Realization of Degree 10 Minimum Spanning Trees in 3-Space

James King*

Abstract

We show that any tree whose maximum degree is at most 10 can be drawn in 3-space such that it is the minimum spanning tree of its vertices.

1 Introduction

This paper investigates the realization problem for minimum weight Euclidean spanning trees, defined as follows: given a tree T, place the vertices of T as points in Euclidean space such that T is the minimum spanning tree of its vertices, or determine that no such placement exists [1]. If such a placement exists in $\mathbb{R}^d$, we say that T can be realized in d-space. We refer to the process of placing the vertices of T as realization.

The realization problem has been solved for the case where $d = 2$. Monma and Suri [2] provide a linear time algorithm for the placement of the vertices of T on a plane when no node in T has degree greater than 5. In the same paper, they prove that no appropriate placement exists if T has any node with degree greater than 6. For a tree T of maximum degree 6, Eades and Whitesides [1] prove that determining whether T can be realized in 2-space is NP-complete.

In the case where the vertices are to be placed in 3-dimensional Euclidean space, the realization problem has not been completely solved. Liotta and Di Battista [3] prove that any tree with maximum degree at most 9 can be realized in 3-space, and that no tree with maximum degree greater than 12 can be realized in 3-space. In the case where a tree T has maximum degree 11 or 12, it is not known whether there exists a polynomial time realization algorithm for T, or even whether every such T can be realized. Previously this was also unknown for trees of maximum degree 10.

In this paper we provide a linear-time algorithm for realizing trees of maximum degree at most 10 in 3-space.

*Department of Computer Science, McGill University, jking@cs.mcgill.ca

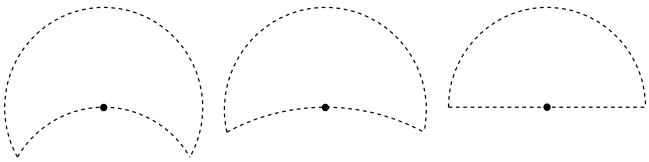

Figure 1: From left to right, cross sections of a 1-cap, a 1/2-cap, and a $1/\infty$-cap.

2 Preliminaries

2.1 Notation

T is the tree we want to realize. V is the vertex set of T. Δ is the maximum degree of any vertex in V. $T(x)$ is the subtree of T rooted at vertex x. $V(x)$ is the vertex set of $T(x)$. (x, y) is the edge between two vertices x and y.

2.2 r-caps

Given some $r < 1$ we now describe a shape that we call an r-cap. Consider a sphere S of radius l centred at c_S and a ball B of radius rl centred at c_B such that c_B is a point on S. Start with B, then remove the region on or within S, not including c_B. What we have now is the r-cap that we denote by $C(r, c_S, c_B)$. We say that the *centre* of this r-cap is c_B. Again, c_B is included in the r-cap but no other point on S is.

2.3 Placement and Restriction

As our realization algorithm runs it *places* vertices and *restricts* vertices to r-caps. When a vertex is placed, its position is fixed and will never change (except in the case where the entire set of placed vertices is translated, rotated, or dilated). If a vertex x is restricted to an r-cap C it means that either x has already been placed in C or x will eventually be placed somewhere in C. If we say that $V(x)$ is restricted to C, it means that every vertex in $V(x)$ is restricted either to C or to another r-cap that lies completely within C.

2.4 The Independence Property

If during the placement of the vertices of T we have a vertex x and its parent p that have both been placed, the significance of $C(r, p, x)$ for some positive $r < 1$ is as follows. If $V(x)$ is restricted to $C(r, p, x)$ and all other

vertices are placed or restricted such that (p, x) is guaranteed to be the shortest edge between a vertex in $V(x)$ and a vertex not in $V(x)$, then we say that $C(r, p, x)$ is independent. If this is the case, we are guaranteed that p and x will be adjacent in MST(V) when all vertices have been placed. Consider in particular a placement of V such that, for every vertex x with a parent p, there is some positive $r < 1$ such that $C(r, p, x)$ is independent. In MST(V) every non-root vertex would necessarily be adjacent to its parent, so MST(V) would be equivalent to T.

2.5 The Construction Lemma

For $r_2 < r_1 < 1$ we say that a vertex x is r_1 *to* r_2 *constructible* if, given an r_1-cap $C(r_1, p, x)$ where $dist(p, x) = l/r_1$, we can place each child x_i of x in $C(r_1, p, x)$ such that

1. $dist(x, x_i) = l(1 - r_2)$
2. for any x_i, x_j with $i \neq j$, $dist(x_i, x_j) \geq l(1 + r_2)$
3. $C(r_2, x, x_i)$ is completely contained within $C(r_1, p, x)$.

