Steiner points in tree metrics don't (really) help

Anupam Gupta *
Computer Science Division
University of California
Berkeley CA 94720.
Email: angup@cs.berkeley.edu

Abstract

Consider an edge-weighted tree $T = (V, E, w : E \to \mathbb{R}^+)$, in which a subset R of the nodes (called the *required* nodes) are colored red and the remaining nodes in $S = V \backslash R$ are colored black (and called the *Steiner* nodes). The shortest-path distance according to the edge-weights defines a metric d_T on the vertex set V.

We now ask the following question: Is it possible to define another weighted tree $T^* = (R, E^*, w^* : E^* \to \mathbb{R}^+)$, this time on *just the red vertices* so that the shortest-path metric d_{T^*} induced by T^* on the vertices in R is "close" to the metric d_T restricted to the red vertices? I.e., does there exist a weighted tree $T^* = (R, E^*, c^*)$ and a (small) constant α such that $d_T(u, v) \leq d_{T^*}(u, v) \leq \alpha\ d_T(u, v)$ for any two red vertices $u, v \in R$?

We answer this question in the affirmative, and give a linear time algorithm to obtain a tree T^* with $\alpha \leq 8$. We also give two applications of this result: an upper bound, in which we show that emulating multicasts using unicasts can be almost as good as general multicasts for certain performance measures; and a lower bound, in which we give a simple combinatorial proof of the fact that the metric generated by a graph of girth g must suffer a distortion of at least $\Omega(g)$ when approximated by a tree.

1 Introduction

Suppose we are given an edge-weighted tree $T = (V, E, w)$, in which a subset $R \subseteq V$ of the nodes are called the *required* nodes and are colored red. The rest of the nodes in $S = V \backslash R$ are colored black and called *Steiner* nodes. The shortest-path distances in the tree (w.r.t. the edge-weights w) define a metric d_T on the vertex set V. We now ask the following question: Is it possible to define a new weighted tree $T^* = (R, E^*, w^*)$ on just the red vertices so that the distances between red points are almost the same in both T and T^*? I.e., does there exist a tree $T^* = (R, E^*, w^*)$ and a small constant α such that the shortest-path metric d_{T^*} of T^* satisfies

$$d_T(u, v) \leq d_{T^*}(u, v) \leq \alpha\ d_T(u, v)$$

for any two red vertices $u, v \in R$? Furthermore, can the weight of T be within a constant factor of the weight of T^*?

It is clear that we cannot preserve all the distances exactly. E.g. consider the star $K_{1,n}$ with unit weight edges, the leaves being the required nodes and the central vertex being the sole Steiner node. In this case, the distances between any two of the nodes in R is exactly 2. It is simple to show that no tree on the n vertices in R can achieve this metric exactly. However, we can approximate distances in R to within a factor of 2 by the star $K_{1,n-1}$. Furthermore, this turns out to be the best we can do, and in Section 6 we show that any graph with n vertices and less than $\binom{n}{2}$ edges cannot approximate distances in the uniform metric to better than a factor of 2.

But the star is a particularly simple tree, and it seems conceivable that preserving distances to within a constant may not be possible for more complex trees. In this paper, we show that this is indeed possible, and our main result is the following:

*Supported by NSF grant CCR-9820951.

THEOREM 1.1. *Given a tree $T = (V, E, w)$ and a set of* required *vertices $R \subseteq V$, there exists a tree $T^* = (R, E^*, w^*)$ such that for all $x, y \in R$,*

$$1 \leq \frac{d_{T^*}(x,y)}{d_T(x,y)} \leq 8. \tag{1.1}$$

Furthermore, the weight of T^* is at most 4 times the weight of T. The tree T^* can be obtained in polynomial time from T.

We will also show how to obtain T^* from T in linear time in Section 5.1. Clearly, the constant in equation (1.1) cannot be less than 2, as the aforementioned example of the star shows. We shall also show a better lower bound of $4(1 - o(1))$ in Section 6.

To the best of our knowledge, the problem of removing Steiner nodes from an arbitrary tree has not been addressed before. An algorithm which works for a special class of trees called k-hierarchically well-separated trees (k-HSTs) was given by Konjevod *et al.* [5].

