bounds on the total wire length of complete m-ary trees ($m \geq 2$) on a two-dimensional grid if the leaves are constraint to lie on a grid line. For the case of $m = 2$ we reproof an older result of [5] in a more direct way. The construction results in a significant reduction of the total wire length compared to the obvious "symmetric" layout.

1 Introduction

One of the most fundamental problems that arises in the construction of circuits with small total wire length (see [6]) is the question which layout of a balanced binary tree (or more generally of a balanced m-ary tree for any $m \geq 2$) on a grid has optimal total wire length. The importance of tree-like structures as an organizing principle for processing elements was stressed out by many authors, see *e.g.*[2, 11, 10]. Therefore, the layout of trees in the context of VLSI-circuits

*** corresponding author

* Research for this article was partially supported by the Fonds zur Förderung der wissenschaftlichen Forschung (FWF), Austria, project P12153, and the NeuroCOLT project of the EC.

has been studied intensively. Mead and Rem introduced the H-tree layout, thus showing that a complete binary tree with n leaves can be implemented in $O(n)$ area [7]. This upper bound on area induces an upper bound on wire length. In this strategy, leaves are placed throughout the chip surface. If leaves have to be placed on the boundary of a convex region, the area of any layout is bounded by $\Omega(n \log n)$, which was shown by Brent and Kung [3]. A generalization to noncomplete trees is due to Yao [12]. Paterson, Ruzzo, and Snyder addressed the problem of minimizing the longest edge in a tree layout. It was shown in [8] that in any layout for a complete binary tree of n leaves, there is an edge of length $\Omega(\sqrt{n}/\log n)$. Furthermore a layout is given that achieves this bound. This result is extended to binary trees (*i.e.* also noncomplete binary trees) in [9].

The total wire length of tree-layouts was also discussed by Fischer and Paterson in [5]. A preliminary version of this paper appeared in [4].The Authors give an efficient algorithm that produces an optimal layout for weighted trees under the L_1-norm if crossings are allowed.[1] Closed-form expressions on the cost of a tree layout for some special cases are given. Especially for the case of unit cost weights and the assumption that leaves are placed in a *natural* order for the tree[2], their model is similar to ours if binary trees are considered. In contrast to [5], we assume that the layout algorithm can choose the order of the leaves on the line. Lemma 1 implicitly states that we can assume without loss of generality that this order is natural. Furthermore, in our model we place nodes only on intersection points of a two-dimensional grid, whereas in the model of Fischer and Paterson, nodes may be placed anywhere on a one-dimensional line. However, the construction of the optimal layout given in Section 3 shows that we can always places nodes onto grid-points to achieve an optimal layout, even if we were not restricted to do so. Therefore, the cost of the tree layout in this

[1] In a weighted tree, each edge has a corresponding weight. The cost of and edge is its wire length multiplied with its weight.

[2] The leaves of a tree lie in a natural order, if the leaves of any two disjoint subtrees lie in different intervals of the line they are arranged on.

special case of [5] can be identified with the horizontal wire length in our model for $m = 2$. This is also reflected in the results, since our results for m-ary trees in the case of $m = 2$ are identical to those of Fischer and Paterson in this special case. The layout of m-ary trees on a grid was to our knowledge not yet considered in the literature. Since the degree of grid nodes is at most 4, merely trees of degree four or less were studied.

The layout of trees is also studied in the context of graph drawing. For a recent survey on graph drawing we refer to [1]. However, the problem of minimizing the total wire length of trees has previously not been addressed in this context.

We assume here that all n leaves of the tree have to lie on adjacent intersection points on one horizontal row of the grid. This is a case of practical interest. For example, the leaves may represent input/output ports, and such ports are normally placed on the border of a chip. Since ports are usually much larger than electrical elements, the total wire length is very sensitive to the horizontal layout of the tree. All intermediate nodes of the tree are to be placed on intersection points of the grid and edges of the tree are required to be realized by edge-disjoint paths along the grid lines. Since the vertical components of all wires contribute in any reasonable layout just a linear term to the total wire length, it suffices to focus on the horizontal components of the wires, and thus on the horizontal coordinates of the inner nodes of the tree on the grid. Intuitively one might think that a symmetric layout where the horizontal position of each node is as close as possible to the middle between the horizontal position of its children is optimal. However we show in this article that there exists another layout that reduces the total wire length by up to 33 %. Furthermore we show that this other layout strategy is optimal for the length of horizontal edges.

We will give a precise definition of the problem in Section 2. In Section 3 we will consider a layout strategy for binary trees. We give a simple argument which shows that the naive "symmetric" layout, which requires a total wire length of $\frac{n}{2}\log_2 n + O(n)$, is not optimal.The strategy presented uses a total wire length

of $\frac{n}{3}\log_2 n + O(n)$. This reproofs an upper bound that is implicit in one of the results of [5] in a more direct way. Section 3 serves as a basis for Section 4, where an upper bound for the case of complete m-ary trees is given. This layout strategy yields a total wire length of $\frac{m-1}{m+1} n \log_m n + O(n)$. Finally it is shown in Section 5 that this layout strategy is optimal for any $m \geq 2$ in the sense that no other layout can achieve for any $m \geq 2$ a total wire length $a \cdot n \cdot \log_m n + O(n)$ with $a < \frac{m-1}{m+1}$. This lower bound argument is of some interest from the technical point of view since there exist many other layouts that require less wire length on some levels of the tree, but have to spend then more wire length on other levels of the tree. The lower bound argument has to take care of all these possible trade-offs, and hence cannot be carried out by a simple inductive proof.

2 The Layout Model

We consider a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, $E \subseteq V^2$. Since the graphs we consider are balanced trees, we can level the nodes such that the root is in depth 0, all nodes that are incident to the root are in depth 1 and so on. More formally, the depth of a node v is the length of the shortest path from the root to v. The *successors* of a node v in depth i is the set of nodes that are incident with v and have depth $i + 1$.

