The minimum degree spanning tree problem on directed acyclic graphs

Guohui Yao*
School of Computer Sci and Tech
Shandong University
Jingshi Road 73, 250061 Ji'nan
P R China

Daming Zhu
School of Computer Sci and Tech
Shandong University
Jingshi Road 73, 250061 Ji'nan
P R China

Shaohan Ma
School of Computer Sci and Tech
Shandong University
Jingshi Road 73, 250061 Ji'nan
P R China

Abstract: The minimum degree spanning tree problem has been studied extensively. In this paper, we present a polynomial time algorithm for the minimum degree spanning tree problem on directed acyclic graphs. The algorithm starts with an arbitrary spanning tree, and iteratively reduces the number of vertices of maximum degree. We can prove the algorithm must reduce a vertex of the maximum degree for each phase, and finally result in an optimal tree. The algorithm terminates in $O(mn\log n)$ time, where *m* and *n* are the number of edges and vertices respectively.

Key–Words: algorithm, spanning tree, directed acyclic graph, polynomial time, minimum degree

1 Introduction

The minimum degree spanning tree (MDST) problem is that of constructing a spanning tree of an undirected graph $G = (V, E)$, whose maximal degree is the smallest among all spanning trees of G. The problem is a generalization of the Hamilton Path problem, thus can easily be shown to be NP-hard. The directed version of the problem has a root vertex $r \in V$ as part of the input, asks to construct an incoming (or outgoing) spanning tree rooted at r for which the maximal indegree (outdegree for an outgoing spanning tree) of a vertex is minimized. We call this problem as DMDST problem in this paper. In the Steiner version of the problems, along with the input graph, we are also given a distinguished set of vertices $D \subseteq V$; the problem is to find a tree of minimum degree which spans at least the set D. The two former problems are the special cases of the minimum degree Steiner tree problem when $D = V$.

Computing low degree trees is a fundamental problem, and has many applications. Both these problems and their variants have been studied extensively [1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13]. A survey on minimum degree problems has appeared in a book on approximation algorithms [9].

Fürer and Raghavachari [5] provided a polynomial time algorithm to approximate the MDST problem to within one of optimal. In other words, their algorithm finds a spanning tree whose degree is at most $\Delta^* + 1$ in polynomial time. Clearly no better approximation algorithms are possible for this problem unless P=NP. Their algorithm also extends to the Steiner version of the problem, but only works on undirected graphs. The currently best known approximation algorithm for the DMDST problem is due to Raghavachari [10]. The algorithm computes a tree whose maximal degree is at most $O(\Delta^* + \log n)$, where Δ^* is the degree of some optimal tree for the problem, and n is the number of vertices. Fraigniaud [2] showed that the Steiner version of the DMDST problem cannot be approximated within $(1 - \epsilon)\ln|D|$ for any $\epsilon > 0$ unless $NP \subset DTIME(n^{\log\log n})$, as opposed to the undirected version.

In this paper, we consider the problem of computing minimum degree spanning trees of directed acyclic graphs (DAG_DMDST). In this special case, given as input are a directed acyclic graph and a root vertex. In contrast with the DMDST problem, the DAG_DMDST problem can be solved in polynomial time. We present an exact algorithm for this problem. The running time of the algorithm is shown to be $O(mn\log n)$. The algorithm does not employ any exhaustive enumeration, and is likely to be much faster in practice.

2 Definitions and notation

The input is an arbitrary directed acyclic graph $G = (V, E)$, and a root vertex $r \in V$. Let n be the number

*Supported by national natural science foundation of China (60573024).

of vertices of G. We always assume r to be reachable from all vertices of G.

Definition 1 *A rooted spanning tree T of G is a subgraph of G with the following properties*:
(1) T contains all the vertices of G but does not contain any cycles.
(2) The outdegree of every vertex except r is exactly one.
(3) There is a path in T from every vertex to r.

In other words, the subgraph of G with $n-1$ edges in which there is a directed path from every vertex to r is a spanning tree rooted at r. This is known as an incoming spanning tree. One can also define the notion of an outgoing spanning tree, in which r can reach every vertex of G through directed paths. In this paper, the spanning tree always refers to an incoming spanning tree.

Let T be a spanning tree rooted at r. For each edge (v, w) in T, we call w the parent of v, denoted by $p(v)$. Each vertex except r has a unique parent, and r has none. Vertex v is an *ancestor* of u if there is a directed path from u to v. We call u a descendent of v if v is an ancestor of u. We define the *subtree* of T rooted at v to be the subtree spanned by v and all the descendents of v in T, denoted by T_v. Its vertices set and edges set denoted by $V(T_v)$ and $E(T_v)$ respectively.

