A Network-Flow Technique for Finding Low-Weight Bounded-Degree Spanning Trees

Sándor P. Fekete [*][†] Samir Khuller [‡] Monika Klemmstein[*] [§]
Balaji Raghavachari [¶] Neal Young [‖]

Abstract

Given a graph with edge weights satisfying the triangle inequality, and a degree bound for each vertex, the problem of computing a low weight spanning tree such that the degree of each vertex is at most its specified bound is considered. In particular, modifying a given spanning tree T using *adoptions* to meet the degree constraints is considered. A novel network-flow based algorithm for finding a good sequence of adoptions is introduced. The method yields a better performance guarantee than any previously obtained. Equally importantly, it yields the best performance guarantee among the class of algorithms that rely solely on the topology and edge weights of the given tree. The performance guarantee is the following. If the degree constraint $\mathrm{d}(v)$ for each v is at least 2, the algorithm is guaranteed to find a tree whose weight is at most the weight of the given tree times $2 - \min\left\{\frac{\mathrm{d}(v)-2}{\deg_T(v)-2} : \deg_T(v) > 2\right\}$, where $\deg_T(v)$ is the initial degree of v. Examples are provided in which no lighter tree meeting the degree constraint exists. Linear-time algorithms are provided with the same worst-case performance guarantee.

Choosing T to be a minimum spanning tree yields approximation algorithms for the general problem on geometric graphs with distances induced by various L_p norms. Finally, examples of Euclidean graphs are provided in which the ratio of the lengths of an optimal Traveling Salesperson path and a minimum spanning tree can be arbitrarily close to 2.

[*]Center for Parallel Computing, Universität zu Köln, D-50923 Köln, Germany.

[†]E-Mail: `sandor@zpr.uni-koeln.de`.

[‡]Computer Science Department and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported by NSF Research Initiation Award CCR-9307462 and an NSF CAREER Award CCR-9501355. E-mail : `samir@cs.umd.edu`.

[§]E-Mail: `mklemmst@zpr.uni-koeln.de`.

[¶]Department of Computer Science, The University of Texas at Dallas, Box 830688, Richardson, TX 75083-0688. Research supported in part by NSF Research Initiation Award CCR-9409625. E-mail : `rbk@utdallas.edu`.

[‖]Dept. of Computer Science, Dartmouth College, Hanover NH 03755-3510. Part of this research was done while at School of ORIE, Cornell University, Ithaca NY 14853 and supported by Éva Tardos' NSF PYI grant DDM-9157199. E-mail : `ney@cs.dartmouth.edu`.

1 Introduction

Given a complete graph with edge weights satisfying the triangle inequality and a degree bound for each vertex, we consider the problem of computing a low-weight spanning tree in which the degree of each vertex is at most its given bound. In general, it is NP-hard to find such a tree. There are various practical motivations: the problem arises in the context of VLSI layout and network design [8, 12, 20] (such as in the Bellcore software *FIBER OPTIONS*, used for designing survivable optimal fiber networks). The special case of only one vertex with a degree-constraint has been examined [5, 6, 9]; A polynomial time algorithm for the case of a fixed number of nodes with a constrained degree was given by Brezovec et al. [2]. Computational results for some heuristics for the general problem are presented in [14, 19, 21]. Papadimitriou and Vazirani [15] raised the problem of finding the complexity of computing a minimum-weight degree-4 spanning tree of points in the plane. Some geometric aspects are considered in [10, 13, 17].

In this paper, we consider modifying a given spanning tree T, to meet the degree constraints. We introduce a novel network-flow based algorithm that does this optimally in the following sense: if for some algorithm a worst-case performance guarantee can be proved that is solely a function of the topology and edge weights of T, then that performance guarantee also holds for our algorithm. We prove this by showing that our algorithm finds the *optimal* solution for graphs in which the weight of each edge (u, v) equals the cost of the $u \rightsquigarrow v$ path in T.