We would like to define a layout of a graph $G = (V, E)$, i.e. we would like to map the graph onto the two dimensional plane. In our model, we consider a 2-dimensional grid with unit length in between neighboring grid-lines. We want to embed the graph into this grid graph such that each node $v \in V$ is mapped onto a node of the grid-graph by an injective function. Furthermore, the edges of the graph are mapped onto edge-disjoint paths of the grid-graph, connecting the corresponding nodes. Thus, a layout can be defined by a graph $L_G = (V', E')$ which is a subgraph of the grid-graph (see Figure 1).

The total wire length $TWL(L_G)$ of a layout L_G is the number of edges used in the grid-graph. We refer to the horizontal wire length $HWL(L_G)$ as the number of horizontal wires and the vertical wire length $VWL(L_G)$ as the

number of vertical wires used in the grid-graph. For some layout L_G, it holds that $TWL(L_G) = HWL(L_G) + VWL(L_G)$.

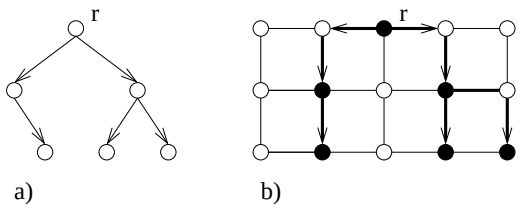

Fig. 1. A graph G (a) and a layout L_G of G in the grid-graph (b). In (b), filled circles are nodes corresponding to nodes in G and bold lines are edges corresponding to edges in G. $HWL(L_G) = 3, VWL(L_G) = 5$ and $TWL(L_G) = 8$.

3 Upper bounds for binary trees

We investigate layouts for binary trees with respect to total wire length. In this section, we construct an algorithm which builds a tree of small total wire length. In Section 5, we show that the horizontal wire length of this layout is optimal. We will only consider full balanced trees, *i.e.* trees of minimal depth. We assume that the leaves of the tree are given by consecutive nodes on a horizontal grid line. Then, one can build a binary tree by inserting inner nodes and edges into the graph. The common way of drawing trees is illustrated in Figure 2a. Each inner node is placed in the middle of its successors, as far as this is possible in our grid model. The horizontal wire length of such a layout would be at least $\frac{n}{2}((\log_2 n) - 1) + 1$ as we show below.

Proof. The naive layout places each node horizontally in the middle of its successors. We first assume that this is possible, although it is not always possible in the grid model. Then we correct the bound by the maximal error we made by this assumption. This maximal error is at most half a grid unit for each inner node of the tree and we assume that the horizontal wire length of the layout is reduced by

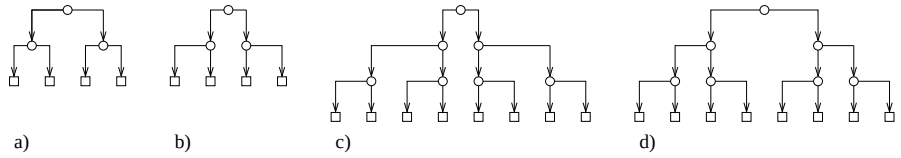

Fig. 2. Different layouts for binary trees. Rectangles are leaves and circles are inner nodes. The layout in (a) is not optimal, since one can shift the successors of the root towards it (b). A strategy with short wires from the root (c) and one that minimizes the wire-length of the sub-trees (d).

placing nodes onto grid-nodes. This amounts to a reduction of at most $\frac{1}{2}(n-2)$ in the horizontal wire length. If we assume that we can always place a node in the middle of its successors, the horizontal wire length of a naive layout T_n with n leaves can be described by the recursive formula $HWL(T_n) = \frac{n}{2} + 2HWL(T_{n/2})$ with the additional constraint that $HWL(T_1) = 0$. This recursion evaluates to $\frac{n}{2}\log_2 n$. Together with the correction term this results in a horizontal wire length of at least $\frac{n}{2}((\log_2 n) - 1) + 1$. ∎

This layout is not optimal, since one can reduce the horizontal wire length by shifting these nodes towards the root. This is possible because the horizontal wire length of the subtree is independent of the placement of its root, as long as it stays in between its successors (see Figure 2b). However, Figure 2b merely deals with a tree of depth 2. In trees of larger depth it is not so clear where to place the successors of the root, since minimizing the length of wires from the root will increase the wire length of the sub-trees. The layout in Figure 2c places the inner nodes in such a way that the wire length is optimized for the root wires, but not for the sub-trees. In Figure 2d, the subtrees are optimized, while the wires needed to connect them to the root are longer. These effects cancel for a depth 3 tree, but for trees of depth 4 or higher, the latter layout (Figure 2d) guarantees a smaller total wire length.

To keep things simple, we assume that the number of leaves is $n = 2^k$ for some natural number $k \geq 0$. In the following, we describe easy rules that define a tree-

layout as given in Figure 2d. The underlying graph of the layout is constructed in the common way: Insert the root r into the set of nodes V. Recursively build the tree for the $\frac{n}{2}$ leftmost leaves and the tree for the $\frac{n}{2}$ rightmost leaves. Then, connect the roots of these trees with r. Since the connectivity of the tree is given, we can now describe the layout of the vertices and edges. Figure 2d indicates that there are two ways to place a node. Either above its leftmost or above its rightmost successor. This reflects the principle of minimizing the length to the predecessor node without increasing the wire length of its sub-tree. The rules can be described as follows (see Figure 3):

Let u be a node to be placed on the grid. Let T_u denote the subtree which is rooted at u. Let v be the predecessor of u.

- If T_u is the leftmost sub-tree with respect to v (i.e. its leaves are to the left of all other leaves in T_v), place u above its rightmost successor.
- If T_u is the rightmost sub-tree with respect to v (i.e. its leaves are to the right of all other leaves in T_v), place u above its leftmost successor.

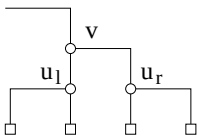

Fig. 3. u_l is the root of the leftmost subtree with respect to v. Therefore, u_l is placed above its rightmost successor. u_r is the root of the rightmost subtree with respect to v. Therefore, u_r is placed above its leftmost successor.

It is straight-forward to design an algorithm that constructs such a tree-layout. Such an algorithm would start to place nodes that are incident to leaves, since their placement is independent of the placement of other nodes. After these nodes are placed, it places nodes that are incident with them and so on. We call this algorithm *Greed*.The horizontal wire length of a layout produced by Greed is given in the following theorem.