Applications: This question arises in the relevant and interesting problem of implementing Internet multicasting using unicasts. In multicast transmissions, a routing tree $T = (V, E)$ is defined on the hosts participating in the multicast, as well as the routers that connect these hosts and forward messages. The edges are the physical connections between hosts and routers forming a tree. (The restriction that they form a tree is imposed to keep these protocols simple.) If a multicast is initiated at any host u, the edges of the tree are imagined as being directed *away* from u; when any internal node v in the tree receives a packet from its parent, it makes $\delta(v)$ copies of it, where $\delta(v)$ is its out-degree, and sends one to each child. However, multicasting has proven very difficult to implement for various reasons. One of these is that routing mechanisms on most routers are designed to handle only unicasts, where each router forwards a packet to a unique neighbor; furthermore, since they are implemented in hardware they are difficult and expensive to alter.

On the other hand, routing mechanisms on the hosts are usually implemented in software, and so several researchers [2, 3, 1] have suggested that a virtual tree be defined on just the hosts themselves (where the edges between hosts correspond to paths in the original network) and multicast be implemented on this tree in much the same fashion as described above, using unicast connections to send packets between adjacent hosts. There are several properties that we may want these virtual trees to satisfy: e.g., we may want the total cost of transmissions, which is the same as the weight of the tree, be small, and that the delay for any host to receive any transmission be close to that in the original tree. If we set the hosts to be the nodes in R and the routers to be the Steiner vertices, it suffices to have a virtual tree $T^* = (R, E^*)$ of low weight that preserves the distances in T between the hosts, which is the very question we consider in this paper.

As a very different application of this result, this result also allows us to obtain simple lower bounds on how well distances in graphs can be approximated by trees. Rabinovich and Raz [6] have shown that the unweighted n-cycle $C_n = (V_n, E_n)$ cannot be approximated well by any tree; i.e., for any tree $T = (V, E)$ with $V_n \subseteq V$ and $d_T(x, y) \geq d_{C_n}(x, y)$ for all $x, y \in V_n$, there are two vertices $u, v \in V_n$ which are adjacent in the cycle C_n but are at distance at least $\Omega(n)$ in the tree T. Using Theorem 1.1, we can bypass the topological arguments required in [6] and give a purely combinatorial proof of this result in Section 7.1.

The rest of this paper is organized as follows. In Section 2, we lay down some basic notation and definitions. In Section 3, we will give the algorithm mentioned in the main theorem above and prove the various claimed properties. We then prove some lower bounds in Section 6, and show that we cannot get a constant better than $4(1-o(1))$ in our main theorem. Finally, in Section 7, we end with some details about the two applications mentioned above.

2 Some definitions

In this paper, we only consider finite metrics. Given an edge-weighted graph $G = (V, E, w : E \to \mathbb{R}^+)$, the metric defined by G is $d_G(\cdot,\cdot)$, where $d_G(x, y)$ is defined to be the length of the shortest path between x and y in G (with respect to the edge-weights w). Given a subset $R \subseteq V$, the metric induced by R in G is the restriction $d_G|_{R\times R}$. We often blur the distinction between a graph and the metric generated by it.

Given two metric spaces, (V, ν) and (W, μ), and a map $f : V \to W$, define the following quantities:

$$\|f\| = \max_{x,y \in V} \frac{\mu(f(x), f(y))}{\nu(x,y)};$$
$$\|f^{-1}\| = \max_{x,y \in V} \frac{\nu(x,y)}{\mu(f(x), f(y))}.$$

Then we say that f has *contraction* $\|f^{-1}\|$, *expansion* $\|f\|$ and *distortion* $D(f) = \|f\| \cdot \|f^{-1}\|$. We say that (V, ν) *r-approximates* (W, μ) (or that the *distortion* between μ and ν is at most r) if there exists a map $f : V \to W$ with $D(f) \leq r$. Often we shall consider two graphs $G' = (V', E')$ and $G = (V, E)$ such that $V \subseteq V'$. In such cases, let f be the identity map between the vertices in the metrics $(V, d_{G'}|_{V \times V})$ and (V, d_G), and define both $D(G \to G')$ and $D(G' \to G)$ to be $D(f)$.

