On Maximum Leaf Trees and Connections to Connected Maximum Cut Problems

Rajiv Gandhi
Rutgers University, Camden, NJ
rajivg@camden.rutgers,edu *

Mohammad Taghi Hajiaghayi
University of Maryland, College Park, MD
hajiagha@cs.umd.edu †

Guy Kortsarz
Rutgers University, Camden, NJ
guyk@camden.rutgers.edu ‡

Manish Purohit
Google, Mountain View, CA
mpurohit@google.com

Kanthi Sarpatwar
IBM Research, Yorktown Heights, NY
sarpatwa@us.ibm.com

Abstract

In an instance of the (directed) Max Leaf Tree (MLT) problem we are given a vertex-weighted (di)graph $G(V, E, w)$ and the goal is to compute a subtree with maximum weight on the leaves. The weighted Connected Max Cut (CMC) problem takes in an undirected edge-weighted graph $G(V, E, w)$ and seeks a subset $S \subseteq V$ such that the induced graph $G[S]$ is connected and $\sum_{e \in \delta(S)} w(e)$ is maximized.

We obtain a constant approximation algorithm for MLT when the weights are chosen from $\{0, 1\}$, which in turn implies a $\Omega(1/\log n)$ approximation for the general case. We show that the MLT and CMC problems are related and use the algorithm for MLT to improve the factor for CMC from $\Omega(1/\log^2 n)$ (Hajiaghayi et al., ESA 2015) to $\Omega(1/\log n)$.

1 Introduction

Given a vertex-weighted graph $G = (V, E, w)$, the Max Leaf Tree problem is to find a subtree T such that the total weight of the leaves of T is maximized. The closely related *Maximum Leaf Spanning Tree* problem seeks to find a *spanning* tree T of G that maximizes the sum of weights on the leaves of T. The Maximum Leaf Spanning Tree problem on unweighted graphs has been very well studied and constant factor approximation algorithms are known both for undirected [5, 8, 10, 11, 13] as well as directed graphs [3, 4].

In the unweighted case, it is easy to observe that any subtree T can be augmented to a spanning tree T' without decreasing the number of leaves. Consequently, on unweighted graphs –

*Supported in Part by NSF Grant number 1218620

†Supported in part by NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHS/AFOSR grant FA9550-12-1-0423, and another DARPA SIMPLEX grant.

‡Supported in Part by NSF Grant Number 1218620 and NSF Grant Number 1540547

both undirected and directed (in this case, we assume that all the nodes have a directed path from the root), the Max Leaf Tree problem is equivalent to the Maximum Leaf Spanning Tree problem. However, the two problems differ significantly in their weighted versions. In this paper, we study the approximability of the weighted Max Leaf Tree problem on both directed and undirected graphs.

Hajiaghayi et al. [6] initiated the study of the Connected Submodular Maximization problem: Given a graph $G = (V, E)$ and a non-negative submodular set function $f : 2^V \to \mathbb{R}^+ \cup \{0\}$, find a subset S of vertices that maximizes $f(S)$ such that the induced graph $G[S]$ is connected. It can be readily observed that the weighted Max Leaf Tree problem is a special case of the Connected Submodular Maximization problem where the submodular function is $f(S) = \sum_{u \in N(S)} w(u)$ where $N(S)$ refers to the set of all vertices that are not in S but have neighbors in S. Another important special case of the Connected Submodular Maximization problem is the Connected Maximum Cut (CMC) problem: Given an edge-weighted undirected graph $G = (V, E, w)$ find a set $S \subseteq V$ that maximizes the total weight of edges in the cut $\delta(S, V \setminus S)$ such that the induced graph $G[S]$ is connected. Hajiaghayi et al. [6] obtained the first $\Omega(\frac{1}{\log^2 n})$ approximation algorithm for the Connected Maximum Cut problem on weighted graphs.

Contribution and Techniques

Our key results can be summarized as follows.

1. We obtain the first $\Omega(\frac{1}{\log n})$ approximation algorithm for the Max Leaf Tree problem with general weights on directed and undirected graphs.
2. We show that an α-approximation algorithm for the weighted Max Leaf Tree problem leads to an $\Omega(\alpha)$-approximation algorithm for the Connected Maximum Cut problem on general weighted graphs. Combined with the previous result, we obtain an $\Omega(\frac{1}{\log n})$-approximation, thus improving upon the $\Omega(\frac{1}{\log^2 n})$-approximation obtained by Hajiaghayi et al. [6].