We also show the following more concrete performance guarantee: If the degree constraint $\mathrm{d}(v)$ for each v is at least 2, our algorithm finds a tree whose weight is at most the weight of the T times

$$2 - \min\Big\{\frac{\mathrm{d}(v) - 2}{\deg_T(v) - 2} : \deg_T(v) > 2\Big\},$$

where $\deg_T(v)$ is the initial degree of v. For instance, the degree of each vertex v can be reduced by nearly half, to $1 + \lceil \deg_T(v)/2 \rceil$, without increasing the weight of the tree by more than 50%. For comparison, note that a factor of 2 is straightforward with standard shortcutting techniques. We also describe *linear-time* algorithms that achieve this ratio.

This performance guarantee is optimal in the sense that for any $D \geq d \geq 2$, if T is a complete rooted $(D-1)$-ary tree with unit edge weights and the edge weights in G are those induced by paths in T, then the weight of any spanning tree with maximum degree d is at least the weight of T times $2 - \frac{d-2}{D-2} - o(1)$.

The restriction $\mathrm{d}(v) \geq 2$ is necessary to obtain constant performance bounds. Consider the case when T is a simple path of unit weight edges, with the remaining edge weights again induced by T. Any spanning tree in which all but one vertex has degree one is heavier than T by a factor of

$\Omega(n)$, the number of vertices in T.

For many metric spaces, graphs induced by points in the space have minimum spanning trees of bounded maximum degree. In such cases our algorithms can be used to find spanning trees of even smaller degree with weight bounded by a factor strictly smaller than 2 times the weight of a minimum spanning tree (MST). For example, in the L_1 metric, a degree-4 MST can be found [17], so that we can find a degree-3 tree with weight at most 1.5 times the weight of an MST. We discuss similar results for the L_1, L_2, and L_∞ norms. For some of these norms, this improves the best current performance guarantees.

Finally, we disprove the following conjecture of [11]: "In Euclidean graphs, perhaps a Traveling Salesman path of weight at most $(2-\varepsilon)$ times the minimum spanning-tree weight always exists..."

Our algorithms modify the given tree by performing a sequence of *adoptions*. Our polynomial-time algorithm performs an optimal sequence of adoptions. Adoptions have been previously used to obtain bounded-degree trees in weighted graphs [10, 16, 18]. The main contributions of this paper are a careful analysis of the power of adoptions and a network-flow technique for selecting an optimal sequence of adoptions. The method yields a stronger performance guarantee and may yield better results in practice. The analysis of adoptions shows that different techniques will be necessary if better bounds are to be obtained.

In the full version of their paper, Ravi et al. [16, Thm. 1.9] (if slightly generalized and improved[1]) gave an algorithm with a performance guarantee of

$$2 - \min\left\{\frac{\mathrm{d}(v)-2}{\deg_T(v)-1} : v \in V, \deg_T(v) > 2\right\}$$

provided each $\mathrm{d}(v) \geq 3$. The performance guarantee of our algorithm is better.

In Euclidean graphs (induced by points in $\mathbb{R}^d$), minimum spanning trees are known to have bounded degree. For such graphs, Khuller, Raghavachari and Young [10] gave a linear-time algorithm to find a degree-3 spanning tree of weight at most 5/3 times the weight of a minimum spanning tree. For points in the plane, the performance guarantee of their algorithm improves to 1.5; if the tree is allowed to have degree four, the ratio improves further to 1.25.

In unweighted graphs, Furer and Raghavachari [4] gave a polynomial-time algorithm to find a spanning tree of maximum degree exceeding the minimum possible by at most one. In arbitrary weighted graphs, Fischer [3] showed that a minimum spanning tree with maximum degree $O(\delta^* + \log n)$ can be computed in polynomial time, where δ^* is the minimum maximum degree of any

[1]To obtain the improved bound one has to change the proof slightly by upperbounding $c(v_1v_2) - c(vv_2)$ by $c(vv_1)$ and not $c(vv_2)$ as is done in [16].

minimum spanning tree. He also provided an algorithm that finds a minimum spanning tree with degree $k(\delta^* + 1)$ where k is the number of distinct edge weights.

2 Adoption

Fix the graph $G = (V, V \times V)$ and the edge weights $w : V \times V \to \mathbb{R}$. The algorithm starts with a given tree T and modifies it by performing a sequence of *adoptions*. The adoption operation (illustrated in Figure 1) is as follows:

$\textsc{Adopt}(u, v)$

Precondition: Vertex v has degree at least two in the current tree.