Theorem 1. *If, for some $k \geq 1$, the $n = 2^k$ leaves of a totally balanced binary tree are placed on consecutive grid nodes of a horizontal grid-line, then there exists a layout in the grid model with horizontal wire length of* $\frac{1}{3}n\log_2 n + \frac{1}{9}(n + (-1)^{(\log_2 n)+1})$.

Proof. Let T_n be a binary tree layout for $n = 2^k$ leaves constructed by Greed. We define the *root width* $b(k)$ to be the distance in between the successors of the root (*i.e.* nodes of depth 1 in the tree). Figure 4 illustrates the root width of a tree constructed by Greed. Since the subtrees of T_n are constructed in the same manner, $b(1), \ldots, b(k)$ can be found within sub-trees of T_n. To be more precise, $b(i)$ appears 2^{k-i} times (see Figure 4), and the horizontal wire length of T_n is the sum of these root widths:

$$HWL(T_n) = \sum_{i=1}^{k} 2^{k-i} b(i) \tag{1}$$

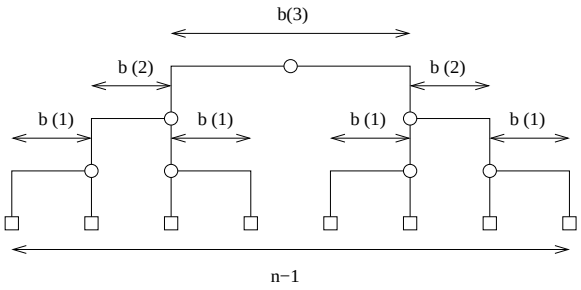

Fig. 4. A binary tree layout. Rectangles are leaves and circles are inner nodes. The *root width* of this tree is $b(3) = 3$. But since the subtrees are constructed in the same manner, also $b(2)$ and $b(1)$ appear within this tree.

A recursive formula for $b(k)$ can be stated. Consider the line segment l that is defined by the leftmost and rightmost leaves as endpoints. l is $n - 1 = 2^k - 1$ in length. Projecting the wires from the root to its successors onto l results in an interval of size $b(k)$ on this line. The root is connected to two trees of root-width $b(k-1)$. Because of the placement of nodes, the projections of the root-widths of

the tree and its left and right subtrees are directly adjoining and non-overlapping (see Figure 4). Projecting the root widths of leftmost and rightmost subtrees of depth i for $i = 1, \ldots, k$ onto l results in adjoining and non-overlapping intervals that fully cover l. Hence, for any $k \geq 2$ it holds that

$$b(k) = (2^k - 1) - 2\sum_{i=1}^{k-1} b(i) \tag{2}$$

and $b(1) = 1$. We show by induction on $k \geq 1$ that the following holds:

$$b(k) = \frac{1}{3}(2^k + (-1)^{k+1}). \tag{3}$$

Obviously, the induction hypothesis, Equation 3, is true for $k = 1$. Using Equation 2, we show the inductive step:

$$\begin{aligned} b(k+1) &= (2^{k+1} - 1) - 2\sum_{i=1}^{k} b(i) = (2^{k+1} - 1) - \frac{2}{3}\sum_{i=1}^{k}(2^i + (-1)^{i+1}) \\ &= (2^{k+1} - 1) - \frac{2}{3}\Big(\sum_{i=1}^{k} 2^i + \sum_{i=1}^{k}(-1)^{i+1}\Big) \\ &= (2^{k+1} - 1) - \frac{2}{3}\Big(2^{k+1} - 2 + \frac{1}{2}((-1)^{k+1} + 1)\Big) \\ &= \frac{1}{3}2^{k+1} - \frac{1}{3}(-1)^{k+1} = \frac{1}{3}(2^{k+1} + (-1)^{(k+1)+1}). \end{aligned}$$

We can now sum up the root widths that occur in our tree layout with Equation 1 to compute the horizontal wire length of the layout.

$$\begin{aligned} HWL(T_n) &= \sum_{i=1}^{k} 2^{k-i} b(i) = \sum_{i=1}^{k} 2^{k-i}\frac{1}{3}\big(2^i + (-1)^{i+1}\big) \\ &= \frac{1}{3}\Big(\sum_{i=1}^{k} 2^k + \sum_{i=1}^{k} 2^{k-i}(-1)^{i+1}\Big) = \frac{1}{3}\Big(k2^k + \sum_{j=0}^{k-1} 2^j(-1)^{k-1+j}\Big) \end{aligned}$$

We eliminate the remaining alternating sum with the formula

$$\sum_{j=0}^{k} 2^j(-1)^{k+j} = \frac{2^{k+1} + (-1)^k}{3}. \tag{4}$$

Since $2^k = n$ and $k = \log_2 n$, the horizontal wire length of this layout evaluates to

$$HWL(T_m^n) = \frac{1}{3}k2^k + \frac{1}{9}\Big(2^k + (-1)^{k+1}\Big) = \frac{1}{3}n\log_2 n + \frac{1}{9}\Big(n + (-1)^{(\log_2 n)+1}\Big).$$

In the Section 5, we will show that this bound is optimal even if we do not restrict inner nodes to lie on grid-points. As far as the vertical wire length is concerned, there is one vertical wire for each node except for the root, which results in a vertical wire length of $VWL(T_n) = 2(n-1)$. Hence, the total wire length of T_n is $TWL(T_n) = \frac{1}{3}n\log_2 n + \frac{1}{9}(n + (-1)^{(\log_2 n)+1}) + 2(n-1) = \frac{n}{3}\log_2 n + O(n)$.

4 Upper Bounds for balanced trees

The simplicity of the tree construction in Section 3 is instructive. However, with a somewhat more elaborate model, one can generalize this construction to balanced m-ary trees for any $m \geq 2$. We face the problem that for any $m \geq 4$, there is no way to use edge-disjoint paths to connect nodes with their successors (since the degree of inner nodes of the tree is more than 4). Hence, we have to extend our model. Consider a computational circuit[3] C. This circuit has an underlying directed graph $G = \{V, E\}$ where V is viewed as the set of gates and input-ports of the circuit and E defines connections between them. In a physical implementation of C, if some gate g is connected to m successors, the edges from g to its successors may be implemented by a wire that starts at g and spreads later on to serve as input for the successors. Such a connection strategy is reasonable as long as m is constant, *i.e.* does not grow with the size of the circuit.

