



ELSEVIER

Optimal one-page tree embeddings in linear time

Robert A. Hochberg^{a,*}, Matthias F. Stallmann^b

^a Department of Computer Science, East Carolina University, Greenville, NC 27858-4353, USA

^b Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8207, USA

Received 6 June 2002; received in revised form 19 February 2003

Communicated by S. Albers

Abstract

In the minimum linear arrangement problem one wishes to assign distinct integers to the vertices of a given graph so that the sum of the differences (in absolute value) across the edges of the graph is minimized. This problem is known to be NP-complete for the class of all graphs, but polynomial for trees—algorithms of time complexity $O(n^{2.2})$ and $O(n^{1.6})$ were given by Shiloach [SIAM J. Comput. 8 (1979) 15–32] and Chung [Comput. Math. Appl. 10 (1984) 43–60], respectively. We present a linear-time algorithm for finding the optimal embedding (arrangement) in a restricted but important class of embeddings called *one-page embeddings*.¹

© 2003 Elsevier Science B.V. All rights reserved.

Keywords: Algorithms; Optimal embedding; One-page embedding; Anchored tree

1. Introduction

Given a graph $G = (V, E)$, a *linear arrangement* π of G is a bijection $\pi : V \rightarrow \{1, 2, \dots, n\}$, where $n = |V|$. The *cost* of a linear arrangement π is given by the sum

$$C[\pi, G] = \sum_{(u,v) \in E} |\pi(u) - \pi(v)|,$$

and a *minimum linear arrangement* of G is a linear arrangement which minimizes this sum.

For the class of all graphs, the problem of finding a minimum linear arrangement was shown in 1976 by Garey et al. [4] to be NP-complete. In that same year,

Goldberg and Klipker [6] gave an $O(n^3)$ algorithm that solved the problem when G is a tree. Shiloach [9] improved this bound to $O(n^{2.2})$ in 1979, and Chung [2] further improved it in 1983 to $O(n^\lambda)$ for any λ satisfying $\lambda > \lg 3 \approx 1.6$. Recently, Shahrokhi et al. [8] showed that an algorithm for minimum linear arrangement could be used to find the bipartite crossing number of trees, thus showing that an $O(n^\lambda)$ algorithm also exists for that problem.

In what follows, we use the term *embedding* instead of “linear arrangement”. Hence we will speak of optimal embeddings and one-page embeddings, rather than optimal or one-page linear arrangements.

The present work discusses the problem of embedding trees *on one page*, and gives a linear time algorithm for this restricted problem. This is an improvement, for the case of trees, of a 1988 result of Fredrickson and Hambrusch [3] which gives an optimal

* Corresponding author.

E-mail addresses: hochberg@cs.ecu.edu (R.A. Hochberg), mfms@cs.ncsu.edu (M.F. Stallmann).

¹ A more extensive report is in Robert Hochberg’s thesis [7].

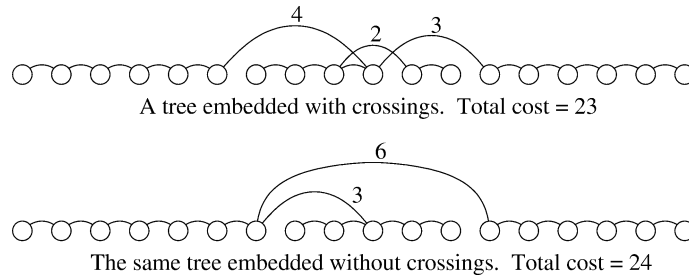


Fig. 1. A tree embedded with and without crossings.

one-page embedding of any outerplanar graph in time $O(n^2)$.

2. One-page embeddings

Let T be a tree and suppose we wish to find an embedding π such that all the edges of T can be drawn on one side of the number line without any pair of edges crossing. See Fig. 1. Such embeddings are called *one-page* embeddings and are a special case of *book embeddings* [1].