1.1 Related Work

Max Leaf Spanning Tree is a classical problem that has been very well-studied from the perspective of approximation algorithms [5, 8, 10, 11, 13] as well as fixed parameterized tractability [1, 3]. As observed by Drescher and Vetta [4], the weighted Max Leaf Spanning Tree problem is as hard as the Independent Set problem and hence cannot be approximated within a factor of $\Omega(\frac{1}{n^{1-\epsilon}})$ unless P=NP. For unweighted directed graphs, Drescher and Vetta [4] obtained an $\Omega(\frac{1}{\sqrt{OPT}})$ approximation for the Max Leaf Spanning Tree problem. Daligault and Thomassé [3] discovered the first constant approximation algorithm for the same.

Hajiaghayi et al. [6] introduced the Connected Maximum Cut problem and gave an $\Omega(\frac{1}{\log n})$ approximation algorithm on unweighted graphs as well as an $\Omega(\frac{1}{\log^2 n})$ approximation algorithm on general weighted graphs. Lee, Nagarajan and Shen [9] studied a generalization of the connected maximum cut where connectivity and cut are defined by different graphs. Optimization of submodular functions over graphs subject to a connectivity constraint has been well studied in different contexts especially for monotone submodular functions [2, 7].

2 Preliminaries

Problem Definition. (Directed) Weighted Maximum Leaf Tree : Given a graph $G = (V, E)$ and a

weight function $w : V \to \mathbb{R}^+$, find a subtree T that maximizes $\sum_{v \in L(T)} w(v)$ where $L(T)$ denotes the set of leaves of the tree T. In the directed version, we are also given a root vertex r and the goal is to find an out-tree T rooted at r that maximizes $\sum_{v \in L(T)} w(v)$.

Problem Definition. Connected Max Cut : Given a graph $G = (V, E)$ and a weight function $w : E \to \mathbb{R}^+$, find a subset $S \subset V$ that maximizes $\sum_{e \in \delta(S, V \setminus S)} w(e)$ such that the subgraph induced by S is connected.

3 Approximation Algorithms for General Graphs

In this section, we consider the Weighted Maximum Leaf Tree on directed graphs and provide the first $\Omega(\frac{1}{\log n})$ approximation algorithm. Naturally all our results are applicable for undirected graphs as well. Our approach is to first reduce the problem to an instance of the *unweighted* maximum leaf tree problem in two stages and then use a constant factor approximation algorithm for the same.

Stage 1: Reduction to binary weighted max leaf tree.

We partition the vertices in V into $\log_2 n$ classes based on their weight as follows. Let OPT denote the total weight of leaves of an optimal weighted Max Leaf Tree and let $w_{max} = \max_{v \in V} w(v)$, $w_{min} = \min_{v \in V | w(v) \neq 0} w(v)$ denote the weights of the heaviest and lightest (non-zero) vertices in V respectively. Without loss of generality, we can assume that $OPT \geq w_{max}$ and hence we reset the weights of edges whose weights are less than $\frac{w_{max}}{2n}$ to 0. This alteration modifies OPT by at most an $1/2$ fraction and we can now assume that $w_{min} > \frac{w_{max}}{2n}$. We define thresholds $\tau_0 = \frac{w_{max}}{2n}$ and $\tau_i = \tau_0 2^i$ for $i \in [\log_2 2n]$. The vertices in V are grouped into $O(\log n)$ classes using these thresholds as follows - let $V_i = \{v \in V | \tau_{i-1} < w(v) \leq \tau_i\}$.

Let O denote an optimal solution for the Max Leaf Tree problem and let $OPT_i = \sum_{v \in V_i \cap L(O)} w(v)$ denote the contribution of vertices from V_i to the optimal solution. Let $i^* = \mathbf{arg\,max}_{i \in [\log_2 2n]} OPT_i$ denote the vertex class that has the highest contribution. We now set $w'(v) = 0$ for all $v \notin V_{i^*}$ and set $w'(v) = 1$ for all $v \in V_{i^*}$ to obtain a new instance $\mathcal{I}'$ of the Max Leaf Tree problem with only $\{0, 1\}$ weights.

Lemma 3.1. *If there exists an α-approximation algorithm for the Max Leaf Tree problem with $\{0, 1\}$ weights, then there exists an $\Omega(\frac{\alpha}{\log n})$-approximation algorithm for the Max Leaf Tree problem with general weight functions.*

Proof. Given an instance $\mathcal{I} =< G, w >$ of the general weighted Max Leaf Tree problem, we obtain an instance $\mathcal{I}' =< G, w' >$ of the Max Leaf Tree problem with $\{0, 1\}$ weights using the procedure described above. By construction, we have that $OPT(\mathcal{I}') \geq \frac{OPT_{i^*}}{\tau_{i^*}} \geq \frac{OPT}{\tau_{i^*} \log_2 2n}$.