3 Removing the Steiner nodes

In this section, we shall show how to remove the Steiner nodes from the given tree $T = (V, E, w)$ to get a tree $T^* = (R, E^*, w^*)$ on just the required nodes, as claimed in Theorem 1.1. We first prove this for the special case when R is the set of leaves $L(T)$ of the tree T; the proof of the general case will follow as a simple corollary of this special case.

Lemma 3.1. *Given a weighted tree T, we can define a tree T^* on the set of leaves $R = L(T)$ of T such that for any $x, y \in L(T)$,*

$$\frac{1}{4} \leq \frac{d_{T^*}(x,y)}{d_T(x,y)} \leq 2.$$

Proof: Let us choose the root for the tree T arbitrarily from its set of internal vertices $V \backslash R$. (If there are no internal vertices, then T is a single edge, and we are done.) This imposes a partial order on the vertices of T, and hence the relations "ancestor" and "descendant" are well-defined. This also allows us to define the *subtree of* v as all the vertices which are descendants[1] of v. We also assume that all distances in the tree are distinct, since any ties can be broken using (say) lexicographic rules.

We use the term "fringe-distance" of a vertex v to mean the distance of the vertex v from the (unique) closest leaf $x \in R$ *lying in its subtree*. We denote this vertex x by $C(v)$, and the fringe-distance of v by $h(v) = d(v, C(v))$. Note that $C(v) = v$ iff v is a leaf.

We give a recursive algorithm to remove Steiner nodes for T: we start with T and remove a set of edges to get a set of smaller trees. These trees are recursively "cleaned" (i.e., the Steiner nodes in all these trees are removed), and then they are joined back together to get a clean tree T^* whose distances are close to those in T.

The Algorithm:

Let T be rooted at a Steiner vertex r. If T has only one required vertex, i.e., it is a path with a required vertex at its end, then T^* is set to be the isolated vertex $C(r)$. Otherwise, we look at all points in T at distance $h(r)/2$ from r. If any such point is not a vertex of T (and lies within an edge), we add a new Steiner vertex in T by subdividing the edge at that point. Let us consider these Steiner vertices $r_1, \ldots, r_k$, and consider the rooted subtrees $T_1, T_2, \ldots, T_k$, with T_i being rooted at r_i. We now delete the edges that lie between r and the r_i's in T. Let us assume that $C(r) \in T_1$. For an example, see figure 1. Applying the above procedure to the tree T on the left gives rise to the trees drawn on the right. In this case, the Steiner vertices r_1, r_2 and r_3 have been added because the cut at distance $h(r)/2$ from the vertex r fell within the edges.

It is easy to see that the vertex $C(r)$ will also be the vertex in R that will be closest to r_1, i.e., $C(r) = C(r_1)$. Further, note that $h(r_i) \geq h(r_1)$. We recursively construct the clean trees T_i^* for each T_i. To obtain the clean tree T^*, we connect $C(r) \in T_1^*$ to each $C(r_i) \in T_i^*$ (for $i \neq 1$) with an edge of length $h(r_i)$.

Before we continue, note that we add new Steiner vertices during the process and have to argue that the process actually terminates. Unfortunately, it is not the case that the number of vertices decreases at each step; nor is it the case that the fringe-distance goes down by a constant fraction at each step. However, it is indeed the case that at least one of these two things happens at each step, and thus this process will terminate in $O(n \log \Delta)$ steps, where Δ is the diameter of T.

The Analysis:

[1] At the risk of causing genealogical havoc, we include v among its descendents.