An embedding π is a one-page embedding if there do not exist four vertices $a, u, b, v \in T$ such that (a, b) and (u, v) are edges in T , and $\pi(a) < \pi(u) < \pi(b) < \pi(v)$. The following equivalent characterization is more useful for our purposes.

Fact 1. *An embedding π is a one-page embedding if and only if there do not exist four vertices $a, u, b, v \in T$ with vertex-disjoint paths P_1 from a to b and P_2 from u to v , and $\pi(a) < \pi(u) < \pi(b) < \pi(v)$.*

Requiring an embedding to be “one-page” might prevent it from being optimal. In fact, the tree shown in Fig. 1 is such an example: The two embeddings shown are optimal with respect to allowing crossings and disallowing crossings, yet they differ in cost by 1. On the other hand, we give in this paper a linear time algorithm for finding an optimal, one-page embedding of a tree. This is in contrast to the best-known algorithm for general embeddings, which is of time complexity $O(n^{1.6})$.

3. Some observations and theorems

Throughout this paper we will assume that our tree has n vertices.

Let v_* be some vertex of a tree T , and consider the subtrees generated by deleting v_* . Each of them is called a *branch* of v_* , and for each of these branches T_i there is a vertex $v_i \in T_i$, such that the edge $(v_i, v_*) \in T$. The vertex v_i is called the *root of T_i mod v_** . We denote by n_i the number of vertices in the tree T_i . See Fig. 3.

Let T be a tree, let v_* be a vertex of T , and let T_1, T_2, \dots be the branches of v_* . Suppose π is an optimal, one-page embedding of T . We call a vertex v *visible from the top* if there are no edges $(u, w) \in T$ such that $\pi(u) < \pi(v) < \pi(w)$.

Lemma 2. *If vertex v_* is visible from the top, then each branch T_i of v_* occupies a single interval of integers. That is, $\pi(T_i) = \{s, s + 1, \dots, t\}$ for some integers $s \leq t$.*

Proof. Suppose to the contrary that there exist three vertices u, v, w with $\pi(u) < \pi(v) < \pi(w)$ such that $u, w \in T_i$ but $v \notin T_i$. We distinguish two cases:

Case 1: $\pi(u) < \pi(v_*) < \pi(w)$. Since u and w are in the same branch of v_* , there must be a path from u to w which does not include v_* . Let $u = x_1, x_2, \dots, x_k = w$ be such a path, and let j be the least index such that $\pi(x_j) > \pi(v_*)$. Clearly $j > 1$. But then the edge (x_{j-1}, x_j) gives a contradiction to our assumption that v_* is visible from the top.

Case 2: $\pi(v_*) < \pi(u)$ or $\pi(v_*) > \pi(w)$. We assume without loss of generality that $\pi(v_*) < \pi(u)$. Again, since u and w are in the same branch of v_* , there must be a path P_1 between them which lies entirely inside T_i . Since v is in a different branch, there must be a path P_2 from v_* to v which is entirely disjoint from P_1 . But then by Fact 1, we have a contradiction to π being a one-page embedding. \square

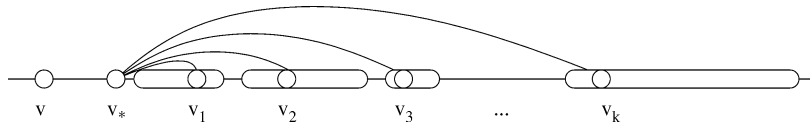


Fig. 2. The proof of Lemma 3.

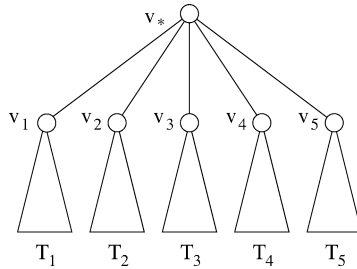


Fig. 3. The branches of a vertex.

We can say more about branches occupying intervals.

