A Near-linear-time Approximation Algorithms for Maximum-leaf Spanning Tree

Hsueh-I Lu and R. Ravi

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-96-06
January 1996

A Near-linear-time Approximation Algorithm for Maximum-leaf Spanning Tree

Hsueh-I Lu
hil@cs.brown.edu
Department of Computer Science
Brown University

R. Ravi
ravi+@andrew.cmu.edu
Graduate School of Industrial Administration
Carnegie Mellon University

January 1996

Abstract

Given an undirected graph G, finding a spanning tree of G with maximum number of leaves is not only NP-complete [11] but also MAX SNP-complete [10]. The approximation ratio of the previously best known approximation algorithm for maximum leaf spanning tree is three [19]. However, the high-order running time required by the previous algorithm makes it impractical. In this paper we give a new factor-three approximation algorithm for the same problem. The running time required by our algorithm is almost linear in the size of G. This improves the previous algorithm by a factor of $\tilde{\Omega}(mn^3)$, where n is the number of nodes and m is the number of edges.

Keywords: Approximation algorithms, NP-complete problems, Spanning trees, Performance ratio, Communication network design, Combinatorial algorithms.

1 Introduction

The NP-complete Maximum Leaf Spanning Tree problem has been extensively studied [12, 17, 23, 25, 7, 3, 19]. It has applications in circuit layouts and communication networks [5, 27]. In [19] a series of approximation algorithms are given. The approximation ratios of the first two algorithms in the series are shown to be five and three. The observation leads to an open question: "is the series of algorithms a *polynomial-time approximation scheme* (PTAS) [14]?" The open question is answered negatively in [10], which shows that Maximum Leaf Spanning Tree is MAX SNP-complete [22]. 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].

The series of approximation algorithms given in [19] is based on the technique of local optimization [4, 16, 18, 21, 28, 8, 24, 9, 13, 6]. The k^{th} algorithm in the series uses k-changes to achieve local optimality. The time complexity, $O(m^k n^{k+1})$, is intolerably high even if k is small. In this paper we give an efficient approximation algorithm for Maximum Leaf Spanning Tree with the best currently achievable approximation ratio for the problem.

2 Preliminaries

Let G be an 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*. Let G' be a subgraph of G. A node v is *in G'* if $v \in V(G')$.

An edge uw is *incident to G'* if exactly one of u and w is in $V(G')$. 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 $\bar{V}_0(G') = V(G)$.

Let T be a subtree of G. One can easily verify that

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

The following lemma is also straightforward.

Lemma 1 Let T be a subtree of G. 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.

3 Leafy Subtrees and Leafy Forests

Let T be a subtree of G. We say T is *leafy* if $\bar{V}_3(T)$ is not empty, and every node in $V_2(T)$ is adjacent 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. In this section we show the following theorem.

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 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 $\bar{V}_3(T)$. Therefore $|V_2(T)| \leq |\bar{V}_3(T)| - 1$. It follows from (1) that

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

□

Let F be a maximally leafy forest of G. Let $T_1, \ldots, T_k$ be the disjoint leafy subtrees of G in F. One can easily verify that F has the following properties.

The properties for maximally leafy forests

1. Let w be a node in $\bar{V}_2(T_i)$. Then w cannot be adjacent in G to any node not in T_i.
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.
3. Let w be a node not in F. If w is adjacent to two distinct nodes not in F, then the degree of w in G is two.

4. Let w be a node not in F. Let w_1, w_2, and w_3 be three distinct nodes adjacent to w in G. If $w_1 \notin F$, then w_2 and w_3 must be in the same leafy subtree T_i for some $i \in \{1, \ldots, k\}$.

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 Let v_i be a node in $\bar{V}_3(T_i)$ 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 $\mathrm{path}_{\hat{T}}(v_i, v_k)$. Let $\hat{v}_1, \ldots, \hat{v}_\ell$ be the distinct nodes in $V_1(\hat{T}) \setminus V(F)$. Namely each $\hat{v}_j$ is a leaf of $\hat{T}$ not in F. For every $j = 1, \ldots, \ell$, let $\hat{u}_j$ be the node in F that is closest to $\hat{v}_j$ in $\mathrm{path}_{\hat{T}}(\hat{v}_j, v_k)$. We show $u_1, \ldots, u_{k-1}, \hat{u}_1, \ldots, \hat{u}_\ell$ are $\ell + k - 1$ distinct nodes in $V(F) \setminus V_1(\hat{T})$, which implies the lemma.

By definition of u_i we know the first edge from u_i in $\mathrm{path}_{\hat{T}}(u_i, v_k)$ is in $\Gamma(T_i)$. 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 $u_i \in \mathrm{path}_{\hat{T}}(v_i, v_k)$. Hence $u_i \in V(F) \setminus V_1(\hat{T})$. 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 $V(F) \setminus V_1(\hat{T})$.