The number of the edges entering into vertex v is the degree of v, denoted by $d(v)$. For a rooted spanning tree, let S_Δ be the set of vertices whose degree is Δ, i.e. $S_\Delta = \{v | d(v) = \Delta\}$. The degree of a rooted spanning tree T, $d(T)$, is the *maximum degree of any of its vertices*, $d(T) = max\{d(v) | v \in V(T)\}$. In this paper, the degree of a tree is always denoted by k. The goal is to find a rooted spanning tree of minimum degree.

3 The exact algorithm for the DAG_DMDST problem

Witness set is first proposed by Fürer and Raghavachari [4] for the MDST problem. It is a subset of vertices which can be used to bound the degree of the spanning tree. Given $G = (V, E)$ to be an undirected simple graph. Let W be a subset of V with size $|W| = w$. If removing W from G splits G into t connected components, then the degree of any spanning tree of G is not less than $\lceil \frac{w+t-1}{w} \rceil$.

Similarly, we can bound the degree of the optimal rooted spanning trees of a DMDST instance.

Lemma 2 *Let $G = (V, E)$ be a directed graph and $r \in V$. Suppose there are subsets of vertices $W \subset V$ and $X \subset V$ that satisfy the following properties:*
(a) $r \notin X$,
(b) For any $x \in X$, if there exists edge $e = (x, w) \in E$, then $w \in W$.
Then the degree of a DMDST tree rooted at r of G is not less than $\left\lceil \frac{|X|}{|W|} \right\rceil$.

Proof: Let T^* be an optimal spanning tree rooted at r for the DMDST problem. Since it is a rooted spanning tree, it contains a path from any vertex to the root. By the conditions of the lemma, a path from a vertex $x \in X$ to r contains one incident edge into a vertex in W. Therefore we have identified $|X|$ incoming edges to vertices in W. Therefore the average degree of vertices in W is at least $\frac{|X|}{|W|}$, implying that there is at least one vertex in T^* whose degree is $\left\lceil \frac{|X|}{|W|} \right\rceil$ or more. □

Lemma 3 *Let T_v be a subtree rooted at v of a directed acyclic graph G. There is no directed path from v to any vertex in the set of vertices of T_v.*

Proof: Suppose there is a path p_1 from v to some vertex w in the set of vertices of T_v. Since w is the descendent of v in T_v, there is a path p_2 from w to v. $p_1 \cup p_2$ is a directed cycle of G, thus violating the acyclic property of G. □

Lemma 4 *Let T be a rooted spanning tree of degree k of a directed acyclic graph $G = (V, E)$. Let Δ^* be the degree of a directed minimum degree spanning tree. Let S be the set of vertices of degree k in T. Let B be an arbitrary subset of vertices of degree k-1 in T. Let X be the set of all the children of vertices in $S \cup B$. If there are no edges from X to $V - (S \cup B)$ of G, then $k = \Delta^*$.*

Proof: Let witness set W be $S \cup B$. By the theorem hypothesis, for any $x \in X$, if edge $e = (x, w) \in E$, then $w \in W$. Hence these sets X and W satisfy the conditions given in the statement of Lemma 2. Each vertex in S has k children and each vertex in B has $k-1$ children, therefore the size of X is $k|S| + (k-1)|B|$. Apply Lemma 2,

$$\Delta^* \geq \left\lceil \frac{|X|}{|W|} \right\rceil = \left\lceil \frac{k|S| + (k-1)|B|}{|S| + |B|} \right\rceil = k.$$

Therefore $k = \Delta^*$. □

The algorithm starts with an arbitrary spanning tree T rooted at r and works in phases. Let k be the degree of T. and S be the set of vertices of degree k. In each phase we try to reduce the size of S by 1. If successful, we move on to the next phase. We will show later that if it is not possible to reduce the size

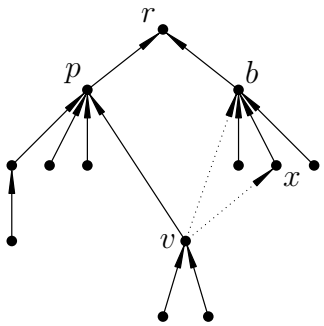
of S, then there is a set B of degree $k-1$ vertices in T such that, the tree T along with the sets S and B satisfies the conditions of Lemma 4 and hence T is an optimal tree.