1 Choose a neighbor x of v in the current tree other than the neighbor on the current $u \rightsquigarrow v$ path.

2 Modify the current tree by replacing edge (v, x) by (u, x).

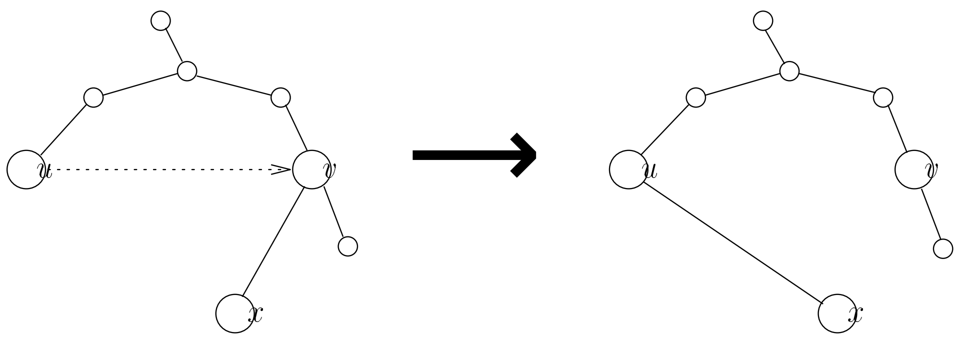

Figure 1: Vertex u *adopts* a neighbor of v

$\textsc{Adopt}(u, v)$ decreases the degree of v by one, at the expense of increasing the degree of u by one and increasing the weight of the tree by $w(x, u) - w(x, v) \leq w(u, v)$.

2.1 The Adoption Network

Definitions 1 *The* deficit *of vertex v with respect to T is* $\deg_T(v) - \mathrm{d}(v)$.

Starting with a given tree, consider a sequence of adoptions $\textsc{Adopt}(u_1, v_1), \textsc{Adopt}(u_2, v_2), \ldots$.

- *The sequence is* legal *if the precondition for each adoption is met.*
- *A sequence is* feasible *if, for each vertex, the change in its degree, i.e., its new degree minus its old degree, is at least its deficit.*
- *The* cost *of the sequence is* $\sum_i w(u_i, v_i)$.

The legal, feasible adoption sequences are precisely those that yield a tree meeting the degree constraints. The cost of a sequence is an upper bound on the resulting increase in the weight of the tree. Our goal is to find a feasible legal sequence of minimum cost. For brevity, we call such a sequence a minimum-cost sequence.

The problem reduces to a minimum-cost flow problem [1] in a flow network that we call the *adoption network* for T. The adoption network is defined as follows. Starting with G, replace each edge (u, v) by two directed edges (u, v) and (v, u), each with cost $w(u, v)$ and infinite capacity. Assign each vertex a demand equal to its deficit.

A *flow* is an assignment of a non-negative real value (called the flow on the edge) to each edge of the network. For each vertex v, the *surplus* at v is the net flow assigned to incoming edges minus the net flow assigned to outgoing edges. A flow is *legal* if the surplus at each vertex is at most one less than its degree. A flow is *feasible* if the surplus at each vertex is at least its demand. The *cost* of the flow is the sum, over all edges, of the cost of the edge times the flow on the edge.

Since the demands are integers, there exists an integer-valued minimum-cost feasible flow [1]. Assuming that each degree constraint is at least 1, their exists such a flow that is also legal. For brevity, we call such a flow a minimum-cost flow.

Lemma 1 *The following statements are true:*

1. *The adoption sequences correspond to integer-valued flows. The correspondence preserves legality, feasibility, and cost.*
2. *The integer-valued flows correspond to adoption sequences. The correspondence preserves legality and feasibility; it does not increase cost.*

Proof: Given a sequence of adoptions, the corresponding flow f assigns a flow to each edge (u, v) equal to the number of times u adopts a neighbor of v. It can be verified that this correspondence preserves legality, feasibility, and cost.