Lemma 3. *Let v_* be a vertex of T , not necessarily visible from the top, and suppose T_1, T_2, \dots, T_k are branches of v_* with roots v_1, v_2, \dots, v_k , such that $\pi(v_*) < \pi(v_1) < \pi(v_2) < \dots < \pi(v_k)$. Then each of the branches T_1, T_2, \dots, T_{k-1} occupies a single interval of integers. See Fig. 2.*

Proof. If some branch T_i with $1 \leq i \leq k - 1$ contains a vertex v to the left of v_* , then by Fact 1, the quadruple of vertices v, v_*, v_i, v_k would yield a crossing in our embedding. This observation reduces the proof to case 2 in the proof of Lemma 2. \square

Lemma 4. *The vertices $v = \pi^{-1}(1)$ and $w = \pi^{-1}(n)$ have degree 1 in T .*

Proof. Suppose not. Since v is visible from the top, we can find two branches T_1 and T_2 mod v with $\pi(T_1) = \{2, 3, \dots, i\}$ and $\pi(T_2) = \{i + 1, i + 2, \dots, j\}$. But then we can create a new embedding π' which has lower cost, contradicting the optimality of π . We do this by “flipping” the branch T_1 across to the other side of v . See Fig. 4. Indeed, define $\pi'(x) = i$ if $x = v$, $i + 1 - \pi(x)$ if $x \in T_1$, and $\pi(x)$ otherwise.

All the lengths within each branch remain the same, as does the distance from v to the root of T_1 . But the distance from v to the root of any other branch (including at least T_2) decreases by $i - 1$. This

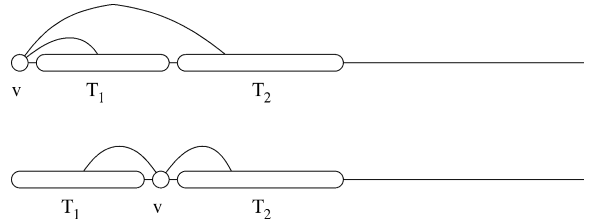


Fig. 4. Proof of Lemma 4. We can flip branch T_1 to the other side of v to obtain an embedding with lower cost.

completes the proof for v . The proof for w is the same. \square

The path between v and w in the lemma above is called the *basic path* of the embedding, and is guaranteed to be monotone since the embedding has no crossings. The vertices on this basic path are exactly those vertices which are visible from the top (see Fig. 5). Lemma 4 is just a special case of a more general theorem, whose proof is essentially the same:

Lemma 5. *If v is a vertex of T , then in any optimal one-page embedding π , the number of branches of v which lie entirely to the left of v is within 1 of the number of branches which lie entirely to the right of v .*

Proof. This is clear if the degree of v is 1. Let us suppose that v has degree at least 2 with at least two more branches lying to its right than to its left. Let T_i be the branch closest to v on the right. By Lemma 3, T_i occupies an interval of integers. If we flip this branch and move it to the left of v , then the only edges which change length are those between v and the roots of its branches. The edges to branches on the right will decrease in length by $|T_i|$ while those to branches on the left will increase by the same amount. Since there are fewer on the left, there will be a net decrease in the cost of the embedding. \square

We now consider the order in which these branches must be embedded in an optimal embedding. The



Fig. 5. An optimal embedding with the basic path highlighted.

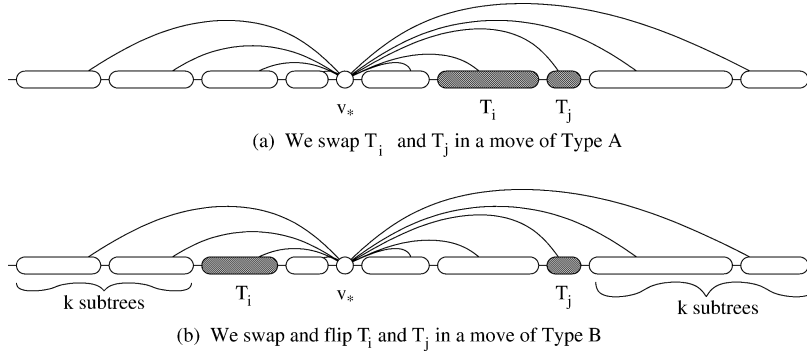


Fig. 6. Proof of Lemma 6. Two ways to rearrange the order of branches without increasing cost.

result is similar to the general, not necessarily one-page embedding case [2,9].

