Optimizing the Degree of Minimum Weight Spanning Trees

Ted Fischer*

TR 93-1338
April 1993

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Department of Computer Science, Cornell University, Ithaca, NY 14853. Research supported by ONR Graduate Fellowship.

Optimizing the Degree of Minimum Weight Spanning Trees

Ted Fischer*

April 15, 1993

Abstract

This paper presents two algorithms to construct minimum weight spanning trees with approximately minimum degree. The first method gives a spanning tree whose maximum degree is $O(\delta^* + logn)$ where δ^* is the minimum possible, and n is the number of vertices. The second method gives a spanning tree of degree no more than $k \cdot (\delta^* + 1)$, where k is the number of distinct weights in the graph. Finding the exact minimum is NP-hard.

*Department of Computer Science, Cornell University, Ithaca NY 14853. Research supported by ONR Graduate Fellowship.

1 Introduction

While it is easy enough to optimize the weight of a spanning tree, it is often more difficult to satisfy constraints which involve the degrees of the vertices. The problem of minimizing the maximum degree of a spanning tree is known to be NP-complete, as the Hamiltonian path problem is merely a special case of this problem. Other related NP-complete problems include finding a spanning tree with the maximum number of leaves [3], one that is isomorphic to a given tree [3], or one where the paths between all pairs of vertices produces the minimum congestion [1]. More NP-complete spanning tree problems can be found in [3] and [5]. Other related problems involve the simultaneous optimization of two parameters, such as the problem of finding the cheapest tree bounded by a given diameter [3].

Approximation algorithms are known for some of these problems. Goemans and Williamson developed a technique that applies to a variety of constrained forest problems, including the generalized Steiner tree problem, and the non-fixed point-to-point connection problem [4]. A result of Khuller, Raghavachari, and Young balances the cost and diameter for spanning trees [6].

Fürer and Raghavachari recently published an polynomial time approximation algorithm which constructs a spanning tree of degree no more than $\delta^* + 1$, where δ^* is the optimum degree [2]. In the same paper, they give a second algorithm for the same problem; this algorithm only produces a tree of degree $O(\delta^* + \log n)$ (where n is the number of vertices in the graph), but the techniques may be more generally applicable.

We consider a generalization of the problem addressed by Fürer and Raghavachari. Let $G = (V, E)$ be a graph with weights $w(e)$ on the edges $e \in E$. If there exists a minimum weight spanning tree of degree δ^*, the first algorithm presented here will construct a minimum weight spanning tree (MWST) of degree $O(\delta^* + \log n)$, extending the second algorithm of Fürer and Raghavachari.

Also presented here is an algorithm to construct a MWST of degree at most $k(\delta^* + 1)$ where k is the number of different weights in the graph. This method uses the first algorithm of Fürer and Raghavachari as a subroutine.

2 An Additive $\log n$ Approximation

Definition 2.1 *Define the* **rank** *of a tree T to be the the ordered n-tuple $(t_n, \ldots, t_1)$ where t_i is the number of vertices of degree i in T. Define a lexicographic order on these ranks; a tree S is of lower rank than tree T if $s_j < t_j$ for some j and $s_i = t_i$ for $i = j + 1, \ldots, n$.*

When an edge is added to a spanning tree, it creates a cycle. Removing any edge from the induced cycle, we are again left with a spanning tree. Define a *swap* to be any such exchange of edges. A swap is *cost-neutral* if the edges exchanged are of equal weight.

Consider a swap of the tree edge $(x, w) \in T$ for the edge $(u, v) \notin T$, with x distinct from u, v. Such a swap may increase the degree in T of both u and v by one, but it will decrease the degree of x. This will decrease the rank of T if the degree of x in T is at least two more than the maximum degree of u and v.

Definition 2.2 *A* **locally optimal** *minimum weight spanning tree is a MWST in which no cost-neutral swap decreases the rank of the tree.*

Theorem 2.3 *If T is a locally optimal MWST, and δ is the degree of T, then $\delta \leq b \cdot \delta^* + \lceil \log_b n \rceil$ for any constant $b > 1$.*

Proof:

Consider a locally optimal MWST, T. Let S_i denote the set of vertices of degree i in T, $U_i = \bigcup_{j=i}^{n} S_j$, and $\sigma_i = |U_i|$. By definition, U_δ is non-empty, so $\sigma_\delta \geq 1$. Since $\sigma_i \leq n$ for all i, the ratio σ_{i-1}/σ_i can not be greater than b for $\log_b n$ consecutive values of i.