Conversely, given an integer-valued flow f, modify it if necessary (by canceling flow around cycles) so that the set of edges with positive flow is acyclic. This does not increase the cost. Next, order the vertices so that, for each directed edge (u, v) with positive flow, u precedes v in the order. Consider the vertices in reverse order. For each vertex u, for each edge (u, v) with positive flow, have u adopt $f(u, v)$ neighbors of v. It can be verified that this sequence of adoptions preserves legality and feasibility, and does not increase cost. □

3 Polynomial-Time Algorithm

An acyclic, integer, minimum-cost flow can be found in polynomial time [1]. The corresponding legal, feasible adoption sequence can be performed in polynomial time as described in the proof of the second part of Lemma 1. This gives a polynomial-time algorithm.

3.1 Optimality in Tree-Induced Metrics

The following lemma shows that this algorithm is optimal among algorithms that examine only the weights of edges of the given tree.

Lemma 2 *Given a weighted graph $G = (V, E)$ and a spanning tree T such that the weight of each edge in G equals the weight of the corresponding path in T, a minimum-cost sequence of adoptions yields an optimal tree.*

Proof: Fix an optimal tree. Note that the degree of v in the optimal tree is at most $\mathrm{d}(v)$; assume without loss of generality that it is exactly $\mathrm{d}(v)$. For each subset S of vertices, let $\deg_T(S)$ and $\mathrm{d}(S)$ denote the sum of the degrees of vertices in S in T and in the optimal tree, respectively. Define a flow on the edges of T as follows: for each edge (u, v) in T, let $f(u, v) = \mathrm{d}(S_u) - \deg_T(S_u)$, where S_u is the set of vertices that are reachable from u using edges in T other than (u, v), provided $f(u, v)$ as defined is non-negative. Inductively it can be shown that for each vertex v, the net flow into it is $\deg_T(v) - \mathrm{d}(v)$, so that the adoption sequence determined by the flow f achieves a tree with the same degrees as the optimal tree.

We will show that the cost of the flow, and therefore the cost of the adoption sequence, is at most the difference in the weights of the two trees. This implies that the tree obtained by the adoption sequence is also an optimal tree.

To bound the cost of the flow, we claim that the flow is "necessary" in the following sense: for each edge (u, v) in T, at least $f(u, v) + 1$ edges in the optimal tree have one endpoint in S_u and the other in $V - S_u$. To prove this, let c be the number of edges in the optimal tree crossing the cut $(S_u, V - S_u)$. Note that $\deg_T(S_u) = 2(|S_u| - 1) + 1$. Since the optimal tree is acyclic, the number of edges in the optimal tree with both endpoints in S_u is at most $|S_u| - 1$. Thus $\mathrm{d}(S_u) \leq 2(|S_u| - 1) + c = \deg_T(S_u) - 1 + c$. Rewriting gives $c \geq \mathrm{d}(S_u) - \deg_T(S_u) + 1 = f(u, v) + 1$. This proves the claim.

To bound the cost of the flow, for each edge (u, v), charge $w(u, v)$ units to each edge in the optimal tree crossing the cut $(S_u, V - S_u)$. By the claim, at least the cost of the flow, plus the cost of T, is charged. However, since the cost of each edge in the optimal tree equals the weight of the corresponding path in T, each edge in the optimal tree is charged at most its weight. Thus, the net charge is bounded by the cost of the optimal tree. $\square$

Note that given the exact degrees of the desired tree (for instance, if the degree constraints sum to $2(|V|-1)$), the optimal flow in Lemma 2 can be computed in linear time.

3.2 Worst-Case Performance Guarantee

The next theorem establishes a worst-case performance guarantee for the algorithm in general graphs satisfying the triangle inequality.

Theorem 3 *Given a graph $G=(V,E)$ with edge weights satisfying the triangle inequality, a spanning tree T, and, for each vertex v, a degree constraint $\mathrm{d}(v) \geq 2$, the algorithm produces a tree whose weight is at most the weight of T times*

$$2-\min\left\{\frac{\mathrm{d}(v)-2}{\deg_T(v)-2} : v \in V, \deg_T(v) > 2\right\}.$$

Proof: The increase in the cost of the tree is at most the cost of the best sequence. By Lemma 1, this is bounded by the cost of the minimum-cost flow. We exhibit a fractional feasible, legal flow whose cost is appropriately bounded. The minimum-cost flow is guaranteed to be at least as good.