Lemma 6. *Let π be an optimal one-page embedding of a tree T , and let v_* be a vertex on the basic path. Let T_1, T_2, \dots, T_k , be the branches of v_* with $n_1 \geq n_2 \geq \dots \geq n_k$. Then we may assume that the vertex v_* and its branches are embedded in the order*

$$\begin{aligned} (T_1, T_3, \dots, T_k, v_*, T_{k-1}, \dots, T_4, T_2) & \text{ if } k \text{ is odd,} \\ (T_1, T_3, \dots, T_{k-1}, v_*, T_k, \dots, T_4, T_2) & \text{ if } k \text{ is even;} \end{aligned}$$

that is, there is an embedding of T in the form given which is an optimal embedding.

Proof. We prove the claim by modifying the embedding π in several steps to obtain an embedding of the form desired. We also show that each step does not increase the cost of the embedding. We distinguish steps of two types, as illustrated in Fig. 6. In a *type A* step, T_j , the smaller of two neighboring branches on the same side of v_* , is swapped with T_i and moved closer to v_* , decreasing cost by $n_i - n_j$. A *type B* step keeps cost the same but allows us to swap branches on opposite sides of v_* .

We now prove the theorem as follows. First apply steps of type A on each side of v_* to move the branches of greatest size, including T_1 , to the “outside”, furthest away from v_* . Then apply a step of type B, if necessary, so that T_1 ends up in its proper place. To get T_2 to its proper position, first make sure that it is

on the right side of v_* , by applying a step of type B, if necessary. Note that this will not affect the position of T_1 . Then we apply any necessary steps of type A so that T_2 moves to the outside position on the right side. Again, we will not affect the position of T_1 . We note that if there are several trees of sizes n_1 or n_2 then we may be doing some steps that switch branches of the same size, just because of the way we indexed our trees. Proceeding in this fashion, we can sequentially move each T_i to its predicted position without increasing the cost of the embedding, yielding another optimal embedding of the predicted type. \square

We next give a partial converse of Lemma 6 which we need in order to prove that the central vertex, discussed in the next section, lies on the basic path. The proof uses no new ideas, so we give only a sketch.

Lemma 7. *Let v_* be a vertex of degree $d > 1$ on the basic path of an optimal one-page embedding, and let*

$$U_1, U_2, \dots, U_s, v_*, U_{s+1}, \dots, U_d$$

be the order in which its branches are embedded. Then for each i , $1 < i < d$,

$$|U_i| \leq |U_1| \quad \text{and} \quad |U_i| \leq |U_d|.$$

Outline of proof. Suppose not. Then there is some branch U_i such that either $|U_i| > |U_1|$ or $|U_i| > |U_d|$. Let us suppose without loss of generality that $|U_i| >$

$|U_1|$. There are two cases: If U_i lies to the left of v_* , then we swap the two branches, leaving all other branches in their same relative positions. If U_i lies to the right of v_* , then we swap and flip the two branches. In either case, the cost decreases, contradicting the optimality of the embedding. \square

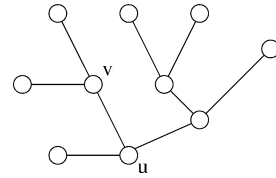


Fig. 7. Branch sizes: $s(u, v) = 3$ and $s(v, u) = 7$.

4. The central vertex

A *central vertex* c of an n -vertex tree T is a vertex such that all of the branches of T mod c have size at most $n/2$ [9]. This differs from a *center* of a tree, a vertex v that minimizes $\max_{u \in T} d(v, u)$. Every tree has exactly one or two central vertices, just as it has one or two centers, though the central vertices need not be the same as the centers.

