Optimal Algorithms to Embed Trees in a Point Set

Prosenjit Bose* Michael McAllister† Jack Snoeyink‡
Department of Computer Science
University of British Columbia

Abstract

We present optimal $\Theta(n \log n)$ time algorithms to solve two tree embedding problems whose solution previously took quadratic time or more: rooted tree embeddings and degree-constrained embeddings. In the rooted tree embedding problem we are given a rooted tree T with n nodes and a set of n points P with one designated point p and are asked to find a straight-line embedding of T into P with the root at point p. In the degree-constrained embedding problem we are given a set of n points P where each point is assigned a positive degree and the degrees sum to $2n-2$ and are asked to embed a tree in P that respects the degrees assigned to each point of P.

1 Introduction

The problem of deciding whether a set of points admits a certain combinatorial structure, as well as computing an embedding of that structure on the point set, has been a recurrent theme in many fields. The list of problems falling into this category is virtually endless. We mention a few of the structures that are current topics of research.

The triangulation of a point set is a structure that has spurred much research because of its many applications in areas such as finite element methods, graphics, medical imaging, Geographic Information Systems (GIS), statistics, scattered data interpolation, and pattern recognition, to name a few [16, 17]. The *proximity graph* of a point set is a graph in which two points are joined by an edge if they are deemed *close* by some proximity measure. The measure determines the type of graph that results; many different measures of proximity have been defined and studied. Proximity graphs have been studied in the context of pattern recognition, geographic variation analysis, computational geometry, graph theory, and graph drawing. See [10] for a survey of this area.

The combinatorial structure of interest in this paper is the tree, which is well-studied in the literature. For example, the study of spanning trees of a set of points has a long history. From a graph drawing perspective (see [6] for a survey of graph drawing), the traditional questions ask whether a (rooted or free) tree $T = (V, E)$ can be embedded in the plane such that some criterion is satisfied: e.g., that the area of the resulting embedding is small [4, 5, 11], that the symmetry

*Partially supported by an NSERC Postdoctoral Fellowship
†Partially supported by an NSERC Postgraduate Fellowship
‡Partially supported by an NSERC Research Grant and a B.C. Advanced Systems Institute Fellowship.

present in the tree is revealed in the embedding [13] , or that T is isomorphic to the minimum-weight spanning tree [7, 15] or proximity graph [1, 2] of the points in which the vertices are embedded. In essence, the tree is given as input; one needs to construct a set of points in which to embed the tree such that it satisfies the criterion.

The two tree embedding problems that we study have a slightly different perspective: the points are given as input, and the tree may or may not be. We say that an n-node tree $T = (V, E)$ can be *straight-line embedded* onto a set of n points P, if there exists a one-to-one mapping $\phi: V \to P$ from the nodes of T to the points of P such that edges of T intersect only at nodes. That is, edges $(\phi(u_1), \phi(v_1)) \cap (\phi(u_2), \phi(v_2)) = \emptyset$, for all $u_1v_1 \neq u_2v_2 \in E$. We show, in the final section, that to obtain a straight-line embedding of any tree in a set of n points requires $\Omega(n \log n)$ time.

The first problem, called the rooted tree embedding problem, was originally posed by Perles at the 1990 DIMACS workshop on arrangements: Given n points P and an n-node tree T rooted at node ν, can T be straight-line embedded in P with ν at a specified point $p \in P$? Perles showed that this was possible if p was on the *convex hull of* P, which is the smallest convex set containing the points P. Pach and Törőcsik [19] showed that it could if p was not the *deepest point* of P, obtained by repeatedly discarding points on the convex hull. Finally, Ikebe et al. [9] showed that there was always such an embedding. All three algorithms use quadratic time as they run. We show that one can use a deletion-only convex hull structure [3, 8] to obtain $O(n \log^2 n)$ time and then improve this to $\Theta(n \log n)$ time.