A convenient way to define such a layout is to insert nodes into V that serve as routing points for the edges. Hence we define a *routing graph* $G' = (V', E')$ where $V \subseteq V'$. Nodes in $V' - V$ are called *routing nodes*. Furthermore, each edge $(v_i, v_j) \in E$ is mapped onto a path in G' that starts with v_i and ends with v_j and any other node on the path is a routing node. Since the graph G is underlying some computational circuit, we need to be careful about routing nodes. As the routing nodes do not represent computational units, it does not

[3] See *e.g.*[10] for a brief introduction in computational circuits.

make sense for routing nodes to have an in-degree larger than one[4]. To get a layout L_G, we can map the nodes of G' onto nodes of the grid-graph by some function $pos : V' \to \mathbb{N}^2$ and furthermore map the edges of G' onto edge-disjoint paths of the grid-graph. The horizontal, vertical, and total wire length can be defined as in Section 2.

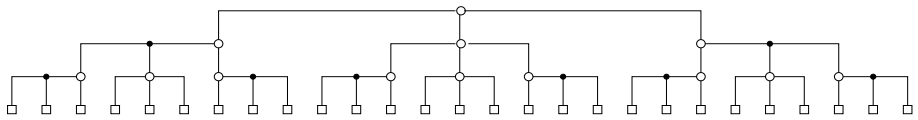

Fig. 5. A layout for a 3-ary tree of depth 3. Rectangles are leaves, open circles are inner nodes, and filled circles are routing nodes. Note that all edges are directed from nodes of smaller depth to nodes of larger depth.

Within this model, the principle of this layout given in Section 3 is also applicable for m-ary trees where m is larger than 2. Figure 5 shows a layout for a 3-ary tree of depth 3. In the following, we give simple rules on how to place inner nodes on the grid that define a tree-layout with small total wire length[5].

Let u be a node to be placed on the plane. Let T_u denote the subtree with root u and v denote the predecessor of u (see Figure 6).

- If T_u is the leftmost sub-tree with respect to v (i.e. its leaves are to the left of all other leaves in T_v), place u above its rightmost successor.
- If T_u is the rightmost sub-tree with respect to v (i.e. its leaves are to the right of all other leaves in T_v), place u above its leftmost successor.
- Otherwise, u may be placed above any of its successors.

Again, it is straight-forward to design an algorithm $Greed_m$ that constructs such a tree-layout. $Greed_m$ start to place nodes that are incident to leaves, since

[4] The in-degree of a node v in a graph G is defined as the number of edges (u, w)in G with $w = v$. Similarly, the out-degree v is defined as the number of edges (u, w)in G with $u = v$.

[5] To keep things simple, we assume that the number of leaves is $n = m^k$ for some natural number $k \geq 0$.

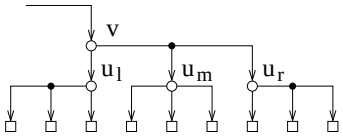

Fig. 6. u_l is the root of the leftmost subtree with respect to v. Therefore, u_l is placed above its rightmost successor. u_r is the root of the rightmost subtree with respect to v. Therefore, u_r is placed above its leftmost successor. u_m may be placed above any of its successors.

their placement is independent of the placement of other nodes. After these nodes are placed, it places nodes that are incident with them and so on. The horizontal wire length of a layout produced by $Greed_m$ can be calculated in the same manner as in Section 3.

Theorem 2. *If, for some $k \geq 1$, $m \geq 2$, the $n = m^k$ leaves of a totally balanced m-ary tree are placed on consecutive grid nodes of a horizontal grid-line, then there exists a layout in the grid model with horizontal wire length of* $\frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}(n + (-1)^{(\log_m n)+1})$.

Proof. Let $T_m^{m^k}$ be a m-ary tree layout constructed by $Greed_m$. We define the *root width* $b_m(k)$ to be the maximum over all horizontal distances in between pairs of nodes that are incident with the root (*i.e.* nodes of depth 1 in the tree). Using the same argument as in Section 3, the horizontal wire length of $T_m^{m^k}$ is

$$HWL(T_m^{m^k}) = \sum_{i=1}^{k} m^{k-i} b_m(i) \tag{5}$$

The root width $b_m(k)$ of a layout by $Greed_m$ for a tree $T_m^{m^k}$ is given by the recursive equation

$$b_m(k) = (m^k - 1) - 2 \sum_{i=1}^{k-1} b_m(i) \tag{6}$$

for any $k \geq 2$ and $b(0) = m - 1$. As in Section 3, one can show that

$$b_m(k) = \frac{m-1}{m+1}(m^k + (-1)^{k+1}). \tag{7}$$

Again, simple algebra is used to compute the horizontal wire length of the layout by Equation 5.

$$\begin{aligned} HWL(T_m^{m^k}) &= \sum_{i=1}^{k} m^{k-i} b_m(i) \\ &= \sum_{i=1}^{k} m^{k-i} \frac{m-1}{m+1}\left(m^i + (-1)^{i+1}\right) \\ &= \frac{m-1}{m+1}\Big(\sum_{i=1}^{k} m^k + \sum_{i=1}^{k} m^{k-i}(-1)^{i+1}\Big) \\ &= \frac{m-1}{m+1}\Big(km^k + \sum_{j=0}^{k-1} m^j(-1)^{k-1+j}\Big). \end{aligned}$$

We eliminate the alternating sum with the formula

$$\sum_{j=0}^{k} m^j(-1)^{k+j} = \frac{m^{k+1} + (-1)^k}{m+1}. \tag{8}$$

Since $m^k = n$ and $k = \log_m n$, the horizontal wire length of this layout evaluates to

$$\begin{aligned} HWL(T_m^n) &= \frac{m-1}{m+1} km^k + \frac{m-1}{(m+1)^2}\Big(m^k + (-1)^{k+1}\Big) \\ &= \frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}\Big(n + (-1)^{(\log_m n)+1}\Big). \end{aligned} \tag{9}$$

∎

Note that this upper bound is a generalization of Theorem 1. If we set $m = 2$, we get exactly the bound given for binary trees. In the following Section, we show that this general bound is optimal even if we do not restrict inner nodes to lie on grid-points. The vertical wire length of this layout is obvious. There is one vertical wire for each node except for the root, which results in a vertical wire length of $VWL(T_m^n) = \frac{mn-1}{m-1} - 1$. Hence, the total wire length of T_m^n is $TWL(T_m^n) = \frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}\Big(n + (-1)^{(\log_m n)+1}\Big) + \frac{mn-1}{m-1} - 1 = \frac{m-1}{m+1} n \log_m n + O(n)$.