The algorithm tries to reduce the degree of vertices with the maximum degree iteratively using the following improvement scheme. Consider a vertex p with the maximum degree k. Let v be one child of p. we can decrease the degree of p by 1 if we can delete the edge (v,p) and find an alternate path for v to reach the root r. We call this as improvement step applied to p. By Lemma 3, the new path from v to r does not go through any vertex of T_v. Let $x \notin V(T_v)$ be the nearest vertex to v in the new path from v to r. We replace the edge (v,p) by the edge (v,x). Since x can reach r in T and v can reach x after improvement, v can also reach r. Vertices of T_v can reach v, they can also reach r. Thus the subtree T_v of p *transferred* to x. It can be verified that the above operation results in another spanning tree since the number of edges is still $n-1$ and all vertices can still reach r. We will perform such an improvement step only if, after the improvement, the degree of the vertex whose degree increased, is less than k. Note that the degree of p decreased by 1 and x increased by 1, the degree of the other vertices do not change.

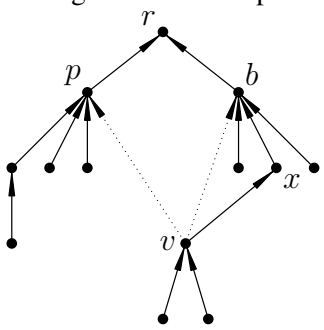
Definition 5 *Let $(v,x) \notin E(T)$ be an edge in G, the parent of v is p, $x \notin V(T_v)$. If $d(x) \geq k-1$, we say that x blocks p from (v,x). If x does not block p, then (v,x) can be used to reduce the degree of p through an improvement step. In such a case, we say p benefits from (v,x). If p benefits from (v,x), we say x is the obliging vertex of p.*

Fig.1 illustrates an example of an improvement step. In this example, the tree edges are shown by thick lines and other edges of G are shown by dotted lines. The degree of p is 4, and we see if v can find an alternate path to r so that the edge (v,p) may be deleted from T, thus decreasing the degree of p to 3. The edge (v,b) is not chose because the degree of b is already 3, and if we chose to add this edge, its degree becomes 4. Decreasing the degree of p by increasing the degree of b to 4 (the maximum degree) does not make progress. In other words, b blocks p from (v,b). The algorithm uses (v,x) to modify the spanning tree, and x is the obliging vertex of p. the new spanning tree is shown in Figure 1b. The degree of p has been reduced to 3.

The following is a top-down view of our algorithm. Let X be the set of all children of vertices in S_k, where S_k is the set of all vertices with maximal degree in T. If there are no edges from X to $V-S_k$, we are done. Lemma 4 can be applied here and hence



a) Spanning tree before improvement



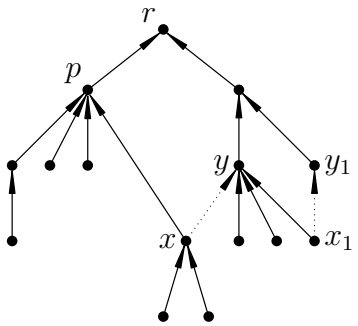
b) Spanning tree after improvement

Fig.1: an improvement applied to vertex p

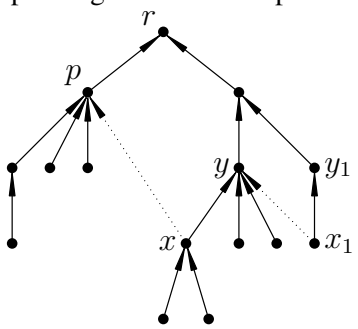
T is an optimal spanning tree. Otherwise any edge (x,y) from X to $V-S_k$ benefit $p(x)$ and can be used to reduce the degree of $p(x)$. If such an edge (x,y) is not blocked by y, we make this improvement reducing the number of degree k vertices by 1 and continue. Otherwise y blocks $p(x)$ from (x,y). Suppose the degree of y can be reduced by 1 through improvement steps. Then y is made non-blocking and in this case, we say that a sequence of improvements propagate to p.

Fig.2 illustrates an example of a sequence of improvements which propagate to p. In this example, the tree edges are shown by thick lines and other edges of G are shown by dotted lines. As shown in Figure 2a, y blocks p from (x,y), and we see if y can be made non-blocking by reducing its degree by 1 through an improvement step. There exists such an improvement – replacing (x_1,y) by (x_1,y_1) and y_1 is non-blocking. The algorithm use the two edges (x,y) and (x_1,y_1) to modify the spanning tree; the new spanning tree is shown in Figure 2b. The degree of p has been reduced by 1.