The second problem, degree-constrained embedding or dc-embedding, is similar, although the tree T is not specified. Consider a point set $P = \{p_1, p_2, \ldots, p_n\}$ in the plane where each p_i is assigned a positive integral value d_i as its *degree*; the degrees satisfy $\sum_{i=1}^{n} d_i = 2n - 2$. Can some tree T be straight-line embedded on the set of points P such that a tree node of degree d_i maps to point p_i, for all i? Tamura and Tamura [22] (now Ikebe) showed that such a tree always exists and presented an $O(n^2 \log n)$ time algorithm to compute one. We present an optimal $O(n \log n)$ time algorithm for this problem.

Similar embedding problems can be posed for planar graph embeddings in a set of points; these problems remain open.

2 Hull Trees

Our algorithms for embedding trees use segments from the convex hull of the unassigned points as tree edges; the convex hull property prevents our tree edges from intersecting. Consequently, we need efficient access to the convex hull of points. Moreover, we need the ability to delete points from the convex hull as we embed tree vertices at them. Overmars and van Leeuwen [18] describe a data structure to store the convex hull that permits arbitrary insertion and deletion of points into the set of points. In our algorithms, we do not need to insert points into the convex hull but simply delete them; we opt for hull trees [3, 8], which provide better amortized time complexities for point deletions. This section provides a brief introduction to hull trees.

A hull tree of a set P of n points stores the upper or lower convex hull of P in which the entire convex hull of P can be represented by two hull trees. A hull tree for P's upper hull is a binary tree in which each leaf is a point of P and internal nodes represent an edge in the upper hull of the node's leaves (figure 1). Each internal node in the tree stores

- the upper hull edge between the leaves in the node's left subtree and the node's right subtree.
- the number of leaves in its subtree.
- in the dc-embedding problem, each leaf has an degree value given in the problem's input. An internal node to the hull tree stores the sum of the degrees of the leaves in its subtree.

If the hull tree is initially balanced then its height is $O(\log n)$.

We use two of the hull tree operations described by Hershberger and Suri [8]: point deletion and set partition. The point deletion operation removes a point from the hull tree. Each of the hull edges at internal nodes from the point's leaf node to the root of the hull tree may need to be recomputed as a result of the deletion; a bottom-up merge of hulls accomplishes this task.

In the set partition operation, we are given a vertical line and we want to split, or partition, the hull tree into two parts: one hull tree for the points left of the vertical line and one hull tree for the remaining points. Assume that the vertical line goes through a point. The path from the root to the leaf for this point in the hull tree contains all the hull edges that cross the vertical line. Split the hull tree along this path, duplicating the path in each part to maintain connectivity. As with the deletion operation, recompute the hull edges that appear along the path in each part to get two hull trees. Finally, use the point deletion operation to remove the duplicated point from one of the hull trees as necessary.

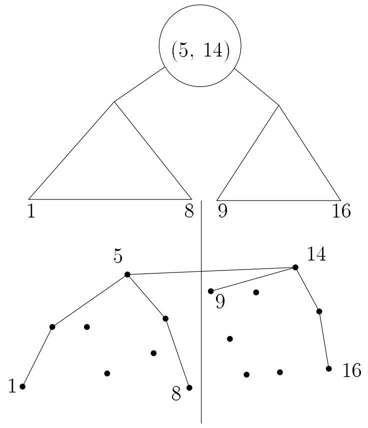

Figure 1: A hull tree node (above) and its upper hull edge (below).

Each of the hull tree operations use $O(\log n)$ amortized time. The set partition operation takes $O(\log n)$ time to divide the hull tree into two parts and to duplicate the path. The remaining time in the set operation is the same as in point deletion; it is the time required to recompute the hull edges along one path. Create a potential function for the hull tree by assigning each internal node a value equal to the number of hull edges that appear above the edge's node. The initial potential of the tree is $O(n \log n)$. When recomputing the hull edges, we either keep the same edge or the replacement hull edge has fewer hull edges below it thus lowering the overall cost of the tree. If we find the replacement hull edge at a node v by walking along the two hulls of the left and right subtrees of v then updating the edges along the path takes $O(\log n + k)$ time where k is the number of hull edges over which we walked and the amount by which the potential function decreases. Consequently, the path update takes $O(\log n)$ amortized time.