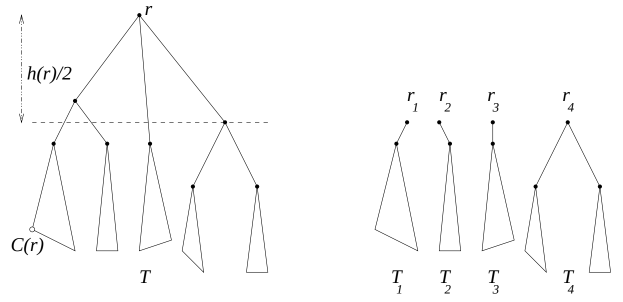

Figure 1: One step of the Algorithm

We will show that T maintains distances between the vertices in R to within a factor of 8. In the following, let d indicate distances in the original tree T, and d^* be the distances in the clean tree T^*. We will first bound the contraction incurred in going from T to T^* by 4, and the expansion by 2.

To show a bound on the contraction, we can use the following simple fact: the contraction of this process is simply the expansion of the reverse process of going from T^* to T. Since it is easy to see that the expansion of a graph is worst for its edges, it suffices to show that the length of each edge $(u, v) \in T^*$ is at least $d(u, v)/4$, and thus the distances in T are at most 4 times those in T^*. It is not difficult to show this, since the edges added to the tree T^* at any step are between $C(r_1)$ and $C(r_i)$ and have a length of $h(r_i)$. However, the distance between $C(r_1)$ and $C(r_i)$ in T is at most $3h(r_1) + h(r_i)$. Now using the fact that $h(r_i) \geq h(r_1)$ gives us that $d(C(r_1), C(r_i)) \leq 4h(r_i)$, which proves the claimed contraction.

Showing that the expansion is bounded is slightly more involved. To prove this, we will need the following technical claim about distances in the cleaned tree.

Claim 4. *Given a tree T with required vertices R and root r, the above algorithm returns a tree T^* on the required vertices such that*

$$d^*(x, C(r)) \leq 2d(x, r) - h(r) \tag{4.2}$$

for any $x \in R$.

Before we prove this claim, let us show that this bound on the distances implies the claimed bound of 2 on the expansion $\|T \to T^*\|$.

Claim 5. *Given a tree T with required nodes R, the above algorithm returns a tree T^* such that*

$$d^*(x, y) \leq 2d(x, y) \tag{5.3}$$

for all $x, y \in R$.

Proof of Claim 5: The proof is by induction on the recursion tree of the algorithm. For the base case, let the tree T have a single required vertex x. In this case, the inequality (5.3) is vacuously satisfied.

Let us now inductively assume that all the pairs of trees (T_i, T_i^*) created by starting with T and performing the cut satisfy the inequality (5.3) above. We shall now prove that it holds for the pair (T, T^*) as well. To prove this, note that when both the vertices lie in the same subtree T_i, the inequality (5.3) holds by the induction hypothesis. Hence we assume that the x and y lie in two different subtrees T_i and T_j respectively.

The distance $d^*(x, y)$ is bounded above by $d^*(x, C(r_i)) + h(r_i) + h(r_j) + d^*(C(r_j), y)$. (It may be less that this if i or j is 1.) However, by Claim 4, we know that $d^*(x, C(r_i)) \leq 2d(x, r_i) - h(r_i)$ and $d^*(y, C(r_j)) \leq 2d(y, r_j) - h(r_j)$. Putting everything together and using the fact that $d(x, r_i) + d(r_j, y)$ is a lower bound for $d(x, y)$, we get that $d^*(x, y) \leq 2d(x, y)$. ■

Let us now complete the argument by proving Claim 4.

Proof of Claim 4: The proof is also by induction on the recursion tree of the algorithm. For the base case, let the tree T have a single required vertex x. In this case, $x = C(r)$ and $d(x, r) = h(r)$, and hence $0 = d^*(x, C(r)) \leq 2d(x, r) - h(r)$.

Let us again inductively assume that all the pairs of trees (T_i, T_i^*) created by starting with T and performing the cut satisfy the inequality (4.2). We now prove that it holds for the pair (T, T^*) as well. There are two cases: when the required vertex x lies in T_1, and when it is in some T_i with $i \neq 1$.