5 Lower Bounds for balanced trees

We show that the bound given in Theorem 2 for the horizontal wire length of a tree layout is tight. We relax the grid-model such that except for the leaves,

which are placed on successive nodes on a horizontal grid line as before, nodes may be placed anywhere on the two-dimensional plane. The wire length of an edge (u, v) in the routing graph is the Euclidean distance between u and v on the plane. Since we merely consider the horizontal components of wires, we can assume that all nodes lie on the horizontal line defined by the leaves of the tree. Hence, the position of a node $v \in V'$ is defined by a function $xpos : V' \to \mathbb{R}$ which maps v onto its x-coordinate. The horizontal wire length of an edge (u, v) is $|xpos(u) - xpos(v)|$ (note that the graph involves also routing nodes).

Before we give the proof, we make another assumptions about a layout for a balanced tree that minimizes the horizontal wire length. For this assumption, we need the following definition. Consider two nonempty, disjunct sets of nodes $S_1, S_2 \in V$ of some layout L. We say that S_1 is *horizontal non-overlapping* with S_2 if some vertical line separates the nodes of S_1 and S_2, *i.e.* for each pair of nodes $v_1 \in S_1$ and $v_2 \in S_2$, it holds that $xpos(v_1) < xpos(v_2)$ or for each pair $v_1 \in S_1$ and $v_2 \in S_2$, it holds that $xpos(v_1) > xpos(v_2)$. If this is not the case, we say that S_1 is *horizontal overlapping* with S_2. Consider a layout for a m-ary balanced tree T. The root of the tree is connected to m subtrees $T_1, \ldots, T_m$. We say that T is horizontal overlapping, if for some $i, j \in \{1, \ldots, m\}$ with $i \neq j$, the set of leaves of T_i is horizontal overlapping with the set of leaves of T_j. Otherwise, we say that T is horizontal non-overlapping. The following lemma states that without loss of generality, we can assume that a tree layout with minimal horizontal wire length is horizontal non-overlapping.

Lemma 1. *For any layout T of a balanced tree in the model given above with horizontal wire length $HWL(T)$, there exists a layout T' where T' is horizontally non-overlapping and the horizontal wire length of T' is smaller or equal to $HWL(T)$.*

Proof. We construct a horizontal non-overlapping layout T' out of T with smaller or equal horizontal wire length. This is done by constructing a layout where no paths cross. In the following, we define an operation on a layout that achieves

this goal if applied correctly. Furthermore, we construct a layout R_2 out of T to show on which nodes to apply the operation.

One can assume that all nodes of T are on the horizontal line defined by the leaves. If this is not the case, shrink the layout to this line and consider this layout. Let r be the root of T.In the first step of the construction of T', we delete all routing nodes out of T and for each node $v \in V - \{r\}$ we place a routing node s_v on the same position as v. We connect each routing node s_v with its corresponding node v. Then for each node u and its successors $u_1, \ldots, u_m$, connect u to the corresponding routing nodes with minimal total wire length (*i.e.* chain the routing nodes as shown in Figure 7). Note that the positions of nodes in V are not altered and the way to connect nodes with their successors is optimal, hence the wire length of the layout does not increase by this manipulation.

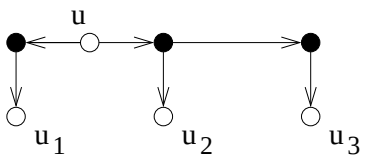

Fig. 7. Filled circles are routing nodes. Given the positions of the nodes (open circles), this is an optimal way to connect u to its successors with regard to the horizontal wire length. The nodes u_1, u_2, u_3 are on the same position as their corresponding routing nodes. The vertical displacement of nodes in the figure is for clarity.

We are now defining an operation on the layout that does not increase the horizontal wire length of the layout. We call such an operation a *flip*-operation. Consider two inner nodes u and v and the sets of their successors S_u and S_v. Assume that u is to the left of v or placed on the same location as v. The operation connects u to the m leftmost nodes in the set of their successors $S_u \cup S_v$ and v to the m rightmost nodes in $S_u \cup S_v$. This operation does not increase the horizontal wire length of the resulting layout. We formalize this idea: Suppose that $posx(u) \leq posx(v)$. Chose disjunct subsets $S'_u, S'_v \in S_u \cup S_v$ of size

m each with the condition that for any pair of nodes $u' \in S'_u$ and $v' \in S'_v$ it holds that $xpos(u') \leq xpos(v')$. The flip-operation $flip_{u,v}$ deletes all connections in between u and its successors and v and its successors and connects u to the nodes in S'_u and v to S'_v in an optimal way. This will not increase the horizontal wire length of the layout.

We will use this operations to construct a horizontal non-overlapping tree out of T'. In order to figure out on which nodes to do that, we construct two other layouts R_1 and R_2 out of T.

We construct R_1 out of T such that we delete routing nodes and connect graph nodes by direct edges. Furthermore, all nodes of depth i are placed on a horizontal line at y-coordinate $-i$ and the x-coordinates are unchanged. Such a stretched layout is given in Figure 8a.