3 Embedding a rooted tree with the root on the convex hull

Before embedding a rooted tree in a set of points where any of the points can be designated to embed the root, we consider the simpler problem where the root of the tree is designated to appear at a specified point of the convex hull. In this restricted case, we embed the tree and preserve the order of the children about each tree node. Ikebe et al. [9] and Pach and Törőcsic [19] each provide a quadratic time algorithm whose time can be reduced to $O(n \log^2 n)$. After briefly sketching this lower complexity algorithm, we present an $O(n \log n)$ algorithm.

Ikebe et al. embed a tree T into a set of points P with the root at a point p on $CH(P)$ by locating rays $\ell_0, \ell_1, \ldots, \ell_m$ from p such that there are exactly $|T_i|$ points between ℓ_{i-1} and ℓ_i where $T_1, \ldots, T_m$ are the subtrees of the root of T. The lines $\ell_1, \ldots, \ell_m$ are found by linear time median search. The subtrees $T_1, \ldots, T_m$ are then recursively embedded in the points between adjacent ℓ_i. This leads to an algorithm with a $\Theta(n^2)$ worst-case time.

If the points of P are placed in a convex hull maintenance structure that supports deletions in $O(\log n)$ time [3, 8] then we can find the lines ℓ_i without resorting to a full median search. Let T_L be the leftmost subtree of T and let T_R be $T \setminus T_L$. Assume that $|T_L| \leq |T_R|$. Delete, one at a time, $|T_L|$ points from the convex hull that appear as the left neighbour of p. These are precisely the points between ℓ_0 and ℓ_1. Rebuild the convex hull maintenance structure for the deleted points; recursively embed T_L in the new convex hull structure and T_R in the convex hull structure left after the deletions. The revised complexity of the algorithm is $O(n \log^2 n)$ from the recurrence $T(n) = T(n-k) + T(k) + O(\min(k, n-k) \log n)$ where $1 \leq k \leq n-1$.

The cost of recomputing convex hulls in the variant of Ikebe et al.'s algorithm remains an expensive operation. Our algorithm uses the same notion of isolating the points for one subtree but only uses vertical separation lines and the upper hull of the points, assuming that p lies on the upper hull. When the root of the tree is not in or immediately adjacent to the isolated set of points, the algorithm uses a path along the upper hull to reach the subset. By handling the leftmost or rightmost subtree of the root and then deleting the points used along the upper hull, the algorithm prevents embedded tree edges from crossing. Since all division lines are vertical, the partition operation on hull trees in section 2 divides the hull tree in $O(\log n)$ time and provides a better time complexity.

Theorem 1 *Suppose that we are given an n-node tree T with root node ν and a set of n points P with the point p on the upper hull $UH(P)$. There is an algorithm that takes $O(n \log n)$ time to straight-line embed T in P with ν at point p.*

Proof: Create a balanced hull tree for the upper hull of P and label each point of P with a value of *any*; the label can take the values *left*, *right*, or *any*. When we solve subproblems recursively, the hull trees and labels are not recomputed or reset.

Let T_L be the leftmost subtree of ν, let ν' be the root of T_L (then ν' is the child of ν in T), and let T_R be $T \setminus T_L$. Let ℓ be a vertical line that has $|T_R| - 1$ points of $P \setminus \{p\}$ to its right. Finally, let q be the left neighbour of p on $UH(P)$ as in figure 2.

If ν has degree one then T_R is empty. Embed ν at p. To complete the embedding, let r be the neighbour of p on $UH(P)$ as dictated by p's label, where a label of *any* dictates either neighbour along $UH(P)$. We prove later that there is such a neighbour on $UH(P)$ in the label's direction. Delete p from $UH(P)$ and recursively embed T_L rooted at ν' into the point set $P \setminus \{p\}$ with ν' going to r.