For every pair of vertices u, v for which (u, v) is an edge of T , define $s(u, v)$ to be the number of vertices in the branch of u which contains v . For example, in Fig. 7, $s(u, v) = 3$, and $s(v, u) = 7$. We note the following:

for any edge $(u, v) \in T$,

$$s(u, v) + s(v, u) = n, \tag{1}$$

for any vertex $u \in T$,

$$\sum_{(u,v) \in T} s(u, v) = n - 1. \tag{2}$$

The next lemma is proved by Shiloach [9], but we present an alternate proof which anticipates the nature of our algorithm.

Lemma 8. *Every tree has a central vertex.*

Proof. Select any vertex v_1 of T . If v_1 is a central vertex, we are done. Otherwise, there is some edge (v_1, v_2) (in fact, exactly one) for which $s(v_1, v_2) > n/2$. Then $s(v_2, v_1) = n - s(v_1, v_2) < n/2$. Is v_2 a central vertex? If so, we are done. If not, then we can find some edge (v_2, v_3) for which $s(v_2, v_3) > n/2$ and $s(v_3, v_2) < n/2$. Is v_3 a central vertex? If so, we are done. If not, then let us continue in this fashion, generating a sequence v_1, v_2, v_3, \dots of vertices. This sequence forms a path in the tree, since $v_i \neq v_{i+2}$ and the tree is acyclic. Clearly it cannot continue until it reaches a leaf, because then $s(v_i, v_{i+1}) = 1 < n/2$. So the process must terminate at some vertex v for which

no edge (v, w) has $s(v, w) > n/2$, and v is a central vertex of our tree. \square

It is not difficult to show that, in fact, every tree has at most two central vertices.² We now come to the theorem which shows why central vertices are relevant to the present problem.

Theorem 9. *In any optimal embedding π of T , any central vertex of T must lie on the basic path.*

Proof. Let x_1, x_2, \dots, x_k be the basic path. By Lemma 4, $s(x_1, x_2) = n - 1$ and $s(x_{k-1}, x_k) = 1$. Thus we can define i as the least index such that $s(x_i, x_{i+1}) \leq n/2$. But then $s(x_{i-1}, x_i) > n/2$ which implies $s(x_i, x_{i-1}) < n/2$. By Lemma 7, x_i is a central vertex. If T has another central vertex, then x_i will have one edge (x_i, v) such that $s(x_i, v) = n/2$. By Lemma 7, this will be edge (x_i, x_{i+1}) , with $s(x_{i+1}, x_i) = n/2$, making x_{i+1} the other central vertex, also on the basic path. \square

5. Anchored trees

Our algorithm for finding an optimal, one-page embedding begins by embedding a central vertex c of the graph, and then embedding its branches according to Lemmas 5 and 6. Care must be taken, however, when embedding these branches. This is because the edge between c and the root of a branch mod c has a length that depends on the embedding of that branch, and this length contributes to the overall cost of the embedding. We are thus led to consider embeddings of *left-anchored trees* and *right-anchored trees*, a fundamental concept also found in previous

² This observation and Lemma 8 were well known prior to Shiloach's paper and are reported, for example, by Geolezian [5].

work [2,9]. They are defined as follows: A left-anchored tree is a tree with a distinguished vertex v_* and an extra edge that joins v_* to a vertex which lies to the left of the embedding and which contributes to the cost of the embedding. Right-anchored trees are defined symmetrically. The next two lemmas describe optimal embeddings of left-anchored trees. They are analogous to Lemmas 5 and 6 for non-anchored trees. The lemmas for right-anchored trees are symmetric.

