JOURNAL OF ALGORITHMS **29,** 132–141 (1998)
ARTICLE NO. AL980944

Approximating Maximum Leaf Spanning Trees in Almost Linear Time

Hsueh-I Lu*

Department of CSIE, National Chung-Cheng University, Tainan, Taiwan

and

R. Ravi†

Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

Received September 10, 1996; revised April 22, 1998

Given an undirected graph, finding a spanning tree of the graph with the maximum number of leaves is MAX SNP-complete. In this paper we give a new greedy 3-approximation algorithm for maximum leaf spanning trees. The running time $O((m + n)\alpha(m, n))$ required by our algorithm, where m is the number of edges and n is the number of nodes, is almost linear in the size of the graph. We also demonstrate that our analysis of the performance of the greedy algorithm is tight via an example.

1. INTRODUCTION

Given a connected undirected graph $G = (V, E)$, the maximum leaf spanning tree problem is to find a spanning tree of G with the maximum number of leaves. This problem finds applications in communication networks and circuit layouts [4, 14]. The maximum leaf spanning tree problem is NP-complete [7] and MAX SNP-complete [6].

Previous Work

The maximum leaf spanning tree problem has been extensively studied [3, 5, 8–12]. Most of the previous work has focused on finding spanning

* E-mail: hil@cs.ccu.edu.tw.
† E-mail: ravi@cmu.edu. Research supported in part by NSF CAREER grant CCR-9625297.

0196-6774/98 $25.00

trees with many leaves in graphs with minimum degree at least d for some $d \geq 3$. For such graphs, good lower bounds on the number of leaves achievable in a spanning tree are derived in [8, 9, 11, 12]. There has also been work on polynomial-time solutions to the problem of determining if a given graph has a spanning tree with at least k leaves for fixed k. The first such algorithm was due to Fellows and Langston [5]. The running time of their algorithm was improved by Bodlaender [3].

In [10], we presented a series of approximation algorithms for the problem based on the technique of local optimization—the algorithm based on k changes swaps k tree edges for k non-tree edges if this resulted in a spanning tree with a higher number of leaves. The approximation ratios of the first two algorithms in the series based on one and two changes were shown to be 5 and 3, respectively. Let m be the number of edges and let n be the number of nodes. The kth algorithm in the series uses k changes to achieve local optimality. The time complexity, $O(m^k n^{k+2})$, is intolerably high even if k is small.

Results

In this paper we give a new greedy approximation algorithm for the maximum leaf spanning tree problem with the best currently achievable approximation ratio, 3. The running time required by our algorithm is almost linear in the size of the given graph. We also show via an example that the analysis of our greedy algorithm is tight.

Lower Bound

In our earlier work, motivated by the improving performance ratio of the series of algorithms we presented [10], we raised the question: ''Does the series of algorithms based on k-changes form a *polynomial-time approximation scheme* (PTAS)?'' However, Galbiati, Maffioli, and Morzenti [6] showed that the maximum leaf spanning tree problem is MAX SNP-complete. Therefore there exists some constant $\epsilon > 0$ such that there is no $(1 + \epsilon)$ approximation for maximum leaf spanning tree unless P = NP [1, 2].

2. PRELIMINARIES

Let G be a connected undirected graph. We use $V(G)$ to denote the set of nodes in G. We refer to an edge uw of G as *the edge incident to u and w*. For a subset of nodes $S \subset V$, let $\Gamma(S)$ denote the set of edges with exactly one endpoint in S. We sometimes overload notation and use $\Gamma(H)$ to denote $\Gamma(V(H))$ for a subgraph H of G. The *degree of v in G* is the

number of edges of G incident to v. Let $V_i(G)$ denote the set of nodes that have degree i in G. Let $\bar{V}_i(G)$ denote the set of nodes that have degree at least i in G. Clearly, $\bar{V}_0(G) = V(G)$. The *leaves of* G are the nodes in $V_1(G)$.

A subtree of G is *nonsingleton* if it has more than one node. Let T be a subtree of G. One can verify that

$$|\bar{V}_3(T)| \leq |V_1(T)| - 2. \tag{1}$$

Let $\text{path}_T(u, w)$ be the path in T that connects u and w. The following lemma is also straightforward.