Otherwise ν has degree at least 2 and there are three possibilities for the line ℓ:

- line ℓ intersects segment $\overline{pq}$ as shown in figure 2
- line ℓ is to the left of point q

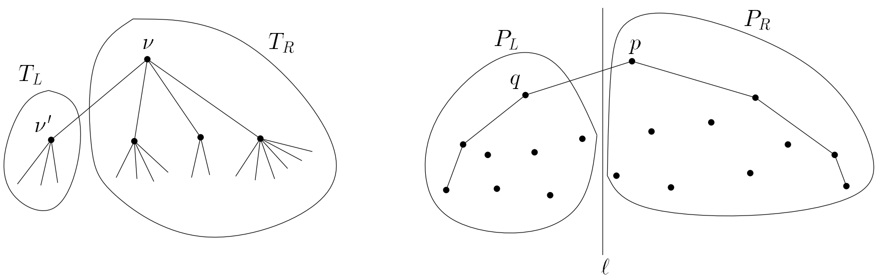

Figure 2: Divided tree T and point set P.

- line ℓ is to the right of point p

If ℓ intersects segment $\overline{pq}$ then partition $UH(P)$ along ℓ into two upper hulls, $UH(P_L)$ to the left of ℓ and $UH(P_R)$ to the right of ℓ. Recursively embed T_L rooted at ν' into the set P_L at point q and embed T_R rooted at ν into the set P_R at point p.

If ℓ lies to the left of q then we shift the root of T and the embedding point in P leftward until we obtain the previous case. The shift assigns point q a label of *right* and recursively embeds T, now rooted at ν', into P at the point q.

We must ensure that ν is eventually embedded at p after the shift. In the recursive call, T_R becomes the rightmost child of ν' in T and the dividing line ℓ' lies to the left of ℓ. Consequently, either $T_L \setminus \nu'$ is embedded completely left of ℓ with ν' still to be embedded at q, or the root of T is shifted left during the embedding of T_L. Assume the former case since the latter case eventually leads to it as the root of T continues to shift left. Then the remaining upper hull to the right of q is identical to $UH(P)$ right of q since all deletions in the embedding of T_L occur to the left of ℓ. In particular, p is still the right neighbour of q on the upper hull. Once $T_L \setminus \nu'$ is embedded, ν' is a leaf with T_R as its only child. The algorithm embeds ν' at q and the label at q, still *right* since the root shifts in T were leftward, generates a recursive call to embed T_R into the remaining points with ν going to point p.

In the final case, ℓ lies to the right of p. If R is the rightmost subtree of ν in T then the vertical line that has exactly $|R|$ nodes to its right lies beyond ℓ so we can embed T rooted at ν into P at point p by descending R rather than T_L as above with left and right interchanged.

Each of the above steps is accomplished in $O(\log n)$ amortized time. When the root is a leaf of the tree, a point gets deleted from the hull tree in amortized $O(\log n)$ time. For the remaining steps, we find the separating line ℓ with a binary search down the hull tree in $O(\log n)$ time. Once we have the separating line, we either partition the point set for T_L and T_R in $O(\log n)$ time to get two hull trees for the recursive subproblems or the root of T is shifted.

Each step either embeds a point of T into P or shifts the root of T is one direction and does not reverse that direction until the root that started the shifting is embedded into P. Each of these actions occurs $O(n)$ times for an overall $O(n \log n)$ time complexity.

Finally, the initial hull tree for $UH(P)$ is computed in $O(n \log n)$ time and the points of P receive their initial labels in $O(n)$ time. ∎

4 Embedding a rooted tree

In this section we simplify the case analysis of Ikebe et al. [9] and apply our new algorithm to compute a straight-line embedding of T in P with the root of T at a specified point. Following Ikebe et al., we do not attempt to preserve the ordering of children at a vertex in this embedding. We begin by improving a quadratic-time algorithm used by previous researchers [19, 9] to embed trees with two nodes mapped to adjacent hull vertices.

Theorem 2 *Suppose that we are given an n-node tree T with distinguished nodes ν and η, and a set of n points P having edge (p, q) on the convex hull $CH(P)$. There is an algorithm that, in $O(n \log n)$ time, embeds T in P with ν at p and η at q.*

Proof: Assume that we have a convex hull maintenance structure for P that supports deletions of hull vertices in amortized $O(\log n)$ time; such a structure can be built initially in $O(n \log n)$ time [3, 8]. If $\nu = \eta$, the algorithm of theorem 1 can complete the embedding in $O(n \log n)$ time. Otherwise, let T_η be the induced subtree of $T \setminus \nu$ that contains η and let T_ν be the complement $T \setminus T_\eta$.