Root the tree T at an arbitrary vertex r. Push a uniform amount of flow along each edge towards the root as follows. Let $p(v)$ be the parent of each non-root vertex v. For a constant c to be determined later, define

$$f(u,v) = \begin{cases} c & \text{if } v = p(u) \\ 0 & \text{otherwise.} \end{cases}$$

The cost of the flow is c times the weight of T. Let v be any vertex. The surplus at v is at least $c(\deg_T(v)-2)$. We choose c just large enough so that the flow is feasible.

There are three cases. If $\deg_T(v)=1$, the deficit at v will be satisfied provided $c \leq 1$ and $\mathrm{d}(v) \geq 2$. If $\deg_T(v)=2$, the deficit at v will be satisfied provided $\mathrm{d}(v) \geq 2$. For $\deg_T(v) > 2$, the deficit will be satisfied provided

$$c \geq \frac{\deg_T(v)-\mathrm{d}(v)}{\deg_T(v)-2} = 1-\frac{\mathrm{d}(v)-2}{\deg_T(v)-2}.$$

Thus, taking

$$c = 1-\min\left\{\frac{\mathrm{d}(v)-2}{\deg_T(v)-2} : v \in V, \deg_T(v) > 2\right\}$$

gives the result. $\square$

4 Optimality of Performance Guarantee

In this section, we show that the worst-case performance guarantee established in Theorem 3 is the best obtainable.

Lemma 4 *Consider an n-vertex weighted graph G with a spanning tree T such that the weight of each edge in T is 1 and the weight of each remaining edge is the weight of the corresponding path in T. If T corresponds to a complete rooted $(D-1)$-ary tree, then the weight of any spanning tree with maximum degree d is at least the weight of T times*

$$2 - \frac{d-2}{D-2} - o(1),$$

where $o(1)$ tends to 0 as n grows.

Proof: Fix any spanning tree T' of maximum degree d. Let S_i denote the vertices at distance i from the root in T. Consider the flow on the edges of T corresponding to T', as defined in the proof of Lemma 2. The proof shows that at least $|S_i|(d-D)-1$ units of flow cross the cut $(V - S_i, S_i)$. Thus the net cost of the flow is at least $\sum_{i=0}^{k-1} |S_i|(d-D)-1$. The cost of T is $\sum_{i=0}^{k-1} |S_{i+1}| - |S_i|$. Since $|S_{i+1}| = |S_i|(D-1)+1$, so $|S_{i+1}| - |S_i| = |S_i|(D-2)+1$, the ratio of the cost of the flow to the cost of T is at least

$$\frac{\sum_{i=0}^{k-1} |S_i|(d-D)-1}{\sum_{i=0}^{k-1} |S_i|(D-2)+1}.$$

Simplifying shows that the ratio is at least $(d-D)/(D-2) - o(1)$. Since the ratio of the cost of T' to the cost of T is 1 more than this, the result follows. $\square$

Next we observe that the $\mathrm{d}(v) \geq 2$ constraint is necessary to obtain any constant performance guarantee:

Lemma 5 *Consider an n-vertex weighted graph G with a spanning tree T such that the weight of each edge in T is 1 and the weight of each remaining edge is the weight of the corresponding path in T. If T corresponds to a path of length n with endpoint r, then the weight of any spanning tree in which each vertex other than r has degree 1 is at least the weight of T times $n/2$.*

The proof is straightforward.

5 Linear-Time Algorithms

Note that to obtain the worst-case performance guarantee a minimum-cost flow is not required. It suffices to find a feasible integer flow of cost bounded by the cost of the fractional flow f defined in the proof of Theorem 3. We describe two methods to find such a flow, and to implement the corresponding sequence of adoptions, in linear time.

Algorithm 1: Let f be the fractional flow defined in Theorem 3. Modify f by repeatedly performing the following short-cutting step: choose a maximal path in the set of edges with positive flow; replace the (c units of) flow on the path by (c units of) flow on the single new edge (u, v), where the path goes from u to v. Let $q(u)$ be the child of v on the path. Stop when all paths have been replaced by new edges. This phase requires linear time, because each step requires time proportional to the number of edges short-cut.