Lemma 10. *If π is an optimal one-page left-anchored embedding of T with anchor v_* , then the number of branches that lie to the right of v_* is either the same as or one more than the number of branches that lie to the left of v_* .*

Proof. Suppose the branches of v_* are embedded in the order $\{T_1, T_2, \dots, T_s, v_*, T_{s+1}, \dots, T_{s+t}\}$. If $s > t$, then we can flip T_s and move it to the other side of v_* , between v_* and T_{s+1} , to achieve an embedding with lower total cost. Indeed, the cost of embedding each branch will not change, while the sum of the length of the anchor edge and the lengths of the edges from v_* to the branches will decrease by $(s - t)|T_s|$, which is positive. If $t > s + 1$, then we flip the branch T_{s+1} to the other side of v_* to achieve a total lower cost. \square

Lemma 11. *Let π be an optimal one-page embedding of a left-anchored tree T with anchor v_* . Let T_1, T_2, \dots, T_k , be the branches of v_* with $n_1 \geq n_2 \geq \dots \geq n_k$.*

Then we may assume that the vertex v_ and its branches are embedded in the order*

$$\begin{aligned} (T_2, T_4, \dots, T_{k-1}, v_*, T_k, \dots, T_3, T_1) & \text{ if } k \text{ is odd,} \\ (T_2, T_4, \dots, T_k, v_*, T_{k-1}, \dots, T_3, T_1) & \text{ if } k \text{ is even;} \end{aligned}$$

that is, there is an embedding of T in the form given which is an optimal embedding.

Proof. We prove the claim by modifying the embedding π in several steps to obtain an embedding of the form desired, just as in the proof of Lemma 6. Steps of type A are the same, and steps of type B are only slightly different to account for the length of the anchor edge. See Fig. 8.

The rest of the proof proceeds exactly as in the proof of Lemma 6. \square

The following theorem summarizes all that has been said in the preceding lemmas, and completely characterizes optimal one-page embeddings of trees.

Theorem 12 (Main theorem). *Let π be an optimal one-page embedding of a tree T with central vertex c and branches T_1, \dots, T_k , with $n_1 \geq n_2 \geq \dots \geq n_k$ as usual. Then we may make the following assumptions about π :*

- (1) c is on the basic path,
- (2) the branches of c lie as described in Lemma 6,
- (3) each branch of c is embedded as described in Lemma 11.

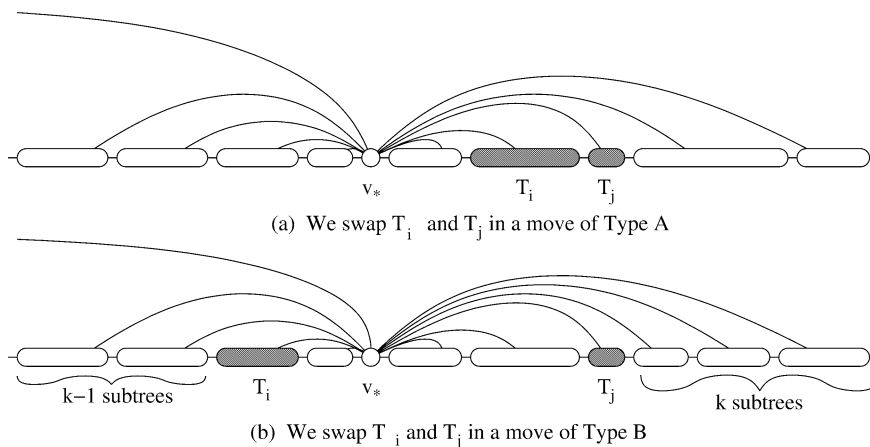


Fig. 8. Proof of Lemma 11. Two ways to rearrange the order of branches without increasing cost.

Proof. All that needs special proof is part (3). To that end we note that the embedding of each branch of c is in fact an anchored embedding. If any of these are not embedded as described in Lemma 11, then we can modify that branch's embedding to obtain an embedding of the desired type without increasing the cost. \square

6. The algorithm

We now give an algorithm for finding an optimal one-page embedding of a tree T in time linear in n , the number of vertices of T . The main algorithm is shown in Fig. 9 and the recursive procedure that embeds branches is in Fig. 10.