Let x be any required vertex in the tree T_1. Since both x and $C(r) = C(r_1)$ lie in the same subtree T_1, it must be the case that

$$\begin{aligned} d^*(x, C(r)) &\le 2d(x, r_1) - h(r_1) \\ &= 2(d(x, r) - h(r)/2) - h(r_1) \\ &= 2d(x, r) - h(r) - h(r_1) \\ &\le 2d(x, r) - h(r). \end{aligned}$$

where the first inequaility follows from the inductive hypothesis. In the other case, let $x \in T_i$ (for $i \neq 1$). Then we have

$$\begin{aligned} d^*(x, C(r)) &= d^*(x, C(r_i)) + h(r_i) \\ &\le (2d(x, r_i) - h(r_i)) + h(r_i) \\ &= 2d(x, r_i) \\ &= 2(d(x, r) - h(r)/2) = 2d(x, r) - h(r). \end{aligned}$$

which completes the proof of the claim. (The inequality again follows from the inductive hypothesis.) ∎

Combining the bound on the expansion given by Claim 5 with the bound on the contraction completes the proof of the Lemma 3.1. ∎

Note that we can now scale up all the edge-weights in the tree T^* output by the above algorithm by a factor of 4 to ensure that $d(x, y) \le d*(x, y)\ 8d(x, y)$. Using this fact, it is easy to show that the weight of this tree T^* is at most 8 times the weight of T. We can improve this bound slightly in the following lemma, the proof of which is an induction very similar to the ones above and is omitted for lack of space.

LEMMA 5.1. *The weight of the tree T^* defined above is at most 4 times the weight of the tree T'.*

We are now in a position to prove the main theorem of the paper.

Proof of Theorem 1.1: We reduce the general statement to the special case, when R is the set of leaves of T and then use Lemma 3.1. Firstly, note that we can always assume that $L(T) \subseteq R$. Indeed, if any leaf x does not lie in the set R, the edge e_x incident to it cannot lie on any shortest path between two required vertices and hence e_x (along with x) can be deleted from the tree. This process is repeated until $L(T) \subseteq R$.

Now suppose $x \in R - L(T)$; i.e., it is an internal required node, and let it have degree $\delta(v)$. Let us split the vertex x into $\delta(v)$ parts, causing T to split into $\delta(v)$ trees $\{T_i\}$, each having its own incarnation of x. We can now define $R_i = V(T_i) \cap R$. Inductively, these trees can be cleaned to get trees T_i^* with $V(T_i^*) = R_i$. We now identify the incarnations of x in the various T_i^* to get the tree T^*. It can be checked that the distortion $D(T \to T^*)$ is at most $\max D(T_i \to T_i^*)$, which is inductively at most 8. Furthermore, the weights of each of the T_i^* is at most 4 times the weight of T_i, and so the same holds for T^* and T. Finally, the extra work involved can easily be done in linear time, which completes the proof. ∎

5.1 A linear-time algorithm To get a faster algorithm that does not depend on the magnitude of the distances, we modify the algorithm of Lemma 3.1 slightly. Firstly, we can assume that there are no Steiner nodes of degree 2, since the two edges incident on such a node can be fused together. Now note that if the root r of the tree T has only one child r_1, then instead of performing the cut, we can recurse on the subtree T_1 rooted at r_1 to obtain the tree T_1^*. It can be easily checked that this tree T_1^* also satisfies the two inequalities (4.2) and (5.3) required for T.

This gives us the basic idea for the faster algorithm: we delete all edges that intersects the (open) ball of radius $h(r)/2$ around the root r to get the trees $T_1, \dots, T_k$. These have roots r_i' (which are the unique children of the roots r_i used in the earlier procedure). Now we can clean these trees, and then attach each of $C(r_i')$ (with $i > 1$) to $C(r_1') = C(r)$, with edges of length $h(r_i)$ (and *not* $h(r_i')$). An analysis very similar to the one in Lemma 3.1 shows that the distortion is no more than 8 for this algorithm as well.