Let j be one of $1, \ldots, \ell$. By definition of $\hat{v}_j$ we know that the first edge from $\hat{u}_j$ in $\mathrm{path}_{\hat{T}}(\hat{u}_j, \hat{v}_j)$ is in $\Gamma(T_{j^*})$ for some T_{j^*} that contains $\hat{u}_j$. By Property 1 we know $\hat{u} \in V_1(T_{j^*})$. 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 $\hat{u}_j \in \mathrm{path}_{\hat{T}}(\hat{u}_j, v_k)$. Hence $\hat{u}_j \in V(F) \setminus V(\hat{T})$. We show that $\hat{u}_1, \ldots, \hat{u}_\ell$ 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 $P = \mathrm{path}_{\hat{T}}(\hat{u}_j, \hat{v}_j)$ and $P' = \mathrm{path}_{\hat{T}}(\hat{u}_{j'}, \hat{v}_{j'})$. 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 $\mathrm{path}_{\hat{T}}(w, \hat{v}_j)$ such that ww_1 is an edge of P. Since $\hat{v}_{j'} \notin P$, there exists a node w_2 in $\mathrm{path}_{\hat{T}}(w, \hat{v}_{j'})$ 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 $P = \mathrm{path}_{\hat{T}}(\hat{u}_j, \hat{v}_j)$ and $P' = \mathrm{path}_{\hat{T}}(u_i, v_k)$. 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 $\mathrm{path}_{\hat{T}}(w, \hat{v}_j)$ such that ww_1 is an edge of P. Since $v_k \notin P$, there exists a node w_2 in $\mathrm{path}_{\hat{T}}(w, v_k)$ 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 $w \notin T_i$. Therefore there exists an edge ww_3 in P where $w_3 \neq w_1$ and $w_3 \neq w_2$. By Property 3 we know $w_3 \in F$, implying $w_3 \in T_i$. This contradicts the fact that F is maximally leafy by Property 4, since $w_1 \notin F$, $w_2 \notin T_i$, and $w_3 \in T_i$. $\square$

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 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 one.

- If e is not incident to any nonsingleton subtree of F, then $k' = k+1$ and $|V_1(F')| = |V_1(F)|+2$. Thus (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 $k' = k-1$ and $|V_1(F')| \geq |V_1(F)| - 2$. Thus (2) holds.

$\square$

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

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

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 three, which the same as the algorithm in [19]. 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 as follows.

```
For every node v in G do
    Create a set S(v) that contains only v.
    Let d(v) be zero.
For every node v in G do
    Let S' be an empty set.
    Let d' be zero.
    For every node u that is adjacent to v in G do
        If u ∉ S(v) and S(u) ∉ S' then
            Increase d' by one.
            Insert S(u) into S'.
```

If $d(v) + d' \geq 3$ then
  For every $S(u)$ in S' do
    Add edge uv to F.
    Union $S(v)$ and $S(u)$.
  Increase $d(v)$ by d'.

One can easily verify that the resulting F is a maximally leafy forest for G. Using the data structure in [26], the first step runs in time $O((m+n)\alpha(m,n))$.

The second step can be done by shrinking each leafy subtree in F into a single node, and then finding a spanning tree for the corresponding shrinked graph. This can certainly 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.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, "Proof verification and the hardness of approximation problems," *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," *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," *Proceedings of Workshop on Algorithms and Data Structures*, (1989), pp. 577–590.

[4] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research* 6, (Nov.-Dec. 1958), pp. 791–812.

[5] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Communications of ACM* 17, (Nov. 1974), pp. 643–644.

[6] T. Dimitriou, and R. Impagliazzo, "Towards an analysis of local optimization algorithms," *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing*, (1996).

[7] 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.

[8] M. Fürer and B. Raghavachari, "Approximating the minimum degree spanning tree to within one from the optimal degree," *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, (1992), pp. 317–324.

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

[10] G. Galbiati, F. Maffioli and A. Morzenti, "A short note on the approximability of the maximum leaves spanning tree problem," *Information Processing Letters* 52, (1994), pp. 45–49.

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

[12] J. R. Griggs, D. J. Kleitman, and A. Shastri, "Spanning trees with many leaves in cubic graphs," *Journal of Graph Theory* 13, (1989), pp. 669–695.

[13] M. M. Halldórsson "Approximating discrete collections via local improvements," *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, (1995), pp. 160–169.

[14] D. S. Johnson, "Approximation algorithms for combinatorial problems," *Journal of Computer and System Sciences* 37, (1988), pp. 79–100.

[15] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, "How easy is local search?" *Journal of Computer and System Sciences* 37, (1988), pp. 79–100.

[16] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *BSTJ* 49, (1970), pp. 291–308.

[17] D. J. Kleitman and D. B. West, "Spanning trees with many leaves," *SIAM Journal on Discrete Mathematics* 4, (Feb. 1991), pp. 99–106.

[18] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the Traveling-Salesman Problem," *Operations Research* 21, (1973), pp. 498–516.

[19] H.-I. Lu and R. Ravi, "The power of local optimization: approximation algorithms for maximum-leaf spanning tree," *Proceedings of the Thirtieth Annual Allerton Conference on Communication, Control and Computing*, (Oct. 1992), pp. 533–542.

[20] C. H. Papadimitriou, A. A. Schäffer and M. Yannakakis, "On the complexity of local search (Extended Abstract)," *Proceedings of the Twenty-second Annual ACM Symposium on the Theory of Computing*, (1990), pp. 438–445.

[21] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc. (1982).

[22] C. H. Papadimitriou and M. Yannakakis, "Optimization, Approximation, and Complexity classes," *Proceedings of the Twentieth Annual ACM Symposium on the Theory of Computing*, (1988) pp. 229–234.

[23] C. Payan, M. Tchuente, and N. H. Xuong, "Arbres avec un nombres maximum de sommets pendants," *Discrete Mathematics* 49, (1984), pp. 267–273.

[24] R. Ravi, B. Raghavachari, and P. N. Klein, "Approximation through local optimality: Designing networks with small degree," *Proceedings, Twelfth Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, (Dec. 1992), LNCS 652, pp. 279–290.

[25] J. A. Storer, "Constructing full spanning trees for cubic graphs," *Information Processing Letters* 13, (1981), pp. 8–11.

[26] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM, Philadelphia, Pennsylvania, (1983).

[27] M. Tchuente, "Sur l'auto-stabilisation dans un réseau d'odinateurs," *R. A. I. R. O. Informatique Théorique* 15, (1981), pp. 47–66.

[28] M. Yannakakis, "The analysis of local search problems and their heuristics," *STACS*, (1990), pp. 289–311 .