In the resulting flow, the only edges with positive flow are edges from leaves of the (rooted) tree T to interior vertices. Round the flow to an integer flow as follows. Consider each vertex v with positive deficit, say D. Using a linear-time selection algorithm, among the edges (u, v) sending flow to v, find the D smallest-weighted edges. Assign one unit of flow to each of these D edges. The resulting flow is integer-valued, feasible, legal, and has cost bounded by the cost of f. This phase requires linear time.

Assume that each vertex maintains a doubly linked list of its children. Given a pointer to any vertex, we can obtain its sibling in constant time. As adoptions are done, this list is maintained dynamically. Perform the adoptions corresponding to the flow in any order: for each edge (u, v) with a unit of flow, have u adopt the right sibling of $q(u)$ (in the original tree T). The tree remains connected because $\mathrm{d}(v) \geq 2$, so at least one child of v is not adopted.

Algorithm 2: Consider the following restricted adoption network. Root the tree T as in the proof of Theorem 3. Direct each edge (u, v) of the tree towards the root. (Non-tree edges are not used.) Assign each edge a capacity of 1 and a cost equal to its weight. Assign each vertex a demand equal to its deficit.

We show below that an integer-valued minimum-cost flow in this network can be found in linear time. Because the fractional flow defined in the proof of Theorem 3 is a feasible legal flow in this network, the minimum-cost flow that we find is at least as good.

Find the flow via dynamic programming. For each vertex v, consider the subnetwork corresponding to the subtree rooted at v. Let $C_j(v)$ denote the minimum cost of a flow in this subnetwork such that the surplus at v exceeds its demand D by j, for $j = 0, 1$. For each child u of v, Let $\delta(u)$

denote $w(u,v)+C_1(u)-C_0(u)$ — the additional cost incurred for v to obtain a unit of flow along edge (u,v). Let U_j denote the $D+j$ children with smallest $\delta(u)$, for $j=0,1$. Then, for $j=0,1$,

$$C_j(v) = \sum_{u \in U_j} \delta(u) + \sum_u C_0(u).$$

Using this equation, compute the C_j's bottom-up in linear time. The cost of the minimum-cost flow in the restricted network is given by $C_0(r)$, where r is the root. The flow itself is easily recovered in linear time.

To finish, shortcut the flow as in the first phase of the previous algorithm and perform the adoptions as in the last phase of that algorithm.

6 Geometric Problems

Our general result has several implications for cases of particular distance functions where it is possible to give a priori bounds on the maximum degree of an MST. For the case of L_2 distances in the plane, there always is an MST of maximum degree 5[13]; for the case of L_1 or L_∞ distances there always exists a MST of maximum degree 4 [13, 17]. Without using any specific structure of the involved distance functions, we note as a corollary:

Corollary 6.1 *Let $T_{\min}$ be an MST and T_k be a tree whose maximal degree is at most k. For L_1 or L_∞ distances in $\mathbb{R}^2$, we get a degree-3 tree T_3 with*

- $w(T_3) < \frac{3}{2} w(T_{\min})$.

For the case of Euclidean distances in the plane, we get bounded degree trees that satisfy

- $w(T_3) < \frac{5}{3} w(T_{\min})$
- $w(T_4) < \frac{4}{3} w(T_{\min})$.

The latter two bounds are worse than those shown by Khuller, Raghavachari and Young [10] using the geometry of point arrangements. (It was shown that $\frac{3}{2}$ and $\frac{5}{4}$ are upper bounds.) We conjecture that the following are the optimal ratios:

Conjecture 6.2 *For the case of Euclidean distances in the plane, we conjecture that there exist bounded degree trees that satisfy*

- $\frac{w(T_3)}{w(T_{\min})} \leq \frac{\sqrt{2}+3}{4} \approx 1.103\ldots$

- $\frac{w(T_4)}{w(T_{\min})} \leq \frac{2\sin(\frac{\pi}{10})+4}{5} \approx 1.035\ldots$

For L_1 and L_∞ distances in $\mathbb{R}^2$, we conjecture

- $\frac{w(T_3)}{w(T_{\min})} \leq \frac{5}{4}$

The best known lower bounds (yielding the ratios of Conjecture 6.2) are shown in Figure 2. (Note that the example for L_∞ metric is obtained by rotating the arrangement in (c) by 45 degrees.)