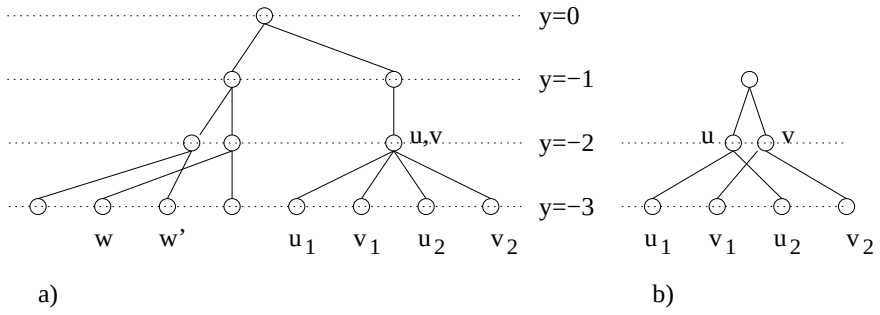

Fig. 8. a) A stretched layout. Nodes of depth i are placed at y-coordinate $-i$. b) We separate nodes that are placed on the same position by a small amount. A detail of the layout in (a) is shown.

In order to handle nodes that are on the same position on the plane , we construct another layout R_2 out of R_1 by the algorithm given in Figure 9. We do that by displacing nodes of same position by a small amount d in the x-direction, such that no two nodes are at the same x-position after the transformation. Let ϵ be the minimum horizontal distance greater than 0 between nodes of the layout. d must be chosen sufficiently small, so that if a node v is to the left of u in R_1,

then v is to the left of u in R_2. This can be guaranteed by choosing $d = \epsilon/2$. In R_2, paths can only cross by edge-crossings.

```
Let d be sufficiently small.
FOR all nodes u,v with pos(u)=pos(v) DO
  Set xpos(u):=xpos(u)-d/2
  Set xpos(v):=xpos(v)+d/2
  d := d/2
DONE
```

Fig. 9. An algorithm that produces a layout R_2 out of R_1.

Figure 8b shows a detail of the transformed version of the layout of Figure 8a. In the layout R_2, there are no two nodes that are placed on the same position. All edge crossings in R_2 are due to crossings that were in R_1 or or due to nodes that were at the same location in R_1. Suppose that there are nodes u, v such that edges (u, u') and (v, v') cross in R_2. A flip operation $flip_{u,v}$ on T' and the corresponding change in the connectivity in R_2 can be used to eliminate all crossings within edges from u and v to their successors. Hence, all the crossings in R_2 can be eliminated without increasing the horizontal wire length in T'.

We show that after crossings were eliminated in R_2, R_2 is horizontal non-overlapping. Suppose that R_2 is horizontal overlapping (see Figure 10). There exist successors of the root l, r with $xpos(l) < xpos(r)$ in R_2 such that for some leaves u, v of the subtree defined by l and w of the subtree defined by r it holds that $xpos(u) < xpos(w) < xpos(v)$. There are paths from l to u, from l to v and from r to w that do not visit the root of R_2. Since $xpos(u) < xpos(w) < xpos(v)$, the path from r to w crosses one of the other paths. Since no two nodes share the same location on the plane, edges cross. Hence, if there are no crossings in R_2, then R_2 is horizontal non-overlapping. T' is also horizontal non-overlapping, since its underlying graph has the same connectivity and the leaves are placed in the same order on the horizontal line. ∎

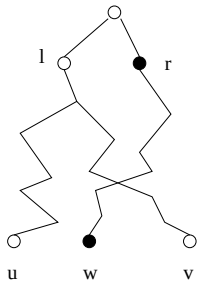

Fig. 10. A tree layout R_2 where l and r are successors of the root. If R_2 is horizontally overlapping, then a path from l crosses with a path from r.

Theorem 3. *For integers $k \geq 1$ and $m \geq 2$, any layout of a balanced tree with $n = m^k$ leaves that lie on a horizontal line with unit distance in between neighboring leaves has horizontal wire length of at least* $\frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}(n + (-1)^{(\log_m n)+1})$.

Proof. We prove the theorem by induction on k for all n of the form $n = m^k$. To avoid cumbersome notation, we introduce the shortcut $f_m(n) = \frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}(n + (-1)^{(\log_m n)+1})$ to denote the bound given above. We will prove a stronger result that implies the theorem. Since the m^k leaves are placed on a horizontal line, one can define the midpoint p_m of the leaves, which is the point that minimizes the maximal distance to a leaf. Let r be the root of the tree. The *deviation* x of the root is defined by $x = xpos(w) - xpos(p_m)$. The wire length of the tree layout will increase if the absolute value of the deviation is too big. On the other hand, we can give a region around the midpoint where no increase in wire-length occurs. This *bound region* for a tree of n leaves is defined by an interval of length $b_m(n) = \frac{m-1}{2(m+1)}(n + (-1)^{(\log_2 n)+1})$ to both sides of the midpoint. If the root is placed within this interval, the wire length needed is independent of the deviation. If the root is placed outside this interval, the wire length increases linearly with the deviation. Hence, we define the *limited deviation* to be $\bar{x} = \max\{0, |x| - b_m(n)\}$. The horizontal wire length of the balanced tree is at least $f_m(n) + \bar{x}$.

Observation 4 *Consider a layout for a m-ary tree with n leaves, deviation x and limited deviation $\bar{x}$. Then the following holds:*

a) $\bar{x} + x \geq -b_m(n)$,

b) $\bar{x} - x \geq -b_m(n)$,

c) $b_m(mn) + b_m(n) = \frac{n(m-1)}{2}$, *and*

d) $mf_m(n) = f_m(mn) - (m-1)n + 2b(n)$.

Proof (Observation 4). Observations 4a,b are easy to verify. Suppose that $|x| \leq b_m(n)$. Then $\bar{x} \pm x = \pm x \geq -b_m(n)$. Suppose that $|x| > b_m(n)$. Then $\bar{x} \pm x = |x| - b_m(n) \pm x \geq -b_m(n)$. Observation 4c is proven by $b_m(mn) + b_m(n) = \frac{m-1}{2(m+1)}(mn+(-1)^{\log_m n}) + \frac{m-1}{2(m+1)}(n+(-1)^{\log_m n+1}) = \frac{m-1}{2(m+1)}(mn+n) = \frac{n(m-1)}{2}$. Finally, we show Observation 4d:

$$\begin{aligned}
mf_m(n) &= \frac{m-1}{m+1}(mn)\log_m n + \frac{m-1}{(m+1)^2}(mn) + \frac{m-1}{(m+1)^2}(-1)^{(\log_m n)+1} \\
&= \frac{m-1}{m+1}(mn)\log_m (mn) - \frac{m-1}{m+1}(mn) + \frac{m-1}{(m+1)^2}(mn) \\
&\quad + \frac{m-1}{(m+1)^2}m(-1)^{(\log_m n)+1} \\
&= \frac{m-1}{m+1}(mn)\log_m (mn) + \frac{m-1}{(m+1)^2}(mn) + \frac{m-1}{(m+1)^2}(-1)^{(\log_m mn)+1} \\
&\quad - \frac{m-1}{m+1}(mn) + \frac{m-1}{(m+1)^2}(m+1)(-1)^{(\log_m n)+1} \\
&= f(mn) - \frac{m-1}{m+1}(mn) + \frac{m-1}{m+1}(-1)^{(\log_m n)+1} \\
&= f(mn) + 2b_m(n) - 2b_m(n) - \frac{m-1}{m+1}(mn) + \frac{m-1}{m+1}(-1)^{(\log_m n)+1} \\
&= f(mn) + 2b_m(n) - \frac{m-1}{m+1}(n + (-1)^{(\log_m n)+1}) - \frac{m-1}{m+1}(mn) \\
&\quad + \frac{m-1}{m+1}(-1)^{(\log_m n)+1} \\
&= f(mn) + 2b_m(n) - n(m+1)\frac{m-1}{m+1} \\
&= f(mn) - (m-1)n + 2b_m(n).
\end{aligned}$$

∎

Observation 4c proves that the bound region $b(mn)$ of a tree of mn leaves is directly adjoining the bound region of its leftmost (respectively rightmost)

subtree of n leaves as shown in Figure 11. This becomes clear if one takes into account that the distance between the midpoint of a tree with mn leaves and the midpoint of its leftmost (respectively rightmost) subtree is $\frac{n(m-1)}{2} = b_m(mn) + b_m(n)$ (as shown above we can assume that the leaves of the leftmost subtree are the n leftmost leaves of the tree). Now we are ready to give the inductive proof.

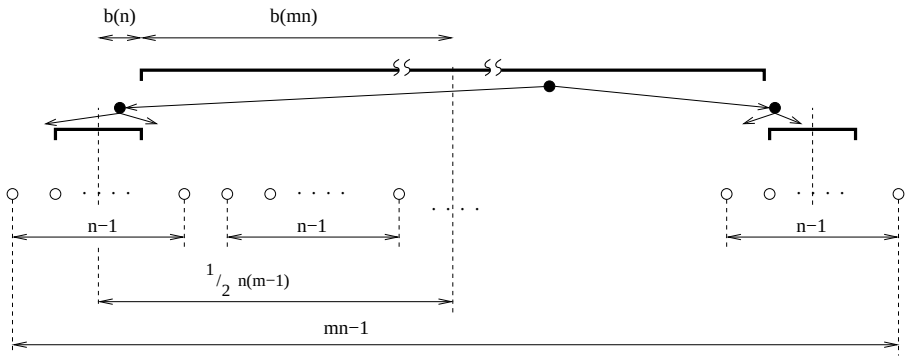

Fig. 11. Bound regions of a tree. Filled circles are the root and its successors and open circles are leaves of a tree. Bold lines indicate the bounded regions of the tree and its sub-trees. These regions are directly adjoining.

Our hypothesis is that any layout for a balanced tree of $n = m^k$ leaves that lie on a row of grid-points needs a horizontal wire length of at least $f(n) + \bar{x}$, where $\bar{x}$ is the limited deviation of the root. In the case of $n = 1$, the minimal wire length needed for a tree is the wire needed to route the only leaf to the position of the root, which is $|x|$. Our bound evaluates to $f(1) + \bar{x} = \bar{x} = \max\{0, |x| - 0\} = |x|$. For the induction step, we consider an arbitrary layout T for mn leaves. The root r of T is connected to m subtrees $T_1, \ldots, T_m$ of n leaves each.

Because we can assume that the tree is horizontal non-overlapping, there is a leftmost and a rightmost subtree, where leaves of the leftmost (respectively rightmost) subtree are the n leftmost (respectively rightmost) leaves of T. Let T_L be the leftmost subtree, r_L be its root, T_R be the rightmost subtree and r_R be its root. Furthermore let x_L denote the displacement of r_L , x_R denote the

displacement of r_R, and x denote the displacement of r. Because of symmetry, we can assume without loss of generality that the root r is placed to the right of the midpoint. The situation is illustrated in Figure 12. We consider two cases, whether r is displaced by at most $b_m(mn)$ or by more than $b_m(mn)$.

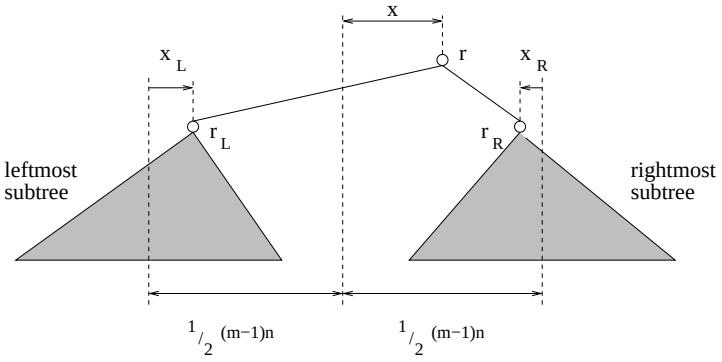

Fig. 12. Inductive step: A m-ary tree with mn leaves. Only the leftmost and the rightmost sub-tree is shown. The root r is displaced by x from the midpoint. The r_L is displaced by x_L from its midpoint and r_R is displaced by x_R from its midpoint.