It is important to note the following: if we place the children of x to satisfy these conditions, then restrict $V(x_i)$ to $C(r_2, x, x_i)$ for each x_i, the independence of $C(r_1, p, x)$ implies the independence of each $C(r_2, x, x_i)$.

Lemma 1 *(Construction Lemma) For any positive $r \leq 1/2$, any vertex with at most 9 children is r to $r^2/256$ constructible.*

Proof. The proof of this lemma describes a scheme for r to $r^2/256$ construction of a vertex x. We have included it separately as Section 3. □

2.6 The Algorithm

Any nontrivial tree T must have a vertex of degree 1, and we label one such vertex, which we denote by v_1, as the root. We place v_1 at $(0, 0, -2)$ and place its child, denoted v_2, at the origin. We then restrict $V(v_2)$ to $C(1/2, v_1, v_2)$. Now $C(1/2, v_1, v_2)$ is independent and we can call RECURSE$(1/2, v_1, v_2)$, whose pseudocode is included as Algorithm 1.

The independence of $C(r, p, x)$, along with the restriction of each $V(x_i)$ to $C(r^2/256, x, x_i)$, ensures the independence of each $C(r^2/256, x, x_i)$. This means that the algorithm places $V(v_2)$ in such a way that every vertex x in $V(v_2)$ is the centre of an r-cap $C(r, p, x)$ where p is the parent of x and $C(r, p, x)$ is independent for some positive $r < 1$. Therefore the embedding has the property that every vertex of T (excluding the root) is adjacent to its parent in MST(V). So MST(V) is equivalent to T. The proof of the algorithm's correctness is completed by the proof of the Construction Lemma in the following section.

Algorithm 1 RECURSE(r,p,x)

Precondition: $C(r, p, x)$ is independent.

```
if x is a leaf then
  return;
end if
place the children of x in C(r,p,x) as in the proof of
the Construction Lemma;
for all children x_i of x do
  restrict V(x_i) to C(r^2/256, x, x_i);
end for
for all children x_i of x do
  RECURSE(r^2/256,x,x_i);
end for
```

Each call to RECURSE(r, p, x) takes constant time and we make at most one call for each vertex. There is no superlinear overhead, so our algorithm runs in linear time.

One drawback to our algorithm is how quickly the distances between points shrink as we descend down the tree. The reciprocal of the distance between a parent and child grows doubly exponentially with the depth of the parent. In Monma and Suri's 2-space algorithm [2] the function grows slightly less quickly, like $f(x) = c^{x^2}$ for a constant c.

3 Proof of the Construction Lemma

In this section we describe how to r to $256r^2$ construct a vertex x within $C(r, p, x)$, where p is the parent of x. For generality we can assume that x has 9 children; it should be clear that if we can realize any complete 9-ary tree we can realize any tree in which all nodes have at most 9 children (simply realize the complete tree, then remove unwanted vertices). Recall that our root has only one child so no vertex can have more than 9 children if $\Delta \leq 10$.

As our algorithm runs, it can perform any number of dilations, translations, and rotations on the set of placed vertices, since these affine transformations cannot change the topology of MST(V). Note that the transformations will also be applied to the r-caps to which vertices are restricted. We can assume that $C(r, p, x)$ is such that x is located at the origin and p is located at $(0, 0, -1/r)$. If this is not the case we can simply transform the entire set of placed vertices to

make it so before we place the children of x. This will not affect the topology of MST(V).

We place the children of x on the sphere of radius λ centred at the origin, where $\lambda = 1 - r^2/256$. Starting at $\lambda(1,0,0)$ we place children around the 'equator' of this sphere as the consecutive vertices of a regular hexagon on the plane normal to the z-axis. We shall henceforth refer to the first, third, and fifth of these vertices as the *equatorial children* and refer to the second, fourth, and sixth as the *lowered equatorial children*, for reasons that will soon be clear. We place the last 3 children at coordinates $\lambda(0, \sqrt{1/3}, \sqrt{2/3})$, $\lambda(1/2, -\sqrt{1/12}, \sqrt{2/3})$, and $\lambda(-1/2, -\sqrt{1/12}, \sqrt{2/3})$. We will refer to these as the *polar children*.

The 9 children are now placed such that they are all at distance λ from the origin and at distance at least λ from each other, but some are at distance exactly λ from each other. To space out the children of x, we rotate them about the origin in three steps. First we rotate each of the lowered equatorial children towards the negative z-axis by an angle of α. Next we rotate the polar children by an angle of β so that their z-coordinates do not change but they are closer to the lowered equatorial children and farther from the equatorial children. Viewed from the positive z-axis above the points this would simply be a counterclockwise rotation. Finally, we rotate the polar children towards the negative z-axis by an angle of γ so that they are farther away from each other. We must be careful when choosing values for these parameters. If α is too great, the lowered equatorial children of x will be too close to the parent of x. If β is too great, the polar children will be too close to the lowered equatorial children. If γ is too great, the polar children will be too close to the equatorial children and the lowered equatorial children. Appropriate values for the angles are given in the following table.