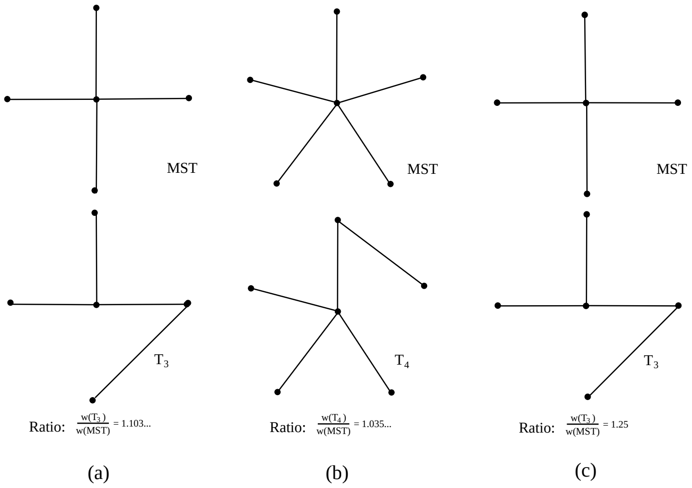

Figure 2: The worst known examples for: (a) $\frac{w(T_3)}{w(T_{\min})}$, L_2 distances
(b) $\frac{w(T_4)}{w(T_{\min})}$, L_2 distances
(c) $\frac{w(T_3)}{w(T_{\min})}$, L_1 distances.

6.1 Geometric Hamiltonian Paths

We conclude this paper by settling a question raised in [11], in the negative:

"In Euclidean graphs, perhaps a Traveling Salesman path of weight at most $(2-\varepsilon)$ times the minimum spanning-tree weight always exists and can be found in polynomial time."

Theorem 6.3 *For an arrangement of points in the plane with Euclidean distances, the ratio $\frac{w(T_2)}{w(T_{\min})}$ can be arbitrarily close to 2.*

Proof: Let n and k be sufficiently large. Construct a point set as follows (see Figure 3):

Take base points at $(0,0)$ and $(2n^k, 0)$.

For $j = 0, \ldots, k$, add points as follows:

For $i = 1, \ldots, n^j$

Add *level* j point at $((2i-1)n^{k-j}, n^{k-j})$;

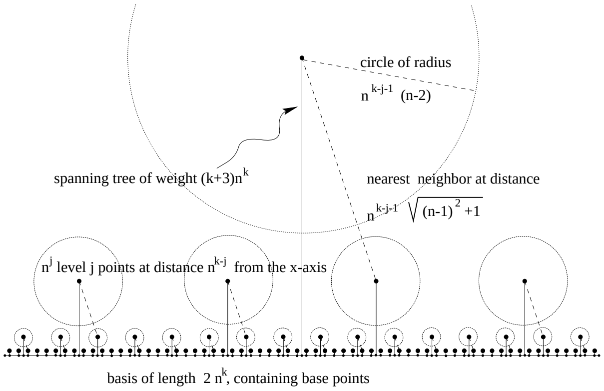

Figure 3: A class of examples showing $\frac{w(T_2)}{w(T_{\min})} \to 2$

add *base* point at $((2i-1)n^{k-j}, 0)$.

The points at level j, i.e., at height n^{k-j} have nearest neighbors at distance at least $n^{k-j-1}(n-1)$. To prove the lower bound, we draw a circle centered at each point at level $j < k$. For the points at level j, the radius of the circle is $(n^{k-j-1}(n-2))$. The circles corresponding to two points do not intersect. Since each point has degree two in a Hamilton cycle, twice the sum of the radii of the circles gives us a lower bound on the length of the Hamilton cycle. This can be computed as follows (observe that we can always pick $n \geq 2k$).

$$2\sum_{j=0}^{k-1} n^j(n^{k-j-1}(n-2)) = 2kn^{k-1}(n-2) \geq 2(k-1)n^k.$$

Since no edge can have length more than $2n^k$, we conclude that no Hamilton path can have a weight smaller than $2(k-1)n^k - 2n^k = 2(k-2)n^k$.