We can simplify things even further. Note that if T^* is the cleaned tree for T, the length of each edge $\{u, v\}$ in T^* is at least $d_T(u, v)$, and we can reduce it so that it is exactly this value without increasing the distortion. Indeed, as explained above, the contraction of the map from $T \to T^*$ is the expansion of the reverse map $T^* \to T$, and hence is worst for edges of T^*. However, each edge of T^* preserves its distance in T, so there is no contraction at all. Furthermore, since we are reducing the distances in T^*, the expansion of the map $T \to T^*$ can only decrease.

In the composite procedure, we first do a depth first search on the tree starting from the root, and compute all the distances $h(v)$ for each vertex v. We then perform the algorithm as mentioned above, and whenever we attach two vertices $\{u, v\}$ by an edge in T^*, we can simply put $d_T(u, v)$ as the length of the edge. It is not difficult to show that the procedure can be implemented in linear time.

6 Lower bounds

To warm up, let us prove the lower bound of 2 for the star graph $S_n = K_{1,n}$ with unit edge weights, where R is the set of n leaves of the star. We want to show that, for any tree $T = (R, E)$, it is the case that $D(S_n \to T)$ is at least 2. In fact, we can prove a much stronger result as the following proposition shows.

LEMMA 6.1. *If G is an edge-weighted graph on the n vertices in R with fewer than $\binom{n}{2}$ edges, then*

$$D(S_n \to G) \geq 2.$$

Proof: Let us assume (w.l.o.g.) that the edge weights $w(\cdot,\cdot)$ of G satisfy the triangle inequality, and the smallest edge-weight (and hence the smallest distance) in G is exactly 2. Now consider a pair of vertices $\{u, v\}$ which are not connected by an edge in $E(G)$. Such a pair must always exist, since G has less than $\binom{n}{2}$ edges. Since the length of each edge in G is now at least 2, and the shortest path between u and v must have at least two edges, $d_G(u, v)$ is at least 4. This implies that the largest distance in G is at least 4, and hence $D(S_n \to G)$ is at least 2. ■

A more conceptually involved argument given below shows a lower bound of $4(1 - 1/2^k)$ for the complete ternary tree T_k with k levels, where the edge weights are geometrically decreasing.

THEOREM 6.1. *There exists a sequence of weighted trees $T_k = (V_k, E_k)$ and sets $R_k \subseteq V_k$ such that for any k, any tree T'_k with $V(T'_k) = R_k$ and $d_{T'_k} \geq d_{T_k}|_{R_k \times R_k}$ has*

$$d_{T'_k}(u, v) \geq 4(1 - o(1))\ d_{T_k}(u, v)$$

for some vertices $u, v \in R_k$.

Proof: Let T_k denote the ternary tree with k levels having $\frac{1}{2}(3^{k+1} - 1)$ vertices, and root r_k. Equivalently, we can describe T_k as formed by connecting a vertex r_k to three copies of the tree T_{k-1}. Let the length of the edges connecting r_k to the three copies be $\max\{1, 2^{k-2}\}$. Let the set R_k be the leaves of T_k and the internal nodes be the Steiner nodes as usual. Note that the distance from r_k to any leaf of T_k is 2^{k-1}.

Let us look at any tree T on the vertices R_k with no contraction, i.e., $d_T \geq d_{T_k}$. Either the expansion of T is more than 4, in which case we are done; or it is at most 4. In the latter case, we will show that T has diameter at least $4 \cdot (2^k - 1)$. Since the distance between any two points in T_k is at most 2^k, this implies that the distance between some two points must have been blown up by a factor of at least $4 - 1/2^{k-2}$, which will complete the proof of the theorem. We shall prove this by induction on the number of levels k in the tree.

For the base case, consider T_1: this has three non-Steiner vertices, and any tree on three points must be a path with 2 edges. Further, since we no contraction, both the edges must have length at least 2. Thus it has a diameter of at least $4 = 4(2^1 - 1)$.