LEMMA 1. *Let T be a subtree of G.*

1. *Let u, v, and w be three distinct nodes of G. If v is in $\text{path}_T(u, w)$, then v is not a leaf of T.*

2. *Let u, v_1, v_2, and w be four nodes of G. If u is not in $\text{path}_T(v_1, w)$ and u is not in $\text{path}_T(v_2, w)$, then u is not in $\text{path}_T(v_1, v_2)$.*

3. LEAFY SUBTREES AND LEAFY FORESTS

Let T be a subtree of G.

DEFINITION 1. We say T is *leafy* if $\bar{V}_3(T)$ is not empty and every node in $V_2(T)$ is adjacent in T to exactly two nodes in $\bar{V}_3(T)$. A forest F of G is leafy if F is composed of disjoint leafy subtrees of G. We say F is *maximally leafy* if F is not a subgraph of any other leafy forest of G.

Lower Bound

The following lemma ensures that at least one-third of the nodes in a leafy subtree T are leaves of T.

LEMMA 2. *Let T be a leafy subtree of G. Then $|V(T)| \leq 3|V_1(T)| - 5$.*

Proof. Since T is leafy, each node in $V_2(T)$ must be adjacent in T to exactly two nodes in $\bar{V}_3(T)$. Consider the induced subtree on $\bar{V}_2(T)$, where each node of degree 2 is replaced by an edge. The number of edges in this subtree is one less than the number of nodes. Therefore $|V_2(T)| \leq |\bar{V}_3(T)| - 1$. It follows from (1) that

$$\begin{aligned} |V(T)| &= |V_1(T)| + |V_2(T)| + |\bar{V}_3(T)| \\ &\leq |V_1(T)| + 2|\bar{V}_3(T)| - 1 \\ &\leq 3|V_1(T)| - 5. \end{aligned}$$

∎

Properties of Maximally Leafy Forests

Let F be a maximally leafy forest of G. Let $T_1, \ldots, T_k$ be the disjoint leafy subtrees of F. One can verify that F has the following properties. We use the example in Fig. 1 to illustrate each property. The dark lines in the figure are the edges in the maximally leafy forest F, which is composed of three leafy subtrees T_1, T_2, and T_3.

1. Let w be a node in $\overline{V}_2(T_i)$. Then w cannot be adjacent in G to any node not in T_i. (Nodes x_1 and x_2 are two examples of w; e.g., suppose x_1 were adjacent to a node such as x_5. Then F would not be maximal since the edge (x_1, x_5) could be added to F.)

2. Let w be a node in T_i. Let w_1 and w_2 be two distinct nodes adjacent to w in G. If w_1 is not in F, then w_2 must be in T_i. (Nodes x_3 is an example of w. If x_3 had two neighbors not in F, both these edges could be added to F, contradicting its maximality.)

3. Let w be a node in F. If w is adjacent to two distinct nodes not in F, then the degree of w in G is 2. (Nodes x_5 and x_6 are two examples of w. Note that such nodes are not in F. If the degree of say x_5 were greater than 2, then x_5 and its three neighbors not in F could be added as an additional star in F, contradicting its maximality again.)

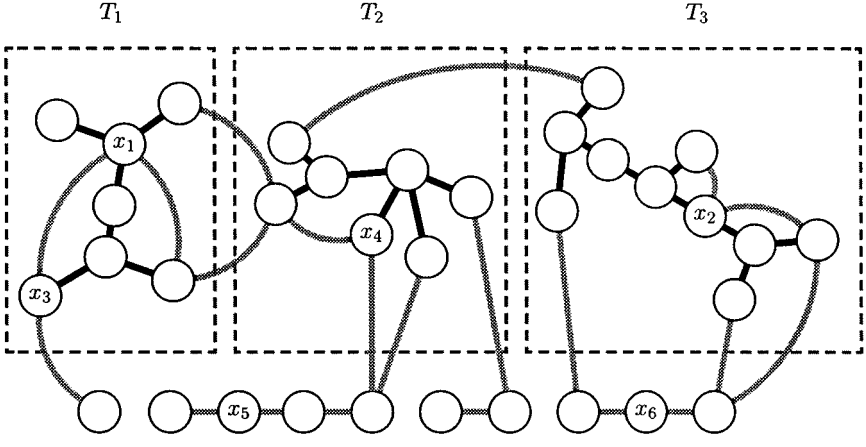

FIG. 1. An example of a maximally leafy forest represented by dark edges. Gray edges are the remaining edges in G.