We can find a line through p that cuts off $|T_\nu| - 1$ other points by repeatedly deleting the point of the hull adjacent to p that is different from q. When done, delete p and apply theorem 1 to embed T_ν in the deleted points with ν at p. This takes $O(|T_\nu| \log n)$ time.

Point q is on the hull of the points that remain. Let p' be the hull vertex adjacent to q that is visible from p. Let ν', the child of ν in subtree T_η, be the root of T_η. Recursively embed T_η with ν' at p' and η at q. The total time required for data structure building, point deletion, and tree embedding is $O(n \log n)$. ∎

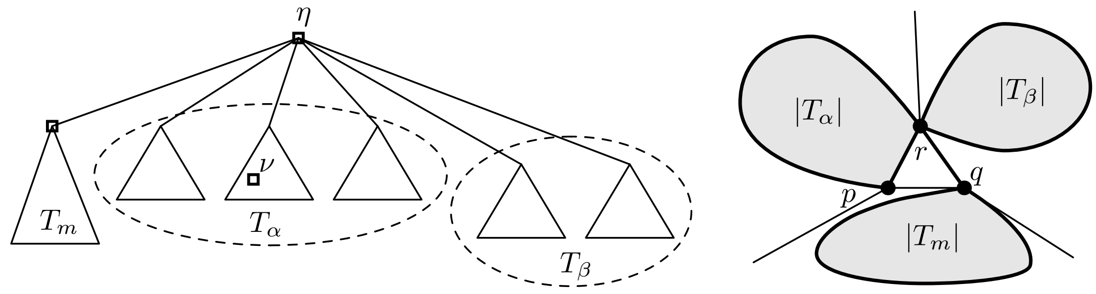

Figure 3: Partitioning T at a centroid η and embedding in P

Now we can embed a rooted tree T in a point set P with the root at a chosen point p. The basic idea is illustrated in figure 3: Use a centroid node to partition T into a subtree T_m and two forests T_α and T_β such that we can find in P the vertices of an empty triangle $\triangle pqr$ with rays from p, q, and r that divide P into convex sets in which T_m, T_α and T_β can be embedded according to theorems 1 or 2. The partition of the tree influences the partition of the point set, and vice versa. Some special cases (when the point p is on the convex hull, the root is a centroid, or a forest is empty) are handled along the way.

Theorem 3 *Given an n-node tree T with root node ν and a set of n points P, we can embed T in P with ν at a chosen point $p \in P$ in $O(n \log n)$ time.*

Proof: If p is on the convex hull $CH(P)$, then we can use the algorithm of theorem 1. Otherwise, sort the points of P radially around p.

Let η be a centroid node of T—that is, if we remove η and its incident edges from T, then we are left with connected subtrees with at most $|T|/2$ nodes [12]. For our purposes, the size of a subtree is the number of nodes other than ν that it contains. Let subtree T_m be a maximum-size subtree of $T \setminus \eta$. We now determine forests T_α and T_β and three rays from p that form angles $\leq \pi$ whose interiors contain $|T_\alpha|$, $|T_m| - 1$, and $|T_\beta|$ points, as in figure 4.

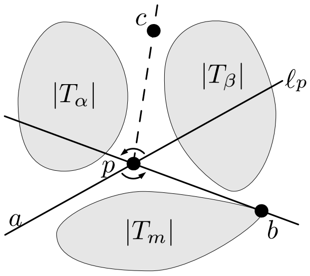

Figure 4: Find bisector ℓ_p, then $\vec{pb}$ and $\vec{pc}$

First, find a line ℓ_p through p that bisects the points—each open halfspace contains $(n-1)/2$ points. To see that such a line exists, consider the integer function $D(\theta)$ whose value is the difference between the number of points in the left and right halfspaces of a line through p at angle θ. By our general position assumption, the value of D changes by ± 1 when the line hits or passes a point. Since $D(\theta) = -D(\theta + \pi)$, the function $D(\theta)$ has a zero.