Step (1) (in Fig. 9) is done in linear time with a depth-first search from an arbitrary vertex of T using the relations

$$s(u, v) = I + \sum_{\substack{\text{edges } (v, w) \\ w \neq u}} s(v, w)$$

Embed the tree (main program)

procedure EMBED(T)

```
(1) for each edge  $(u, v)$  of  $T$  do
    compute  $s(u, v)$  and  $s(v, u)$  end do
(2)  $c \leftarrow$  a central vertex of  $T$ 
(3) for each vertex  $v$  do
    sort  $v$ 's adjacency list by  $s(v, x)$  end do
    ▷ Let  $n_i$  be the size of  $T_i$ , the  $i$ th branch of  $c$ ,
    ▷ where  $n_1 \geq n_2 \geq \dots \geq n_k$ ,
    ▷ and let  $v_1, \dots, v_k$  be the roots
    ▷ of  $T_1, \dots, T_k$ , respectively
(4)  $leftSum \leftarrow rightSum \leftarrow 0$ 
    for  $i = k$  downto 1
        if  $i$  is even then
            EMBEDBRANCH( $v_i, rightSum, 1$ )
             $rightSum \leftarrow rightSum + n_i$ 
        else ▷  $i$  is odd
            EMBEDBRANCH( $v_i, -leftSum, -1$ )
             $leftSum \leftarrow leftSum + n_i$ 
        endif
    end do
(5)  $\pi(c) \leftarrow leftSum + 1$ ;  $relPos[c] \leftarrow 0$ 
    for each vertex  $v$  do
         $\pi(v) \leftarrow \pi(c) + relPos[v]$  end do
```

Fig. 9. Algorithm for a one-page embedding of the tree T .

and $s(v, u) = n - s(u, v)$. To find a central vertex, step (2), we use the algorithm given in the proof of Lemma 8, also linear time.

Step (3) enables the two critical loops, step (4) in Fig. 9 and the **for** loop in Fig. 10, to be executed in the correct order. A two-pass bucket sort is used to sort the adjacency lists of all vertices at the same time (as described by Yao in 1975 [10]). The first pass sorts all edges (u, v) by $s(u, v)$ —actually two copies of each undirected edge are involved, both (u, v) and (v, u) .

The second pass, a stable sort by u , the first endpoint, extracts the individual adjacency lists maintaining their sorted order. Time is $O(n)$ because both sorts involve keys in the range $[1, n]$.

Step (4) performs the embedding specified in Lemma 6. Positions of all other vertices are initially computed relative to the location of the central vertex c and adjusted in step (5). Because of this, it is important to start embedding the smallest branches, those closest to the central vertex, first.

The procedure EMBEDBRANCH follows Lemma 11 in the same way that step (4) followed Lemma 6. The branches closest to v are embedded recursively first, with special care taken to ensure that the even-numbered (smaller in total size) branches are embedded under the anchor edge. The variable *before* keeps track of the total size of these branches. In case of a

Embedding branches

procedure EMBEDBRANCH($v, base, dir$)

```
▷ Suppose  $v$  has branches  $T_1, \dots, T_k$ 
▷ with sizes  $n_1 \geq \dots \geq n_k$ ,
▷ and roots  $v_1, \dots, v_k$ .
▷ (does not include the branch whose root
▷ is on the path toward the central vertex)
 $before \leftarrow after \leftarrow 0$ 
for  $i = k$  downto 1 do
    if  $i$  is even then
        EMBEDBRANCH( $v_i, base - dir * before, -dir$ )
         $before \leftarrow before + n_i$ 
    else ▷  $i$  is odd
        EMBEDBRANCH( $v_i, base + dir * after, dir$ )
         $after \leftarrow after + n_i$ 
    endif
end do
 $relPos[v] \leftarrow base + dir * (before + 1)$ 
```

Fig. 10. Algorithm for embedding branches.

left anchor, the even branches actually precede v in the embedding, but in a right anchor, they follow v . The variable dir keeps track of this fact; it is 1 for a branch with a left anchor and -1 for a right anchor.