Upper Bound

The crux of the proof of the performance guarantee is an upper bound we derive on the maximum number of leaves in *any* tree relative to the number of leaves in any maximally leafy forest of the graph.

THEOREM 1. *Let F be a maximally leafy forest of G. Let T be a spanning tree of G such that F is a subgraph of T. Then*

$$|V_1(T)| \geq |V_1(\hat{T})|/3$$

for any spanning tree $\hat{T}$ of G.

The following two lemmas are essential to proving the theorem.

LEMMA 3. *Let F be a maximally leafy forest of G that is composed of k disjoint leafy subtrees $T_1, \ldots, T_k$ of G. Then*

$$|V_1(\hat{T})| \leq |V(F)| - k + 1$$

for any spanning tree $\hat{T}$ of G.

Proof. The outline of the proof is as follows. First, we identify a representative node of degree 3 or higher in each of the trees in F, consider the paths in $\hat{T}$ between them, and show that roughly k distinct leaves in F occur as internal nodes in these paths. We do this by picking one of these representatives, say v_k in T_k, as a ''root'' node and examining the paths from other representative nodes toward this root in $\hat{T}$. Next, we consider nodes that are leaves in $\hat{T}$ but not leaves in F and show that the path from each such node to v_k in $\hat{T}$ contains a distinct leaf of F as an internal node. This relates the number of leaves of F and $\hat{T}$ as desired.

We now commence the formal proof. Let v_i be a node in $\overline{V}_3(T_i)$ for every $i = 1, \ldots, k$. For every $i = 1, \ldots, k-1$, let u_i be the node in T_i that is farthest from v_i in $\text{path}_{\hat{T}}(v_i, v_k)$. Let $\hat{v}_1, \ldots, \hat{v}_l$ be the distinct nodes in $V_1(\hat{T}) \backslash V(F)$. Namely, each $\hat{v}_j$ is a leaf of $\hat{T}$ not in F. For every $j = 1, \ldots, l$, let $\hat{u}_j$ be the node in F that is closest to $\hat{v}_j$ in $\text{path}_{\hat{T}}(\hat{v}_j, v_k)$. We show $u_1, \ldots, u_{k-1}, \hat{u}_1, \ldots, \hat{u}_l$ are $l + k - 1$ distinct nodes in $V(F) \backslash V_1(\hat{T})$, which implies the lemma.

By definition of u_i we know the first edge from u_i in $\text{path}_{\hat{T}}(u_i, v_k)$ is in $\Gamma(T_i)$. (Recall that each edge in $\Gamma(T_i)$ is incident to a node in T_i and a node not in T_i.) By property 1 we know $u_i \in V_1(T_i)$. Therefore $u_i \neq v_i$ and $u_i \neq v_k$. It follows from Lemma 1 that $u_i \notin V_1(\hat{T})$, since $u_i \in \text{path}_{\hat{T}}(v_i, v_k)$. Hence $u_i \in V(F) \backslash V_1(\hat{T})$. Since $T_1, \ldots, T_k$ are disjoint and $u_i \in T_i$ for every $i = 1, \ldots, k-1$, we know $u_1, \ldots, u_{k-1}$ are $k-1$ distinct nodes in $V(F) \backslash V_1(\hat{T})$.