Let T be an α-approximate solution for the instance $\mathcal{I}'$. We now observe that the tree T is an $\Omega(\frac{\alpha}{\log n})$-approximate solution for the original instance $\mathcal{I}$ as well.

$$w(L(T)) \geq \tau_{i^*-1} w'(L(T)) \geq \tau_{i^*-1} \alpha OPT(\mathcal{I}') \geq \frac{\tau_{i^*-1} \alpha OPT}{\tau_{i^*} \log_2 2n} \geq \frac{\alpha OPT}{2 \log_2 2n}$$

□

Stage 2: Reduction to unweighted max leaf tree.

Lemma 3.2. *Given a digraph $G = (V, E)$ with $\{0, 1\}$-vertex weights, we can construct an unweighted digraph $G' = (V', E')$ in polynomial time, such that G' has a Max Leaf Tree solution T' of weight at least ψ if and only if G has a Max Leaf Tree solution T of weight at least ψ.*

Proof. Let $v \in V \setminus \{r\}$ be a non-root vertex in G with weight 0 and let $\{v_1, v_2, \ldots, v_l\}$ denote the set of its neighbors. Consider the digraph G' obtained from G by deleting v along with all its incident edges and adding a directed edge between every ordered pair of its neighbors $\{v_i, v_j\}$ that does not already exist, if the path $v_i \to v \to v_j$ exists in G. Let T denote a tree of leaf weight ψ in G. If $v \notin T$, then clearly $T' = T$ is the required solution in G'. If $v \in T$, then let u be the parent of v in T and $C(v)$ be its children in T. Now the tree T' can be obtained by deleting v, along with all its edges, and instead adding the edges between u and each $w \in C(v)$. Clearly, all these edges exist in G'.

Finally, to prove the other direction, suppose we have a feasible solution T' of leaf weight ψ in G'. Now, if T' is a subtree of G, then $T = T'$ is a feasible solution in G of weight ψ. Otherwise, T' contains an edge $(u, w) \notin E$ such that $(u, v) \in E$ and $(v, w) \in E$. We can now obtain the subtree T from T' by adding the vertex v and replacing such edges (u, w) by (u, v) and (v, w). The proof of the lemma then follows from induction.

□

Theorem 3.3. *There exists a polynomial time $\Omega(\frac{1}{\log n})$ approximation algorithm for the weighted Max Leaf Tree problem in general graphs.*

Proof. Lemma 3.2 shows that one can reduce the binary weighted Max Leaf Tree problem to the unweighted Max Leaf Tree problem without a loss in the approximation factor. However, in the unweighted case, one can always assume without loss of generality that the Max Leaf Tree is also spanning. Consequently, well known constant factor approximation algorithms for the maximum leaf spanning tree problem (see [13] for undirected graphs and [3] for directed graphs) now yield a constant factor approximation algorithm for the maximum leaf tree with $\{0, 1\}$ weights. The theorem now follows from Lemma 3.1. □

4 Connections to Connected Maximum Cut

We now show that the weighted Max Leaf Tree problem can be used to obtain a simple $\Omega(\frac{1}{\log n})$-approximation algorithm for the weighted connected maximum cut problem.

Theorem 4.1. *Given a polynomial time α-approximation algorithm for the weighted Max Leaf Tree problem, we can obtain an $\frac{\alpha}{4}$-approximation algorithm for the weighted Connected Max Cut problem.*

Let $G = (V, E, w)$ denote an instance $\mathcal{I}$ of the weighted connected max cut problem. To obtain an instance $\mathcal{I}'$ of the Max Leaf Tree problem, we define a weight function $w' : V \to \mathbb{R}^+$ as follows - $w'(v) = \sum_{e=(u,v)} w(e)$. We first claim that optimal solution value of the Max Leaf Tree instance so constructed is at least as large as the optimal solution of the weighted connected max cut instance.

Claim 4.2. $OPT(\mathcal{I}') \geq OPT(\mathcal{I})$

Proof. Let S denote the optimal solution for the weighted connected max cut of graph G. Let $\mathcal{N}(S)$ denote the set of all vertices that have neighbors in S but are not in S themselves, i.e. $\mathcal{N}(S) = \{u \in V \setminus S \mid \exists v \in S \text{ such that } (u,v) \in E\}$.