Parameter	Value
α	$\arcsin\left(\frac{r}{2} - \frac{r^2}{256} - \frac{r^3}{256}\right)$
β	$\arccos\left(\frac{\sqrt{3}(1-\frac{r^2}{16})(1-\frac{2r^2}{256})}{2}\right) - \frac{\pi}{6}$
γ	$\arcsin\left(\frac{1}{\sqrt{3}(1-\frac{2r^2}{256})}\right) - \arcsin\left(\frac{1}{\sqrt{3}}\right)$

It is not difficult to verify that, for each child x_i of x, the angles ensure that $C(r^2/256, x, x_i)$ lies completely within $C(r, p, x)$. This matter is essentially handled by the selection of an adequately small value for α. The more onerous part of proving that our r to $r^2/256$ construction scheme is correct for vertices with 9 children is checking that our rotations space out x's children such that no two are within a distance of $1 + r^2/256$ of each other.

We do not need to check all pairs of these 9 children; we actually only need to check three pairs — for a polar child u, the nearest equatorial child v, and the nearest lowered equatorial child w, we need to check (u, v), (u, w), and (v, w). Symmetry, along with the fact that all three of any kind of child are far enough from each other (this fact can easily be checked), takes care of the rest. The inequalities that we want to prove rely heavily upon the fact that, for our purposes, $0 < r < 1/2$.

Fortunately for the reader there is insufficient space for the calculations, which are neither elegant nor particularly enlightening. Interested readers can see the full calculations online in a preprint version of this paper at `www.cs.mcgill.ca/~jking/papers/spantree.pdf`.

When the children of x are in place they are at distance $1 - r^2/256$ from x and at distance at least $1 + r^2/256$ from each other. Since each $C(r^2/256, x, x_i)$ lies completely within $C(r, p, x)$, this proves the Construction Lemma. As a result, the correctness of our realization algorithm is proven.

4 Conclusions and Future Work

It was previously unknown whether every tree of maximum degree 10 could have its vertices embedded in 3-space such that it was the minimum spanning tree of its vertices. We gave a linear time algorithm for embedding any such tree in this way.

The natural progression is to ask whether or not there exists a polynomial time algorithm for realization of degree 11 trees in 3-space.

Conjecture 1 *There are degree 11 trees that cannot be realized as minimum spanning trees in 3-space.*

Our conjecture is based on the observation that one cannot place 10 points on a hemisphere of radius r such that no two of the 10 points are at distance less than r from each other. When a realization algorithm places a vertex x of a tree T, there is a bounded region in which the vertices in $V(x)$ must be placed. As more vertices of T are placed the regions become more and more restrictive, though each will always contain a hemisphere. We believe there is some finite h such that the complete 10-ary tree of height h cannot be realized because the regions available for building subtrees become too restrictive after a certain depth.

We also believe that our algorithm can be extended to realize trees of higher degree in higher dimensional Euclidean space. After all, our algorithm is essentially an extension of Monma and Suri's algorithm [2] from 2-space to 3-space.

Conjecture 2 *We conjecture that any tree of maximum degree $H(d)+1$ can be realized in d-space, where $H(d)$ is the maximum number of points that can be placed at distance at least r from each other on a d-hemisphere of radius r.*

It seems extremely likely that such realization problems could be solved recursively by extending the notion of independent r-caps to higher dimensions. The main difference in higher dimensions would be how children are spaced out during the recursive construction. 18 vertices of the regular polytope known as the 24-cell [4] can be placed at distance at least r from each other on a 4-hemisphere of radius r. Conjecture 2 therefore implies that any tree of maximum degree 19 can be realized in 4-space.

References

[1] Eades, P., Whitesides, S.: The realization problem for euclidean minimum spanning trees is NP-hard. Algorithmica **16** (1996) 60–82

[2] Monma, C., Suri, S.: Transitions in geometric minimum spanning trees. Discrete & Computational Geometry **8** (1992)

[3] Liotta, G., Di Battista, G.: Computing proximity drawings of trees in the 3-dimensional space. In: 4th Int. Work. Algorithms and Data Structures. Volume 955 of Lecture Notes in Computer Science., Springer (1995) 239–250

[4] Sloane, N.J.A.: Tables of spherical codes. (www.research.att.com/ njas/packings/)