Let j be one of $1, \ldots, l$. By definition of $\hat{v}_j$ we know that the first edge from u_i in $\text{path}_{\hat{T}}(\hat{u}_j, \hat{v}_j)$ is in $\Gamma(T_{j^*})$ for some T_{j^*} that contains $\hat{u}_j$. By property 1 we know $\hat{u}_j \in V_1(T_{j^*})$. Therefore $\hat{u}_j \neq \hat{v}_j$ and $\hat{u}_j \neq v_k$. It follows from Lemma 1 that $\hat{u}_j \notin V_1(\hat{T})$, since $\hat{u}_j \in \text{path}_{\hat{T}}(\hat{u}_j, v_k)$. Hence $\hat{u}_j \in V(F) \setminus V_1(\hat{T})$. We show that $\hat{u}_1, \ldots, \hat{u}_l$ are distinct and $\hat{u}_j \notin \{u_1, \ldots, u_{k-1}\}$.

Assume for a contradiction that $\hat{u}_j = \hat{u}_{j'}$ for some $j' \neq j$. Let $P = \text{path}_{\hat{T}}(\hat{u}_j, \hat{v}_j)$ and $P' = \text{path}_{\hat{T}}(\hat{u}_{j'}, \hat{v}_{j'})$. Let w be the node in $V(P) \cap V(P')$ that is closest to $\hat{v}_j$ in P. Since $\hat{v}_j \notin P'$, there exists a node w_1 in $\text{path}_{\hat{T}}(w, \hat{v}_j)$ such that ww_1 is an edge of P. Since $\hat{v}_{j'} \notin P$, there exists a node w_2 in $\text{path}_{\hat{T}}(w, \hat{v}_{j'})$ such that ww_2 is an edge of P'. Clearly $w_1 \neq w_2$ and $w_1, w_2 \notin F$. It follows from property 2 that $w \neq \hat{u}_j$, implying that $w \notin F$. Therefore there exists an edge ww_3 in P where $w_3 \neq w_1$ and $w_3 \neq w_2$. This contradicts the fact that F is maximally leafy by property 3.

Assume for a contradiction that $\hat{u}_j = u_i$ for some $i \in \{1, \ldots, k-1\}$. Let $P = \text{path}_{\hat{T}}(\hat{u}_j, \hat{v}_j)$ and $P' = \text{path}_{\hat{T}}(u_i, v_k)$. Let w be the node in $V(P) \cap V(P')$ that is closest to $\hat{v}_j$ in P. Since $\hat{v}_j \notin P'$, there exists a node w_1 in $\text{path}_{\hat{T}}(w, \hat{v}_j)$ such that ww_1 is an edge of P. Since $v_k \notin P$, there exists a node w_2 in $\text{path}_{\hat{T}}(w, v_k)$ such that ww_2 is an edge of P'. Clearly $w_1 \neq w_2$ and $w_1 \notin F$. By definition of u_i we know $w_2 \notin T_i$. It follows from property 2 that $w \notin T_i$. Hence $u_i \notin \text{path}_{\hat{T}}(w, v_k)$ by definition of u_i and $\hat{u}_j \notin \text{path}_{\hat{T}}(\hat{v}_j, w)$ by definition of $\hat{u}_j$. Since $\hat{u}_j = u_i$ by assumption, it follows from Lemma 1 that $\hat{u}_j \notin \text{path}_{\hat{T}}(\hat{v}_j, v_k)$. This contradicts the definition of $\hat{u}_j$. ∎

LEMMA 4. *Let F be a forest of G that has k disjoint nonsingleton subtrees. Let T be a spanning tree of G such that F is a subgraph of T. Then* $|V_1(T)| \geq |V_1(F)| - 2(k-1)$.

Proof. The intuition behind this proof is that the trees in F can be connected into a single spanning tree by iteratively adding a single edge that merges two disconnected trees and in the process destroys (increases the degree of) at most two leaves in these trees. More formally, let F' be the forest of G obtained from F by adding an edge e in $T \setminus F$ to F. Let k' be the number of disjoint nonsingleton subtrees of F'. We show

$$|V_1(F')| - 2(k'-1) \geq |V_1(F)| - 2(k-1). \tag{2}$$

The lemma thus follows inductively, since the number of disjoint nonsingleton subtrees in T is 1.

- If e is not incident to any nonsingleton subtree of F, then adding e forms a new subtree on two singletons. Hence $k' = k + 1$ and $|V_1(F')| = |V_1(F)| + 2$. Therefore (2) holds.