By definition, we thus have $\delta(S, V \setminus S) = \delta(S, \mathcal{N}(S))$ and $\sum_{e \in \delta(S,\mathcal{N}(S))} w(e) = OPT(\mathcal{I})$. Since S is a feasible solution for the connected max cut, $G[S]$ is connected and thus there exists a subtree $\tilde{T}$ of G with $L(\tilde{T}) = \mathcal{N}(S)$. Since $\tilde{T}$ is a feasible solution for the maximum leaf tree problem, we have the following.

$$OPT(\mathcal{I}') \geq w'(L(\tilde{T})) = \sum_{u \in \mathcal{N}(S)} w'(u) = \sum_{u \in \mathcal{N}(S)} \sum_{e=(u,v)} w(e) \geq \sum_{e \in \delta(S,\mathcal{N}(S))} w(e) \geq OPT(\mathcal{I})$$

□

Let T denote a α-approximate solution for the weighted Max Leaf Tree problem obtained via the given approximation algorithm and let L denote the set of leaves of T. Let w_L denote the weight of edges in $G[L]$, the subgraph induced by L in the graph G. We now partition L into two disjoint sets L_1 and L_2 such that the weight of edges in $w(\delta(L_1, L_2)) \geq \frac{w_L}{2}$. This can be done by applying the standard algorithm for Max-Cut (e.g. see [12]) on $G[L]$. Now, consider the two connected subgraphs $T \setminus L_1$ and $T \setminus L_2$. We first claim that every edge in $\delta(L)$ belongs to either $\delta(T \setminus L_1)$ or $\delta(T \setminus L_2)$. Indeed, any edge e in $\delta(L)$, belongs to one of the four possible sets, namely $\delta(L_2, T \setminus L)$, $\delta(L_1, V \setminus T)$, $\delta(L_1, T \setminus L)$ and $\delta(L_2, V \setminus T)$. In the first two cases, e belongs to $\delta(T \setminus L_2)$ while in the last two cases, e belongs $\delta(T \setminus L_1)$, hence the claim. Further, every edge in $\delta(L_1, L_2)$ belongs to both $\delta(T \setminus L_1)$ and $\delta(T \setminus L_2)$. We have -

$$\begin{aligned} w(\delta(T \setminus L_1)) + w(\delta(T \setminus L_2)) &= w(\delta(L)) + 2w(\delta(L_1, L_2)) \geq w(\delta(L)) + w_L \\ &\geq \frac{\sum_{u \in L} w'(u)}{2} = \frac{w'(L)}{2} \geq \frac{\alpha OPT(\mathcal{I})}{2} \end{aligned}$$

Hence, the better of the two solutions $T \setminus L_1$ or $T \setminus L_2$ is guaranteed to have a cut of weight at least $\frac{\alpha OPT(\mathcal{I})}{4}$. □

References

[1] Paul S Bonsma, Tobias Brueggemann, and Gerhard J Woeginger. A faster fpt algorithm for finding spanning trees with many leaves. In *MFCS*, pages 259–268. 2003.

[2] Gruia Calinescu and Alexander Zelikovsky. The polymatroid Steiner problems. *Journal of Combinatorial Optimization*, 9(3):281–294, 2005.

[3] Jean Daligault and Stéphan Thomassé. On finding directed trees with many leaves. In *IPEC*, pages 86–97. 2009.

[4] Matthew Drescher and Adrian Vetta. An approximation algorithm for the maximum leaf spanning arborescence problem. *TALG*, 6(3):46, 2010.

[5] Giulia Galbiati, Francesco Maffioli, and Angelo Morzenti. A short note on the approximability of the maximum leaves spanning tree problem. *Information Processing Letters*, 52(1):45–49, 1994.

[6] Mohammad Taghi Hajiaghayi, Guy Kortsarz, Robert MacDavid, Manish Purohit, and Kanthi Sarpatwar. Approximation algorithms for connected maximum cut and related problems. In *ESA*, pages 693–704. 2015.

[7] Samir Khuller, Manish Purohit, and Kanthi K Sarpatwar. Analyzing the optimal neighborhood: Algorithms for budgeted and partial connected dominating set problems. In *SODA*, pages 1702–1713, 2014.

[8] Daniel J Kleitman and Douglas B West. Spanning trees with many leaves. *SIAM Journal on Discrete Mathematics*, 4(1):99–106, 1991.

[9] Jon Lee, Viswanath Nagarajan, and Xiangkun Shen. Max-cut under graph constraints. In *IPCO*, pages 50–62, 2016.

[10] Hsueh-I Lu and R Ravi. The power of local optimization: Approximation algorithms for maximum-leaf spanning tree. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 30, pages 533–533, 1992.

[11] Hsueh-I Lu and Ramamurthy Ravi. Approximating maximum leaf spanning trees in almost linear time. *Journal of Algorithms*, 29(1):132–141, 1998.

[12] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[13] Roberto Solis-Oba. 2-approximation algorithm for finding a spanning tree with maximum number of leaves. In *ESA*, pages 441–452, 1998.