Claim 2.4 *For any constant $b > 1$, there exists some x in the range $\delta - \lceil \log_b n \rceil \leq x \leq \delta$ such that $\sigma_{x-1}/\sigma_x \leq b$.*

Suppose we choose x to satisfy this property, and remove from T the edges adjacent to vertices of U_x. Let T_x denote the remaining edges of T. How many connected components (counting isolated vertices) are there in the subgraph T_x? Initially, T is entirely connected, and each edge removed creates a new connected component. There are at least x edges adjacent to each vertex of U_x and, since there are no cycles, we have at most $\sigma_x - 1$ edges which are adjacent to two vertices of U_x and counted twice. Therefore there must be at least $1 + x \cdot \sigma_x - (\sigma_x - 1)$, or $(x-1)\sigma_x + 2$ connected components in T_x.

Consider the graph $\hat{G}$ formed by contracting every component of T_x. Since any MWST of G must contain a MWST of $\hat{G}$, any MWST must include at least $(x-1)\sigma_x + 1$ edges from $\hat{G}$.

Consider an edge, $(v, w) \in E - T$, between two components of T_x. Let $P^T_{(v,w)}$ denote the path from v to w in T, and $P^{\hat{T}}_{(v,w)}$ denote those edges of $P^T_{(v,w)}$ which appear in $\hat{G}$, the edges on the path which are adjacent to a vertex in U_x. Suppose neither v nor w is in U_{x-1}. Since T is locally optimal, no cost-neutral swap can reduce the rank of T, so (v, w) must be more expensive than any edge in $P^{\hat{T}}_{(v,w)}$. Since (v, w) and $P^{\hat{T}}_{(v,w)}$ form a cycle in $\hat{G}$, this implies that (v, w) may not participate in a MWST of $\hat{G}$. Therefore only those edges which are adjacent to U_{x-1} may participate in a MWST of $\hat{G}$, and any MWST of G must contain at least $(x-1)\sigma_x + 1$ edges that are adjacent to U_{x-1}.

Earlier we chose x to satisfy the inequality $\sigma_{x-1}/b \leq \sigma_x$. Substituting, we see there must be at least $((x-1)\sigma_{x-1}/b) + 1$ edges adjacent to Σ_{x-1}. Therefore the average degree of a vertex in U_{x-1} must be at least $\frac{(x-1)\sigma_{x-1}+b}{b \cdot \sigma_{x-1}}$. Therefore $\delta^* > \frac{x-1}{b}$.

Combining this with the possible range for x, we find $\delta \leq b \cdot \delta^* + \lceil \log_b n \rceil$. ∎

Now constructing a locally optimal tree might take exponential time, but in the proof we only used the local optimality condition for the high degree vertices: those in U_k with $k = \delta - \lceil \log_b n \rceil$. Say that such a tree is **pseudo-optimal** for parameter b. We can compute a pseudo-optimal tree in polynomial time with the algorithm on the next page.

Algorithm 2.5 *How to construct a pseudo-optimal MWST.*

0 Start with any MWST, T. Let $b > 1$ be the desired approximation parameter. Let $l < n$ be the number of distinct edge weights, $w_1 \ldots w_l$, in T

1. Let d be the current maximum degree in T.

2. For every vertex, v, in G, check for appropriate improvements. Conduct a depth first traversal of T starting from v.

(a) Let w be the current vertex on the traversal of T, and P_w be the path in T between v and w.

(b) Assign variables $M_1 \ldots M_l$ such that M_i denotes the maximum degree of those vertices adjacent to edges of weight w_i in P_w. For a depth first traversal, this is easily maintained using stacks.

(c) If there is an edge $(v,w) \in E$, let w_i be its weight. If M_i is at least two greater than the degree of v and w, and M_i is at least $\delta - \lceil \log_b n \rceil$, then the edge (v,w) can be used to reduce the high-degree rank of T. Conduct the appropriate swap on T, and repeat to step (1) for the next iteration.

(d) If no appropriate cost-neutral swap was found involving v and w, continue the traversal.

3. If no appropriate cost-neutral swap was found in any of the traversals, terminate.

The tree produced, T, is clearly pseudo-optimal; any cost-neutral swap that can reduce the high-degree rank will be found. Since we are conducting $O(n)$ depth first traversals, and can maintain the M_i variables in constant time per step of the traversal, no more than $O(n^2)$ time will be spent on any iteration.

Lemma 2.6 *Algorithm 2.5 will terminate in $O(n^{2+1/\ln b})$ iterations.*

Proof: We use a potential function identical to that of Fürer and Raghavachari. Define the **potential** of a vertex v to be β^{d_v} for some fixed base $\beta > 2$ where d_v is the degree of v in the current tree. Define $\Phi = \sum_{v \in V} \beta^{d_v}$. Let $i = \delta - \lceil \log_b n \rceil$, the lower limit on the degree for our improvements.