Let us now inductively assume that the diameter for any non-contracting Steiner-less tree for T_{k-1} with distortion less than 4 is at least $4(2^{k-1} - 1)$. To construct the tree for T_k, look at the three subtrees isomorphic to T_{k-1} hanging off the root r, and let the vertices of R_k in them be called $\{A_i\}_{i=1}^3$. We first claim that any tree T on R_k in which the subgraph induced by any A_i is not connected must have distortion at least 4. To see this, if A_1 is such a set, then there are two vertices of A_1 in T which are connected via vertices in $R_k \backslash A_1$. But the distance between them in T will be at least 2^{k+1}, since T is non-contracting and the distance between any vertex in A_1 to any vertex in $R_k \backslash A_1$ is 2^k. This will give us an expansion of at least 4.

Hence T must consist of three trees $\{T_i\}$ which are joined together by some two edges. Further, the length of these edges must be at least 2^k, since we want no contraction. However, we know that the diameter of each of these trees, since they are assumed to have distortion less than 4, is at least $4(2^{k-1} - 1)$, and hence the eccentricity of each vertex is at least half that value[2]. Now it is simple to see

[2]The *eccentricity* of a vertex u is the largest distance from u to any other vertex in the graph.

that the diameter of T must be at least $2 \cdot 2^k + 2 \cdot 2(2^{k-1} - 1) = 4(2^k - 1)$. ∎

7 Two Applications

7.1 Combinatorial Lower Bounds on distortion: In this section, we show that if ρ is the metric generated by the unweighted n-cycle C_n, and T is any tree on the n vertices of the cycle, then the distortion between ρ and d_T is at least $n-1$. As an consequence of this result and Theorem 1.1, we can show lower bounds on the distortion between metrics generated by graphs of large girth and tree metrics. To the best of our knowledge, these are the first combinatorial proofs of this result.

LEMMA 7.1. *If ρ is the metric generated by the n-cycle $C_n = (V_n, E_n)$, and T is a tree on V_n, then*

$$D(\rho \to T) \geq n - 1.$$

Proof: Since scaling the edge weights does not alter the distortion, let us just focus on trees such that $d_T \geq \rho$. Let the optimal distortion possible be D, and let T be a tree which achieves this distortion and which has minimum total length. Note that if $\{u, v\}$ is an edge in T, the length of $\{u, v\}$ is exactly $\rho(u, v)$. If this is not so, we can reduce the length to be $\rho(u, v)$ without violating the property that $d_T \geq \rho$, and reduce the total length of the edges. We claim that T has only vertices of degree at most two and thus T is a path.

Let the vertices of the cycle be numbered $0, 1, \ldots, n-1$, and let the two *semicircles* of vertex i be the sets of vertices $S(i) = \{i+1, i+2, \ldots, i + \lfloor n/2 \rfloor\}$ and $S'(i) = V_n \backslash (S(i) \cup \{i\})$, where addition is modulo n. Let us assume to the contrary that T has a vertex v of degree 3. In this case, at least two of the edges incident on v are in one of the semicircles of v. Say these edges connect v to i and j, and j is further from v than i is (as shown in the figure 2). Let us remove the edge $\{v, j\}$ and add the edge $\{i, j\}$ instead.

It is easy to check that the resulting graph is a tree. We claim that the distortion of this new tree to ρ has not increased. Indeed, any path which earlier used the edge $\{v, j\}$ can now use the edges $\{v, i\}$ and $\{i, j\}$ and not increase its length, since the sum of their lengths is the same as that of $\{i, j\}$. However,

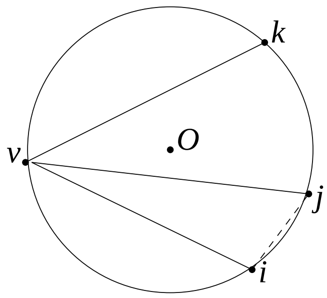

Figure 2: Proof of Lemma 7.1

the sum of the edge lengths has gone down, which contradicts the minimality of the total edge length. Thus all vertices of T must have degree two.

Further, an identical argument shows that if a vertex v has degree 2, then the two edges incident on v cannot both lie in the same semicircle of v. Now consider a pair of adjacent vertices u and v on the cycle such that $\{u, v\}$ is not an edge in T. Such a pair of vertices must exist, since T cannot have all the edges in the cycle. Now to get from u to v using the path T, one must travel a distance at least $(n-1)$. ∎

We can now give combinatorial lower bounds on how well large girth graphs can be approximated by trees. Rabinovich and Raz had proved this and other such results in [6] using topological arguments.