Second, choose an $a \notin P$ on ℓ_p and $b \in P$ so that the interior of angle $\angle apb$ contains $|T_m| - 1$ points and is as large as possible. There are essentially two choices—a can be chosen on either side of p, and then b is determined as the the $|T_m|$th point counterclockwise around p from a. If there is a point of P on $\vec{pa}$ then perturb a into $\angle apb$. When done, the lines $\overleftrightarrow{pa}$ and $\overleftrightarrow{pb}$ determine two opposite angles as in figure 4: angle $\angle apb$ has $|T_m| - 1$ points not including b, and the opposite has at least $|T_m|$ points.

Third, enumerate the sizes of subtrees of $T \setminus \eta$ as $|T_m|$, $n_1, n_2, \ldots, n_k$, and let $N(i) = 1 + n_1 + n_2 + \cdots + n_i$. Choose a point c in the angle opposite $\angle apb$ that is the $N(i)$th point clockwise from $\vec{pa}$, for some $0 \leq i \leq k$. Such an index i exists because the angle contains at least $|T_m|$ points and $N(j+1) - N(j) = n_j \leq |T_m|$. Now let T_α consist of the subtrees $T \setminus \eta$ with sizes $n_1, \ldots, n_i$ and let T_β be the rest.

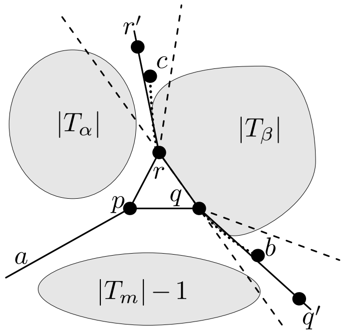

Figure 5: Finding (q, r)

In two special cases we can finish the embedding easily: If $\eta = \nu$, we embed ν into p and embed T_m, T_α, and T_β by the algorithm of theorem 1. If η has degree 2, then T_β is empty—in this case, we embed η into c and embed T_α and T_m into their appropriate angles with (c, b) connecting η to T_m. Whether ν goes with T_α or with T_m, it can be embedded at p according to theorem 2.

Otherwise, determine the convex hull of points inside $\angle bpc$, including b and c but not p. We can assume, without loss of generality, that node ν is in T_m or T_α. Let (q, r) be the hull edge that intersects $\vec{ap}$ and is closer to p. Note that triangle $\triangle pqr$ is empty of points of P, as in figure 5.

Finally, determine $q' \in P$ such that the open region bounded by $\overrightarrow{pa}$, $\overrightarrow{pq}$, and $\overrightarrow{qq'}$ contains $|T_m| - 1$ points; this can be done by sorting points right of $\overleftrightarrow{ap}$ radially around q. The slope of $\overrightarrow{qq'}$ lies between the slopes of $\overrightarrow{rq}$ and $\overrightarrow{pb}$ for the following reasons: If q' is not right of $\overrightarrow{rq}$ (dotted in figure 5) then $\angle aqq'$ does not include any point of P from $\angle cpb$. If $\overrightarrow{qq'}$ does not intersect $\overrightarrow{pb}$, then $\angle aqq'$ includes all points inside $\angle apb$.

Similarly, determine $r' \in P$ such that the interior of $\angle arr'$ contains $|T_\alpha|$ points. The slope of $\overrightarrow{rr'}$ lies between the slopes of $\overrightarrow{pc}$ and $\overrightarrow{qr}$. Thus, the three regions bounded by $\triangle pqr$, $\overrightarrow{pa}$, $\overrightarrow{qq'}$, and $\overrightarrow{rr'}$ are convex.