If we consider the **for** loops in step (4) of the main algorithm and the one in `EMBEDBRANCH` together, each edge of the tree is examined exactly once. The recursion engendered by these loops is a depth-first search that starts at the central vertex and considers subtrees in order of increasing size, an ordering that is precomputed in step (3). The total time for this search is linear since each iteration of these loops takes constant time (ignoring the recursive calls). When v is a leaf, the **for** loop in `EMBEDBRANCH` does nothing—there are no further branches to consider. That the remaining loop is linear time is easy to see.

7. Conclusions and future work

We have presented a linear-time algorithm for minimum linear arrangement of trees, *subject to the constraint that the arrangement be a one-page embedding*. This raises interesting questions about minimum linear arrangement and the related problem of minimizing crossings in a two-layer embedding of a bipartite graph.

Can an optimal one-page embedding be used to approximate an optimal linear arrangement? It is not hard to show that a one-page arrangement costs no more than $3/2$ as much as an optimal arrangement that allows crossings. The worst generic examples we have found to date prove a lower bound of $9/8$ on the approximation ratio, but the true worst-case ratio still eludes us. Because the bipartite crossing number of a tree can be much less than the linear arrangement cost—according to Shahrokhi et al. [8] a linear arrangement of a tree T with cost $L(T)$ implies a bipartite embedding with

$$B(T) = L(T) - n + 1 - \sum_{v \in T} \left\lfloor \frac{\deg(v)}{2} \right\rfloor \left\lceil \frac{\deg(v) - 2}{2} \right\rceil$$

crossings—a constant ratio approximation to the minimum linear arrangement does not necessarily imply anything about the approximation ratio for the crossing number.

Can the set of trees for which an optimal one-page embedding is also an unconstrained minimum linear arrangement be succinctly characterized? If so, this work might lead to a linear-time algorithm for minimum linear arrangement of trees. The same question can be asked for outerplanar graphs. In other words, for what subclass of outerplanar graphs does the result of Frederickson and Hambruch [3] yield a minimum linear arrangement?

Acknowledgements

We appreciate the careful reading of an earlier draft by Dr. Carla Savage, her encouragement, and her suggestions for improving the algorithm's proof of correctness. We also thank the two anonymous referees for their suggestions.

References

- [1] F. Bernhart, P.C. Kainen, The book thickness of a graph, *J. Combin. Theory B* 27 (1979) 320–331.
- [2] F. Chung, On optimal linear arrangements of trees, *Comput. Math. Appl.* 10 (1984) 43–60.
- [3] G.N. Frederickson, S.E. Hambruch, Planar linear arrangements of outerplanar graphs, *IEEE Trans. Circuits Systems* 35 (1988) 323–333.
- [4] M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems, *Theoret. Comput. Sci.* 1 (1976) 237–267.
- [5] G.G. Geolezian, On the problem of optimal arrangement of a graph on an interval, *Dokl. Armenian Akad. Nauk* 56 (1973) 269–271.
- [6] M.K. Goldberg, I.A. Klipker, An algorithm for minimal numeration of tree vertices, *Sakharth. SSR Mecn. Acad. Moambe* 81 (1976) 553–556 (in Russian).
- [7] R.A. Hochberg, Minimum linear arrangement of trees, Master's thesis, North Carolina State University, Department of Computer Science, Raleigh, NC, 2002; Also available at <http://www.lib.ncsu.edu/theses/available/etd-06112002-143058/>.
- [8] F. Shahrokhi, O. Sýkora, L.A. Székely, I. Vrto, On bipartite drawing and the linear arrangement problem, *SIAM J. Comput.* 30 (2001) 1773–1789.
- [9] Y. Shiloach, A minimum linear arrangement algorithm for undirected trees, *SIAM J. Comput.* 8 (1979) 15–32.
- [10] A.C. Yao, An $O(|E| \log \log |V|)$ algorithm for finding minimum spanning trees, *Inform. Process. Lett.* 4 (1975) 21–23.