THEOREM 7.1. *For any unweighted graph $G = (V, E)$ with girth g and any tree $T' = (V', E')$ with $V \subseteq V'$,*

$$D(G \to T') \geq (g-1)/8.$$

Proof: Let us assume to the contrary, and suppose that there is a tree $T' = (V', E')$ with $D(G \to T') < (g-1)/8$. Note that the graph G has an isometric cycle C with g vertices. Since the distortion between μ and d_G is less than $(g-1)/8$, it must be the case that the distortion between the metric induced by C (which is the cycle metric on g vertices) and the restriction of $d_{T'}$ to C is also less than $(g-1)/8$.

Now let us assume that the vertices of C in T' are the required vertices, and the rest of the vertices are Steiner vertices. Now, by Theorem 1.1, there exists a tree on the vertices in C which approximates distances within the g-cycle to a factor of less than $8 \cdot (g-1)/8 = (g-1)$, which contradicts Lemma 7.1 and proves the theorem. ■

In a similar vein, a bound of $(g'-1)/8$ can be proved where g' is the length of the longest isometric cycle in G. Similar arguments can be also employed when there are large nearly-isometric cycles in the graph G. As an example, consider the square grid with n vertices. The metric induced by the vertices on the bounding face is within distortion 2 of the cycle metric on $4\sqrt{n}$ vertices. An argument similar to that above gives a lower bound of $\Omega(\sqrt{n})$ on the distortion incurred by approximating the grid by trees.

7.2 End System Multicast: Recall the problem described in the introduction: we are given a graph G of a network, which has as its vertices the hosts and routers in the network. The edges of the graph indicate the physical connections between the hosts and routers, where the length of the edge indicates the time it takes for a packet to be transmitted across the link. Multicast routing protocols define a *routing tree* T on the hosts and routers, and route all packets along the edges of this tree. Here we look at two quantities of interest: the *cost* of the transmission, which is the total transmission time and is equal to the sum of the edge weights in the tree T'; and the *delay*, which is the time it takes for all hosts to receive the transmission.

If we do not want to change the routing mechanisms on the routers, which currently only support unicast transmissions, we would instead want a tree on just the hosts which can be used to mimic multicast using unicasts. Our result says that for any multicast routing tree T, we can define a routing tree T^* on only the hosts, so that both the cost and delay of emulating the multicast transmission using unicast transmissions would only be a constant factor more than in the original routing tree T. In fact, we have an even better guarantee, i.e., distances between *all* pairs of hosts is almost the same as in the original tree, and so the time for a host to receive a transmission is almost the same as before. Further, this all-pairs guarantee is useful for error-recovery: hosts that drop packets could get them from nearby hosts without much extra delay, since the closest vertices in T are also close in T^*.

At this point, we would like to add that if we could afford to define separate routing trees for each host initiating transmissions (the root) and wanted to preserve the root-node distances as well as to minimize the cost of the routing tree, both in an approximate sense, the results of Khuller et al. [4] could be used instead.

8 Acknowledgments

Many thanks to Yuri Rabinovich for suggesting this problem, to Ashwin Nayak and Alistair Sinclair for useful discussions, and to Yatin Chawathe for informing me about end system multicast.

References

[1] Yatin Chawathe. An architecture for Internet content distribution as an infrastructure service. In preparation.

[2] Yang-Hua Chu, Sanjay Rao, and Hui Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics, Santa Clara, CA*, 2000.

[3] Paul Francis. Yallcast: Extending the Internet multicast architecture. `http:www.yallcast.com`.

[4] Samir Khuller, Balaji Raghavachari, and Neal E. Young. Balancing minimum spanning and shortest path trees. *Algorithmica*, 14(4):305–322, 1995.

[5] Goran Konjevod, R. Ravi, and F. Sibel Salman. On approximating planar metrics by tree metrics. Manuscript.

[6] Yuri Rabinovich and Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete & Computational Geometry*, 19(1):79–94, 1998.