Since we only perform swaps which improve the degree of some vertex in U_i, the reduction in Φ resulting from a swap is at least:

$$\beta^i + 2 \cdot \beta^{i-2} - 3 \cdot \beta^{i-1} = (\beta - 1) \cdot (\beta - 2) \cdot \beta^{i-2} \geq c \cdot \beta^i \text{ for some constant } c.$$

Now $\Phi \leq n \cdot \beta^\delta$, so each swap reduces Φ by at least a fraction of:

$$\frac{c \cdot \beta^i}{n \cdot \beta^\delta} = \frac{c \cdot \beta^{-\lceil \log_b n \rceil}}{n} \geq \frac{c'}{n^{1+\log_b \beta}} \text{ for some constant } c'.$$

Since this argument holds for any $\beta > 2$, we can choose $\beta = e$. Therefore in $O(n^{1+1/\ln b})$ iterations, the potential reduces by a constant fraction, and after $O(n^{2+1/\ln b})$ iterations, the algorithm must halt. ∎

This completes the proof of the main theorem:

Theorem 2.7 *The above algorithm computes, in $O(n^{4+1/\ln b})$ time, a minimum weight spanning tree of degree no more than $b \cdot \delta^* + \lceil \log_b n \rceil$, where δ^* is the minimum possible degree of any minimum weight spanning tree.*

3 A k-factor Approximation

A MWST can be described as a tree containing an edge of minimum cost in any cut. Those nodes that **can** be connected by edges of cost c or less, **must** be connected by such edges. This property is applied to build a MWST of low degree.

Notation 3.1 *Let k be the number of distinct weights used in a MWST. Let $w_1 \ldots w_k$ denote the different weights in increasing order, and E_i be the set of edges of weight w_i or less. Let the w_i-degree of a node, v, in a tree denote the number of edges adjacent to v in the tree which are of weight w_i.*

In phase i, we will use the edges of weight w_i to join the components already spanned by edges of lesser weight. Using an algorithm for the following claim as a subroutine, we approximately minimize the maximum number of weight w_i edges adjacent to any given node.

Claim 3.2 *Given a forest, F, in a graph $G = (V, E)$, we can extend F in polynomial time to a forest F' such that $F \subseteq F'$, F' is a spanning forest of G, and the number of edges in $F' - F$ adjacent to any given node of V is at most one more than the minimum possible.*

This result follows immediately from the techniques used to prove the second algorithm of Fürer and Ragavachari [2].

In the first phase, we merely run the subroutine to find an approximately minimum degree spanning forest, F_1, for the graph $G_1 = (V, E_1)$. In phase i, for $i = 2 \ldots k$, we run the subroutine on the graph $G_i = (V, E_i)$ with initial forest F_{i-1}. Note that this guarantees that any subgraph of G that can be spanned by edges in E_i *will* be spanned by edges of weight E_i in the forest generated. Since we know that the edges E_k span G, the tree produced is a MWST.

Theorem 3.3 *The MWST produced by the above algorithm has maximum degree $\delta \leq k \cdot (\delta^* + 1)$.*

Proof: Suppose there there is a MWST $\hat{T}$ of degree δ^*, yet the tree T produced by our algorithm has some vertex of degree greater than $k \cdot (\delta^* + 1)$. By the pigeonhole principle, there is some i such that the w_i-degree of that vertex is at least $\delta^* + 2$.

Now, any vertices connected by edges in E_{i-1} are also connected in F_{i-1}, since F_{i-1} spans G_{i-1}. And any MWST must span G_i with edges in E_i. Since our subroutine uses $\delta^* + 2$ edges of weight w_i adjacent to a single vertex, any MWST must have some vertex with w_i-degree at least $\delta^* + 1$, implying that no MWST of degree δ^* is possible. By contradiction, the degree of T is no more than $k \cdot (\delta^* + 1)$. ∎

4 Acknowledgements

I would like to thank Éva Tardos for her advice and direction.

References

[1] P.M. Camerini, G. Galbiati, and F. Maffioli. Complexity of spanning tree problems: Part i. *European J. Op. Res.*, 5:346–352, 1980.

[2] M. Furer and B. Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 317–324, 1992.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[4] M.X. Goemans and D.P. Williamson. A general approximation technique for constrained forest problems. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 307–316, 1992.

[5] D.S. Johnson. The *np*-completeness column: An ongoing guide. *J. Algorithms*, 6:145–159, 1985.

[6] S. Khuller, B. Rachavachari, and N. Young. Balancing minimum spanning and shortest path trees. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 243–250, 1993.