In the first case we have $x \leq b_m(mn)$ and therefore $\bar{x} = 0$. The horizontal wire length of the tree is the sum of the wire lengths of the sub-trees and the wires to connect these trees. At least, there is a wire from the root to r_L and r_R.

$$\begin{aligned} HWL(T_{mn}) &\geq \sum_{i=1}^{m} HWL(T_i) + (m-1)n - x_L + x_R \\ &\geq m f_m(n) + \bar{x}_L + \bar{x}_R + (m-1)n - x_L + x_R \\ &\geq f_m(mn) - (m-1)n + 2b_m(n) + (m-1)n + \bar{x}_L - x_L + \bar{x}_R + x_R \\ &\geq f_m(mn) + 2b_m(n) - b_m(n) - b_m(n) \\ &\geq f_m(mn) \geq f(mn) + \bar{x}. \end{aligned}$$

In the second case, we have $x > b_m(mn)$ and therefore $\bar{x} > 0$. Suppose that $xpos(r) \leq xpos(r_R)$ (*i.e.* r is not to the right of r_R, see Figure 13a). Since $\bar{x} > 0$,

it follows that $b_m(n) + x_r \geq \bar{x}$ (this is also shown in Figure 13a). Hence,

$$\begin{aligned} HWL(T_{mn}) &\geq mf_m(n) + \bar{x}_L + \bar{x}_R + (m-1)n - x_L + x_R \\ &\geq f_m(mn) + 2b_m(n) + \bar{x}_L - x_L + \bar{x}_R + x_R \\ &\geq f_m(mn) + b_m(n) + \bar{x}_R + x_R \\ &\geq f_m(mn) + b_m(n) + x_R \geq f_m(mn) + \bar{x}. \end{aligned}$$

Suppose that r is to the right of r_R. In Figure 13b, it is illustrated that the wire needed to connect r_R to r is $\bar{x} - (b_m(n) + x_R)$ long. The wire needed to connect r_L and r_R is as before $n(m-1) - x_L + x_R$ in length. Hence,

$$\begin{aligned} HWL(T_{mn}) &\geq \sum_{i=1}^{m} HWL(T_i) + n(m-1) - x_L + x_R + \bar{x} - (b_m(n) + x_R) \\ &\geq f_m(mn) + 2b_m(n) + \bar{x}_L - x_L + \bar{x}_R + x_R + \bar{x} - b_m(n) - x_R \\ &\geq f_m(mn) + b_m(n) + \bar{x}_L - x_L + \bar{x}_R + \bar{x} \geq f_m(mn) + \bar{x}. \end{aligned}$$

∎

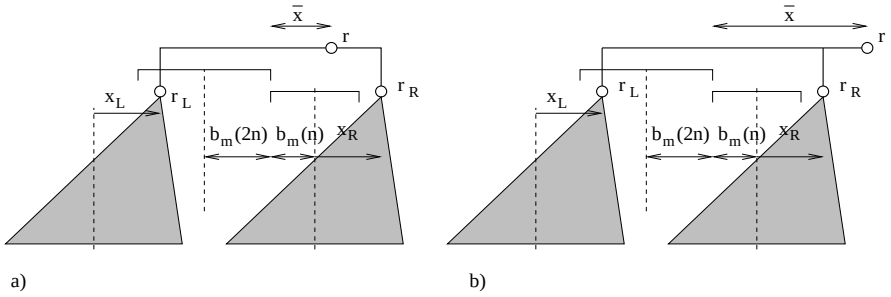

Fig. 13. The root r is out of its bounded region. a) The root is not to the right of r_R. One can see that $b(n) + x_r \geq \bar{x}$. b) The root is to the right of r_R. The wire needed to connect r_R to r is $\bar{x} - (b(n) + x_R)$.

Hence, the layout in Section 4 is optimal with respect to the horizontal wire length. Furthermore, this shows that no layout for an m-ary tree can achieve for any $m \geq 2$ a total wire length $a \cdot n \cdot \log_m n + O(n)$ with $a < \frac{m-1}{m+1}$.

6 Discussion

We have exhibited in this article layouts of complete m-ary trees on a grid that are optimal with regard to their horizontal total wire length. Studies of tree layouts have so far mostly dealt with binary trees [3, 5]. The model we use in this article allows us to deal with m-ary trees in a way that is of relevance in VLSI design. Neither the construction of the optimal layout, nor the proof of its optimality, are obvious. One may interpret this as an indication that the construction of circuits with small total wire length does not only produce circuit architectures that are more interesting from the point of view of physical implementations, but also that this new circuit complexity measure gives rise to a number of interesting new theoretical problems.

References

1. Di Battista, G., Eades, P., Tamassia, R., Tollis, I. G.: Graph Drawing: Algorithms For The Visualization of Graphs Prentice-Hall (New Jersey, USA) (1999)
2. Bhatt, S. N. and Leiserson, C. E. (1982). How to assemble tree machines. *Proceedings of the Fourteenth Annual ACM Symposium on the Theory of Computing*, 77–84.
3. Brent, R. P. and Kung, H.-T. (1980). On the area of binary tree layouts. *Information Processing Letters*, 11(1):46–48.
4. Fischer, M. J. and Paterson, M. S. (1980). Optimal tree layout. *Proceedings of the Twelth Annual ACM Symposium on the Theory of Computing*, 177–189.
5. Fischer, M. J. and Paterson, M. S. (1999). Optimal layout of edge-weighted forests. *Discrete Applied Mathematics*, 90:135–159.
6. Legenstein, R., Maass, W.: Foundations for a circuit complexity theory of sensory processing. Advances in Neural Information Processing Systems **13** (2001) 259–265
7. Mead, C., and Rem M. (1979). Cost and performance of VLSI computing structures. *IEEE J. Solid State Circuits* SC-14(1979):455–462.
8. Paterson, M. S., Ruzzo, W. L., and Snyder, L. (1981). Bounds on minimax edge length for complete binary trees. *Proceedings of the Thirteenth Annual ACM Symposium on the Theory of Computing*, 293–299.

9. Ruzzo, W. L. and Snyder, L. (1981). Minimum edge length planar embeddings of trees. in Kung, H.-T., Sproull, R., and Steele, G. (eds.): *VLSI Systems and Computations*. Computer Science Press (Rockville, Md., USA). pp. 119–123.
10. Savage, J. E.: Models of Computation: Exploring the Power of Computing. Addison-Wesley (Reading, MA, USA) (1998)
11. Ullman, J. D. (1984). *Computational Aspects of VLSI*. Compuer Science Press (Rockville, Md., USA).
12. Yao, A. C. (1981) The entropic limitations of VLSI computations. *Proceedings of the Thirteenth Annual ACM Symposium on the Theory of Computing*, 308–311.