The algorithm is implemented in a bottom-up fashion as follows. At the beginning of each phase of the algorithm, all vertices in $S_k \cup S_{k-1}$ are marked as bad. All other vertices are marked as good. Let X be the set of children of vertices in $S_k \cup S_{k-1}$ and Y be $V-(S_k \cup S_{k-1})$, the set of good vertices. Note that X may contain vertices in $S_k \cup S_{k-1}$. If there are no edges from X to Y of G, the algorithm stops. In



a) Spanning tree before improvement



b) Spanning tree after improvement

Fig.2: improvements which propagate to p

this case, Lemma 4 shows that the set of bad vertices remaining is witness to the fact that $k = \Delta^*$. Otherwise let (x, y) be an edge from X to Y. We consider the parent of x. If $p = p(x)$ is a vertex of degree k, we have identified a set of improvements which propagate to p. Making these changes reduces the size of S_k by 1. Otherwise if p is a vertex of $k-1$, we mark p as good, add it to Y and remove all children of p from X. The good vertex w is the obliging vertex of p. We then go back to look for other edges from X to Y. In all cases, we either find a way to reduce the degree of some vertex in S_k or find a witness set allowing us to apply Lemma 4. The following algorithm implements the above ideas and output an optimal spanning tree.

Procedure PropagateImprovement(p)

1 If the degree of p is less than k, return;
2 Otherwise
   a) Reduce the degree of p by 1 by transferring one of subtrees of p to its obliging vertex x.
   b) PropagateImprovement(x)

End Procedure

Algorithm DAG_DMDST(G, r):

1 Find an arbitrary spanning tree T of G rooted at r. Let k be its degree.
2 Mark vertices of degree k and $k-1$ as bad and all other vertices as good. Let X be the set of children of bad vertices and Y be the set of good vertices.
3 **While** there is an edge (x, y) from X to Y of G do
   a) Let $p = p(x)$ be the parent of x in T and y be the obliging vertex of p.
   b) If the degree of p is $k-1$, mark p as good, add it to Y and remove all children of p from X.
   c) Otherwise the degree of p is k
      i) Run PropagateImprovement(p). Update k if necessary;
      ii) Go back to step 2.

   End While
4 Output the final tree T and its degree k.

End

The algorithm proceeds in iterations. In each iteration, one vertex are designated good. We therefore speak of a vertex being marked good at iteration i. A vertex whose degree in T is less than $k-1$ is said to be good at iteration 0. Let F_i be the vertices of T marked good at iteration i or less. Note that $F_i \subset F_{i+1}$ for all i.

Lemma 6 *Any vertex u marked good can be made non-blocking by a sequence of improvements which propagate to u.*

Proof: The proof proceeds by induction on iteration i. If u was marked good at iteration 0, it has degree less than $k-1$ in T, and is therefore non-blocking by definition. Otherwise if u was marked good at iteration $i > 0$, it has an obliging vertex $x \in F_{i-1}$. Since x was marked good at an iteration $j \leq i-1$, by the inductive hypothesis, x can be made non-blocking by applying improvements within $F_j \subset F_i$. We can reduce the degree of u by transferring one of its subtrees to x, and make it non-blocking. This is done within F_i. Hence our algorithm maintains only vertices which can be made non-blocking. Note that any update needed to make a vertex marked good at iteration i non-blocking is within F_i. □

Lemma 7 *When the algorithm stops, $k = \Delta^*$.*

Proof: Let S be S_k and B be the bad vertices of degree $k-1$. Let X be the set of all the children of vertices in $S \cup B$. Note that the algorithm stops only when there are no edges from X to $V-(S \cup B)$. Hence the tree T along with these sets S, B and X satisfy the conditions of Lemma 4 and we get the desired result. □

4 Analysis of the running time of the algorithm

As mentioned above, the algorithm works in phases. Let S be the set of vertices of degree k. In each phase

we try to reduce the size of S by 1. As the size of S reduces by 1 in each phase except the last one, there are at most $O(n/k)$ phases when the maximal degree is k. The sum of the harmonic series corresponding to reducing values of k is,

$$\sum_{k=n-1}^{2} \left\lfloor \frac{n}{k} \right\rfloor = O(n\log n).$$

Hence there are at most $O(n\log n)$ phases. Each phase can be implemented in $O(m)$ time. Thus we have the following theorem.