We can use theorems 1 or 2 to embed T_α, T_m, and T_β in the appropriate regions with ν at p, the root of T_m at q, and η at r. Sorting, computing convex hull structures, and embedding each take $O(n \log n)$ steps. ∎

5 Finding degree-constrained embeddings

A problem similar to the rooted tree embedding of section 3 is to find a tree with non-crossing straight line edges in a set of points when the vertex degree for each point in the plane is given. Tamura and Tamura [22] called this a degree-constrained embedding (dc-embedding), proved that such an embedding exists if and only if the sum of the degrees for n points is $2n-2$, and provided an algorithm to find a dc-embedding in $O(n^2 \log n)$ time. Using hull trees and partitions, we obtain an $O(n \log n)$ time algorithm for the same problem:

Theorem 4 *If we are given n points $P = \{p_1, p_2, \ldots, p_n\}$ where each point is labeled with a positive integer d_i such that $\sum_{i=0}^{n} d_i = 2n-2$ then there is an algorithm that takes $O(n \log n)$ time to find a dc-embedding on P.*

Proof: Assume that the points P are in a deletion only upper hull maintenance structure as in section 2 and that $n > 1$. Assume that each hull edge stores the sum of the degrees and number of points that appear in the edge's subtree within the hull tree. For the convenience of the proof, assume that the names of the points in P are sorted by x-coordinate: $p_i < p_j$ for $i < j$. Finally, let $S(j) = \sum_{i=1}^{j} d_i$.

The upper hull of P falls into one of three categories:

1. there is a point of degree 1 and a point of degree greater than 1 on the hull
2. all points on the hull have degree 1
3. all points on the hull have degree greater than 1

In the first case, there must exist two such points that are adjacent along the hull. Join these points by an edge, delete the point of degree 1 from the hull, and decrease the degree of the other point by 1 (see figure 6a). The rest of the dc-embedding is then built recursively.

If all the hull points have degree 1 then either $n = 2$, or $d_2 \geq 2$, or there exists an index k such that $S(k-1) < 2(k-1) - 1$ and $S(k) \geq 2k - 1$. If $n = 2$ then join the two vertices and stop. If $d_2 \geq 2$ then join p_1 to p_2 by a tree edge, delete p_1 from $CH(P)$, decrease d_2 by 1, and recurse. If $d_2 = 1$ then $S(2) < 2(2) - 1$ and $S(n-1) = 2(n-1) - 1$ so the third condition holds for some $1 < k < n$. By definition, $S(k) = S(k-1) + d_k$ which implies that $d_k \geq 3$. Partition P and $CH(P)$ at p_k with p_k belonging to both smaller sets (see figure 6b). In the left subset,

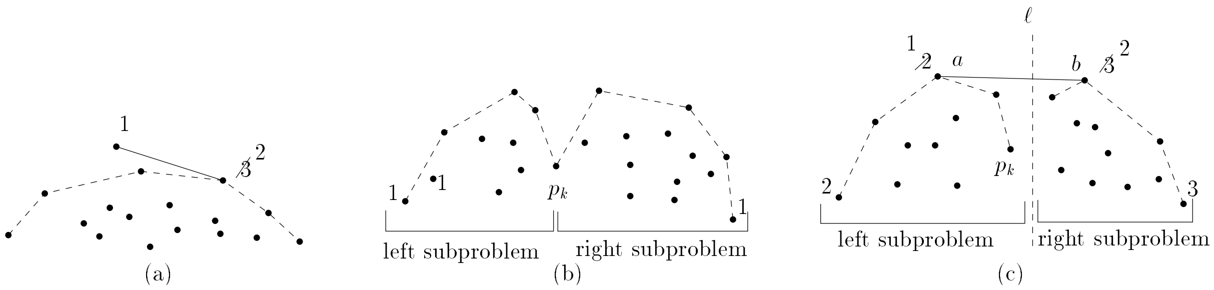

Figure 6: Three cases of recursion for the embedding algorithm.

assign p_k a degree of $2k-2-S(k-1)$ which is at least 1 and at most d_k-1. In the right subset, assign p_k the remaining degree from d_k. Compute a dc-embedding for each subset recursively; the resulting trees will be connected through p_k.

Finally, if all the hull points have degree greater than 1 then there exists an index k such that $S(k) = 2k-1$ since $S(1) > 2(1)-1$ and $S(n-1) \leq 2(n-1)-2$ so the difference between $S(j)$ and $2j-1$ can only decrease in magnitude by 1 whenever p_j has degree 1. Let ℓ be a vertical line between p_k and p_{k+1} and let a and b be the left and right endpoints of the upper hull edge of P that intersects ℓ (figure 6c). Partition P and its upper hull along ℓ, join points a and b by a tree edge, decrease the degrees of points a and b by 1 each, and recursively find the dc-embedding for the subsets of P left and right of ℓ.