• If e is incident to exactly one nonsingleton subtree of F, then $k' = k$ and $|V_1(F')| \geq |V_1(F)|$. Thus (2) holds.

• If e is incident to two nonsingleton subtrees of F, then $k' = k - 1$. Since we may have connected two leaves of F to form F', we know $|V_1(F')| \geq |V_1(F)| - 2$. Therefore (2) holds. ∎

We are now ready to prove the theorem as follows.

Proof for Theorem 1. Suppose F has k disjoint leafy subtrees $T_1, \ldots, T_k$. By Lemma 2 we know

$$\begin{aligned} |V(F)| &= \sum_{i=1}^{k} |V(T_i)| \\ &\leq 3|V_1(F)| - 5k. \end{aligned}$$

It follows from Lemma 3, the above inequality, and Lemma 4 that

$$\begin{aligned} |V_1(\hat{T})| &\leq |V(F)| - k + 1 \\ &\leq 3|V_1(F)| - 6k + 1 \\ &\leq 3(|V_1(T)| + 2(k-1)) - 6k + 1 \\ &\leq 3|V_1(T)|. \end{aligned}$$

∎

4. THE ALGORITHM

We give an approximation algorithm for maximum leaf spanning tree in this section. Given a graph G, our algorithm computes a spanning tree T for G by the following two steps.

1. Obtain a maximally leafy forest F for G.
2. Add edges to F to make it a spanning tree T for G.

It follows from Theorem 1 that the approximation ratio of the above algorithm is 3, which is the same as that of the algorithm in [10]. We show that our algorithm can be implemented to run in time $O((m + n)\alpha(m, n))$.

We implement the first step of our algorithm in Fig. 2. We use $S(w)$ to denote the subtree of F that contains node w. The degree of node w in F is kept in $d(w)$. The variable d' is the maximal number of edges adjacent to v that could be added to F without creating cycles. If uv is one of those d' edges, then $S(u)$ is stores in the set S'. If $d' + d(v)$ is greater than or

```
MAXIMALLYLEAFYFOREST(G)
1   Let F be an empty set.
2   For every node v in G do
3       S(v) := {v}.
4       d(v) := 0.
5   For every node v in G do
6       S' := ∅.
7       d' := 0.
8       For every node u that is adjacent to v in G do
9           If u ∉ S(v) and S(u) ∉ S' then
10              d' := d' + 1.
11              Insert S(u) into S'.
12      If d(v) + d' ≥ 3 then
13          For every S(u) in S' do
14              Add edge uv to F.
15              Union S(v) and S(u).
16              Update d(u) := d(u) + 1 and d(v) := d(v) + 1.
17  Output F as a maximally leafy forest of G.
```

FIG. 2. The procedure for finding a maximally leafy forest F of G.

equal to 3, then we add those d edges to F and union $S(v)$ with those d subtrees $S(u)$.

Using the union-find data structure in [13] for set operations, the first step runs in time $O((m + n)\alpha(m, n))$. In particular, the first condition in step 9 can be implemented using a find operation on the structure $S(v)$. For the second condition, we first observe that for any set $S(u)$, we may use the representative label in the set structure (root of the union-find structure) to denote this set. Also, we observe that for a given node v, if S' contains three or more distinct sets, we proceed to merge these sets into one subsequently (since the condition in step 12 is satisfied). Hence, it suffices to check in the loop in step 8 that for a given node v, the set S' contains at least three distinct sets $S(u)$; subsequently we can combine the check for such candidate edges (v, u) with merging the corresponding sets $S(u)$ and $S(v)$ if the edge passes the test (does not form a cycle). Thus we can assume that the set S' never has more than two elements when we wish to test $S(u) \notin S'$. Consequently, the number of set operations performed is on the order of the degree of v when processing v, giving a total time bound of $O(m\alpha(m, n))$ for this step.

The second step of extending F to a spanning tree can be done by shrinking each leafy subtree in F into a single node and then finding a spanning tree for the corresponding shrunken graph. This can be done in time $O(m + n)$. It follows that the time complexity of our algorithm is $O((m + n)\alpha(m, n))$, which is almost linear in the size of the graph.