Theorem 8 *There is an $O(mn\log n)$-time exact algorithm for the minimum degree spanning tree problem on directed acyclic graphs, where m is the number of edges and n is the number of vertices.*

5 Conclusion

We have presented an exact algorithm for the directed minimum degree spanning tree problem on directed acyclic graphs. We introduced a new notion of witness sets that works in directed graphs. Designing algorithms for the Steiner version of DAG_DMDST problem is also one of our research goals. Fig.3 illustrates an example that shows our algorithm can't resolve the Steiner version of DAG_DMDST problem yet. In this example, there is a vertex p of degree k. Its children are $c_1, \ldots, c_k$. Each of these k children have an edge into vertex s, which is not in the current Steiner tree. And s has an edge into p. It can be verified that there is no improvement possible for vertex p. But, the degree of p can be reduced to $\lfloor k/2 \rfloor + 1$ by connecting $\lfloor k/2 \rfloor$ of p's children through s. In fact if the graph has a number of other extra nodes similar to s, the degree of p can be reduced even to 2. This example shows that our algorithm does not guarantee any performance bound on the degree of the tree for the Steiner case. The reason that we were unable to apply Theorem 4 is that, the children of bad vertices can be connected to an extra vertex which is not in the current Steiner tree.

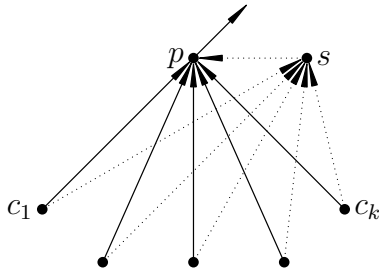


Fig.3: A Steiner Example

Finally, are there algorithms that work in weighted, directed acyclic graphs?

References:

[1] A. Agrawal, P. Klein and R. Ravi, How tough is the minimum-degree Steiner tree? A new approximate min-max equality, Brown University, Technical Report: TR CS–91–49, 1991.

[2] P. Fraigniaud, Approximation algorithms for minimum-time broadcast under the vertex-disjoint paths mode, In *9th Annual European Symposium on Algorithms (ESA '01), volume 2161 of LNCS*, Springer, 2001, pp. 440–451.

[3] M. Fürer and B. Raghavachari, An NC approximation algorithm for the minimum degree spanning tree problem, In: *Proc. of the 28th Annual Allerton Conf. on Communication, Control and Computing*, Monticello, IL: Allerton House, 1990, pp. 274–281.

[4] M. Fürer and B. Raghavachari, Approximating the minimum degree spanning tree to within one from the optimal degree, In: *Proc. of 3rd ACM-SIAM Symp. On Disc. Algorithms (SODA)*, New York: ACM Press, 1992, pp. 317–324.

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

[6] B. Gavish, Topological design of centralized computer networks - formulations and algorithms, *Networks*, 1982, 12: 355–377.

[7] J. Könemann and R. Ravi, A matter of degree: improved approximation algorithms for degree-bounded minimum spanning trees, In: *Proc. of 32nd annual ACM Symposium on Theory of Computing(STOC)*, New York: ACM Press, 2000, pp. 537–546.

[8] E. L. Lawler, Combinatorial optimization: networks and matroids, *Holt, Rinehart and Winston*. New York, 1976.

[9] B. Raghavachari, Algorithms for finding low degree structures, In: *Approximation algorithms*, Boston: PWS Publisher Inc., 1996, pp. 266–295.

[10] B. Raghavachari, The directed minimum-degree spanning tree problem, In: *Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science*. London: Springer –verlag, 2001, pp. 232–243.

[11] R. Ravi, Rapid rumor ramification: approximating the minimum broadcast time, In: P*roc of 35th Annual IEEE Symposium on Foundations of Computer Science(FOCS)*, New York: IEEE Computer Society Press, 1994, pp. 202–213.

[12] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. III, Many birds with one stone: multi-objective approximation algrorithms, In: *Proc. of 25th Annual ACM*

Symp. On the Theory of Computing(STOC), New York: ACM Press, 1993, 438–447.

[13] R. Ravi, B. Raghavachari and P. Klein, Approximation through local optimality: designing networks with small degree, In: *Proc. of 12th Conf. on Foundations of Software Tech. and Theoret. Comp. Sci. (FSTTCS)*, 1992, pp. 279–290.