The time complexity of each step is $O(\log n)$ amortized time. The index k that satisfies the given conditions is found through a binary search in the hull tree in $O(\log n)$ time. The deletions and hull partitions of the steps are each done in $O(\log n)$ amortized time.

Each of the steps occurs $O(n)$ times since it either embeds a tree edge or partitions the convex hull where the partition vertex becomes a leftmost or rightmost hull vertex and is ineligible for a later partition. ∎

6 Lower Bounds

In this section, we provide $\Omega(n \log n)$ lower bounds on quadratic algebraic decision trees for computing a straight-line embedding of a tree onto a set of n points and for computing a dc-embedding on n points. Theorem 5 establishes the optimality of our algorithms for solving these problems. The same bound can be established with a reduction to the unit cost RAM model used by Paul and Simon [20] for their lower bound on sorting.

Theorem 5 *Finding a straight-line embedding of any tree T with n nodes into a set P of n points requires $\Omega(n \log n)$ time.*

Proof: An Euler tour of a tree T embedded into a set of n points gives a chain on $2n$ points in which no segments cross. A careful implementation of Melkman's algorithm [14] will then compute the convex hull of P in $O(n\ )$time. The $\Omega(n \log n)$ lower bound for computing the convex hull of P [21] implies the same lower bound for the tree embedding problem. ∎

References

[1] P. Bose, G. Di Battista, W. Lenhart, and G. Liotta. Proximity constraints and representable trees. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 340–351. Springer-Verlag, 1995.

[2] P. Bose, W. Lenhart, and G. Liotta. Characterizing proximity trees. Report TR-SOCS-93.9, School of Comp. Sci., McGill Univ., Montreal, Quebec, Canada, 1993.

[3] B. Chazelle. On the convex layers of a planar set. *IEEE Transactions on Information Theory*, IT-31:509–517, 1985.

[4] P. Crescenzi, G. Di Battista, and A. Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Comput. Geom. Theory Appl.*, 2:187–200, 1992.

[5] P. Crescenzi and A. Piperno. Optimal-area upward drawings of AVL trees. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 307–317. Springer-Verlag, 1995.

[6] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom. Theory Appl.*, 4:235–282, 1994.

[7] P. Eades and S. Whitesides. The realization problem for Euclidean minimum spanning trees is NP-hard. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 49–56, 1994.

[8] J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. *BIT*, 32:249–267, 1992.

[9] Y. Ikebe, M. Perles, A. Tamura, and S. Tokunaga. The rooted tree embedding problem into points in the plane. *Discrete & Computational Geometry*, 11:51–63, 1994.

[10] J. W. Jaromczyk and G. T. Toussaint. Relative neighborhood graphs and their relatives. *Proc. IEEE*, 80(9):1502–1517, Sept. 1992.

[11] G. Kant, G. Liotta, R. Tamassia, and I. Tollis. Area requirement of visibility representations of trees. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 192–197, Waterloo, Canada, 1993.

[12] D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, 1973.

[13] J. Manning and M. J. Atallah. Fast detection and display of symmetry in trees. *Congressus Numerantium*, 64:159–169, 1988.

[14] A. Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25:11–12, 1987.

[15] C. Monma and S. Suri. Transitions in geometric minimum spanning trees. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 239–249, 1991.

[16] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, 1992.

[17] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1994.

[18] M. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23:166–204, 1981.

[19] J. Pach and J. Törőcsik. Layout of rooted trees. Technical Report CS-TR-369-92, Princeton, Feb. 1992.

[20] W. Paul and J. Simon. Decision trees and random access machines. Logic and Algorithmics, Monograph 30, L'Enseignement Mathématique, 1987.

[21] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction.* Springer-Verlag, New York, NY, 1985.

[22] A. Tamura and Y. Tamura. Degree constrained tree embedding into points in the plane. *Information Processing Letters*, 44:211–1214, 1992.