It can be verified that there is a tree of weight $(k+3)n^k$ that spans the points. Hence this is an upper bound on the weight of $T_{\min}$. It follows that $\frac{w(T_2)}{w(T_{\min})} > \frac{2(k-2)}{k+3}$, which can be arbitrarily close to 2, concluding the proof. $\square$

The above class of examples establishes the same lower bound for L_1 and L_∞ distances.

Acknowledgements: We thank Joe Mitchell for establishing the transatlantic connection between the authors. We thank Chandra Chekuri for asking us about degree 3 trees in the L_1 metric.

References

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin. Network Flows (Theory, Algorithms and Applications). *Prentice Hall*, (1993).

[2] C. Brezovec, G. Cornuéjols, F. Glover. A matroid algorithm and its application to the efficient solution of two optimization problems in graphs. *Mathematical Programming*, **42** (1988), pp. 471-487.

[3] T. Fischer. Optimizing the degree of minimum weight spanning trees. TR 93-1338, Dept. of Computer Science, Cornell University, April 1993.

[4] M. Fürer and B. Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal. *Journal of Algorithms,* **17** (3):409–423, 1994.

[5] H. N. Gabow. A good algorithm for smallest spanning trees with a degree constraint. *Networks* **8** (1978), pp. 201-208.

[6] H. N. Gabow and R. E. Tarjan. Efficient algorithms for a family of matroid intersection problems. *J. Algorithms* **5** (1984), pp. 80-131.

[7] M. R. Garey and D. S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. *Freeman, San Francisco*, (1979).

[8] B. Gavish. Topological design of centralized computer networks – formulations and algorithms. *Networks* **12** (1982), pp. 355-377.

[9] F. Glover, D. Klingman. Finding minimum spanning trees with a fixed number of links at a node. In: B. Roy (ed.), *Combinatorial Programming: Methods and Applications.* D. Reidel Publishing Company, Dordrecht-Holland, 1975. pp. 191-201.

[10] S. Khuller, B. Raghavachari and N. Young. Low degree spanning trees of small weight. *Proc. of 26th Annual Symp. on Theory of Computing*, pp. 412–421, May 1994. To appear in *SIAM Journal on Computing.*

[11] S. Khuller, B. Raghavachari, N. Young, Balancing Minimum Spanning Trees and Shortest-Path Trees. *Algorithmica*, **14** (4), 305–321, 1995.

[12] C. L. Monma, D. Shallcross. Methods for designing communication networks with certain two-connected survivability constraints. *Oper. Res.* **37** (1989), pp. 531-541.

[13] C. L. Monma, S. Suri, Transitions in geometric minimum spanning trees. *Discrete & Computational Geometry*, **8** (3), 265-293, 1992.

[14] S. C. Narula and C. A. Ho. Degree-constrained minimum spanning tree. *Comput. Ops. Res.* **7** (1980), pp. 239-249.

[15] C. H. Papadimitriou, U. V. Vazirani, On two geometric problems related to the traveling salesman problem. *J. Algorithms*, **5**, 231-246, 1984.

[16] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz and H. B. Hunt III. Many birds with one stone: multi-objective approximation algorithms. Manuscript. A preliminary version appeared in *Proc. 25th Annual ACM Symp. on the Theory of Computing*, pp. 438–447, May 1993.

[17] G. Robins and J. S. Salowe. Low-degree minimum spanning trees. *Discrete and Computational Geometry*, **14** (2), 151-166, 1995.

[18] J. S. Salowe. Euclidean spanner graphs with degree four. *Discrete Applied Mathematics*, **54** (1), 55-66, 1994.

[19] M. Savelsbergh and A. Volgenant. Edge exchanges in the degree-constrained minimum spanning tree problem. *Comput. Ops. Res.* **12** (1985), pp. 341-348.

[20] M. Stoer. *Design of survivable networks.* # 1531 Lecture Notes on Mathematics. Springer, Heidelberg, 1992.

[21] A. Volgenant. A Lagrangean approach to the degree-constrained minimum spanning tree problem. *Europ. J. Ops. Res.* **39** (1989), pp. 325-331.