4.1. *Tightness of the Performance Ratio*

The example in Fig. 3 shows that the analysis for the performance ratio of our algorithm is tight. The spanning tree T in dark lines is leafy, since every node in $V_2(T)$ is adjacent in T to exactly two nodes in $\overline{V}_3(T)$. Let the gray lines be the edges of G that are not in T. The maximum leaf spanning tree of G is composed of all the edges incident to x in G. The approximate ratio of T for the example is $9/5$. Generalizing the example in a natural way yields examples in which the performance ratio of the algorithm asymptotically tends to 3.

5. REMARKS

As suggested by the example shown in Fig. 3, the approximation ratio of our algorithm might be improved by growing the maximally leafy forest by the descending order of the degree of nodes in G. To be more specific, we first sort the nodes by their degrees in G. This takes time $O(m + n)$, since the degree of every node is no more than n. We then try the for loop in step 5 from high-degree nodes to low-degree nodes. It would be interesting to determine the performance ratio of this modified approximation algorithm.

ACKNOWLEDGMENTS

We thank an anonymous referee for careful comments that improved the presentation considerably.

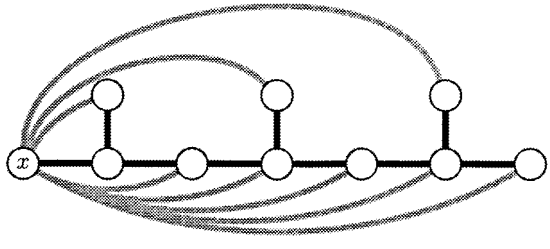

FIG. 3. A maximally leafy spanning tree that has approximation ratio 3.

REFERENCES

1. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and the hardness of approximation problems, *in* ''Proceedings of the Thirty-third Annual IEEE Symposium on Foundations of Computer Science,'' 1992, pp. 14–23.
2. S. Arora and S. Safra, Probabilistic checking of proofs: A new characterization of NP, *in* ''Proceedings of the Thirty-Third Annual IEEE Symposium on Foundations of Computer Science,'' 1992, pp. 2–13.
3. H. L. Bodlaender, On linear time minor tests and depth first search, *in* ''Proceedings of the Workshop on Algorithms and Data Structures,'' 1989, pp. 577–590.
4. E. W. Dijkstra, Self-stabilizing systems in spite of distributed control, *Comm. ACM* **17** (1974), 643–644.
5. M. R. Fellows and M. A. Langston, ''On Well-Partial-Order Theory and Its Applications to Combinatorial Problems of VLSI Design,'' Technical Report CS-88-188, Dept. of Computer Science, Washington State University, 1988.
6. G. Galbiati, F. Maffioli, and A. Morzenti, A short note on the approximability of the maximum leaves spanning tree problem, *Inform. Process. Lett.* **52** (1994), 45–49.
7. M. R. Garey and D. S. Johnson, ''Computers and Intractability: A Guide to the Theory of NP-Completeness,'' Freeman, San Francisco, 1979.
8. J. R. Griggs, D. J. Kleitman, and A. Shastri, Spanning trees with many leaves in cubic graphs, *J. Graph Theory* **13** (1989), 669–695.
9. D. J. Kleitman and D. B. West, Spanning trees with many leaves, *SIAM J. Discrete Math.* **4** (1991), 99–106.
10. H.-I. Lu and R. Ravi, The power of local optimization: Approximation algorithms for maximum-leaf spanning tree, *in* ''Proceedings of the Thirtieth Annual Allerton Conference on Communication, Control and Computing,'' Oct. 1992, pp. 533–542.
11. C. Payan, M. Tchuente, and N. H. Xuong, Arbres avec un nombres maximum de sommets pendants, *Discrete Math.* **49** (1984), 267–273.
12. J. A. Storer, Constructing full spanning trees for cubic graphs, *Inform. Process. Lett.* **13** (1981), 8–11.
13. R. E. Tarjan, ''Data Structures and Network Algorithms,'' SIAM, Philadelphia, PA, 1983.
14. M. Tchuente, Sur l'auto-stabilisation dans un réseau d'odinateurs, *RAIRO Inform. Théor. Appl.* **15** (1981), 47–66.