Leafy spanning k-forests*

Cristina G. Fernandes[1], Carla N. Lintzmayer[2], Mário César San Felice[3]

[1] Instituto de Matemática e Estatística, Universidade de São Paulo, Brazil

[2]Centro de Matemática, Computação e Cognição, Universidade Federal do ABC, Brazil

[3]Departamento de Computação, Universidade Federal de São Carlos, Brazil

cris@ime.usp.br, carla.negri@ufabc.edu.br, felice@ufscar.br

Abstract. *We denote by* MAXIMUM LEAF SPANNING k-FOREST *the problem of, given a positive integer k and a graph G with at most k components, finding a spanning forest in G with at most k components and the maximum number of leaves. A leaf in a forest is defined as a vertex of degree at most one. The case $k = 1$ for connected graphs is known to be NP-hard, and is well studied in the literature, with the best approximation algorithm proposed more than 20 years ago by Solis-Oba. The best known approximation algorithm for* MAXIMUM LEAF SPANNING k-FOREST *with a slightly different leaf definition is a 3-approximation based on an approach by Lu and Ravi for the $k = 1$ case. We extend the algorithm of Solis-Oba to achieve a 2-approximation for* MAXIMUM LEAF SPANNING k-FOREST.

1. Introduction

The problem of, given a connected graph, finding a spanning tree with maximum number of leaves is well-known in the literature. With many applications in network design problems, the best known result for it is a 2-approximation algorithm proposed by Solis-Oba [Solis-Oba 1998, Solis-Oba et al. 2017] more than 20 years ago. We consider a generalization of this problem where the goal is to find a spanning forest with a specific number of components with as many leaves as possible.

For any graph G, let $c(G)$ denote the number of connected components of G. A forest F is called a *k-forest* if $c(F) \leq k$. The MAXIMUM LEAF SPANNING k-FOREST is the following problem: given a positive integer k and a graph G with $c(G) \leq k$, find a spanning k-forest of G with the maximum number of leaves. Let $\mathrm{opt}(G, k)$ denote the maximum number of leaves in such a forest. This problem was introduced by Reis *et al.* [Reis et al. 2017], who presented a 3-approximation algorithm for connected graphs, inspired by a 3-approximation for the case $k = 1$ [Lu and Ravi 1998].

For a forest F, there are two possible definitions of a leaf. A leaf could be defined as a vertex of degree at most one in F, or one can consider each component of F as a rooted tree, and define a leaf as a vertex of degree 1 in the forest which is not a root. The former is the definition we adopt, while the latter is the definition adopted by Reis *et al.* [Reis et al. 2017].

In this paper, we present a 2-approximation algorithm for MAXIMUM LEAF SPANNING k-FOREST, by adapting the 2-approximation of Solis-Oba for the case $k = 1$.

*C. G. Fernandes was partially supported by CNPq (Proc. 308116/2016-0 and 423833/2018-9). C. N. Lintzmayer was partially supported by CNPq (Proc. 428385/2018-4).

2. Revisiting Solis-Oba's algorithm

For a given graph H that has at least one vertex of degree 3 or more, a tree obtained by the procedure MAXIMALSOTREE, given in Algorithm 1, is called an *SO-tree*. The routine EXPAND(H, T, v) receives a tree T in H and a leaf v of T with at least two neighbors in $\overline{V(T)} = V(H) \setminus V(T)$ and adds to T the edges from v to all its neighbors in $\overline{V(T)}$, returning the resulting extended tree.

Algorithm 1 MAXIMALSOTREE(H, r)

Input: a graph H with a vertex r of degree at least 3 in H
Output: an SO-tree in H
1: let T be the tree consisting of r and its neighbors in H
2: $\mathrm{expanding} \leftarrow$ **true**
3: **while** $\mathrm{expanding}$ **do**
4:     **if** there is a leaf v in T with at least two neighbors not in T **then**
5:         $T \leftarrow$ EXPAND(H, T, v)
6:     **else if** there is a leaf v in T with a neighbor u not in T that has at least three neighbors not in T **then**
7:         $T \leftarrow T \cup \{uv\}$
8:         $T \leftarrow$ EXPAND(H, T, u)
9:     **else if** there is a leaf v in T with a neighbor u not in T that has two neighbors not in T **then**
10:         $T \leftarrow T \cup \{uv\}$
11:         $T \leftarrow$ EXPAND(H, T, u)
12:     **else** $\mathrm{expanding} \leftarrow$ **false**
13: **return** T

A *maximal SO-forest* is a maximal collection of disjoint SO-trees. Note that each SO-tree has at least one vertex of degree three or more. Algorithm 2, called SOFOREST(G), obtains a maximal SO-forest in a graph G.

Algorithm 2 SOFOREST(G)

Input: a graph G
Output: a maximal SO-forest in G
1: $F \leftarrow \emptyset$
2: $H \leftarrow G$
3: **while** there is a vertex r with degree at least 3 in H **do**
4:     $T \leftarrow$ MAXIMALSOTREE(H, r)
5:     $F \leftarrow F \cup \{T\}$
6:     $H \leftarrow H - V(T)$
7: **return** F

We call a vertex *tight* if it made the role of vertex u in line 9 during any execution of MAXIMALSOTREE(H, r). For a graph G, let B be the set of all tight vertices within an execution of SOFOREST(G). For a forest F, let $\ell(F)$ denote the number of leaves in F.

In the proof of Lemma 2 of [Solis-Oba et al. 2017], the authors proved the following about the forest F produced by SOFOREST(G):

$$\ell(F) \geq \frac{|V(F)| - |B|}{2} + c(F)\,. \qquad (1)$$

Corollary 10 of [Solis-Oba et al. 2017] states that, for a connected graph G and any spanning tree T of G,

$$\ell(T) \leq |V(F)| - |B| - 2c(F) + 3\,. \tag{2}$$

3. The algorithm

Next we present our algorithm for MAXIMUM LEAF SPANNING k-FOREST. We start by running the first phase of Solis-Oba algorithm, that builds a maximal SO-forest F, even if G is disconnected. Thus, the bound given by Eq. (1) is still valid.

Let $G_1, \ldots, G_{c(G)}$ be the components of G. For $1 \leq i \leq c(G)$, let F_i and B_i be, respectively, the maximal SO-forest F restricted to G_i, and set B restricted to G_i. Also, let T_i be any spanning tree of G_i, and let $\mathcal{F}$ be the forest containing such trees. Then, as in Eq. (2), it holds that $\ell(T_i) \leq |V(F_i)| - |B_i| - 2c(F_i) + 3$, and thus

$$\ell(\mathcal{F}) \leq |V(F)| - |B| - 2c(F) + 3c(G)\,. \tag{3}$$

We prove the following upper bound on $\mathrm{opt}(G, k)$ involving a maximal SO-forest.

Lemma 3.1. *Let F be a maximal SO-forest in a graph G with $c(G) \leq k$. Then*

$$\mathrm{opt}(G, k) \leq 2\,\ell(F) - 4c(F) + 2k + c(G)\,.$$

Consider a component G_i such that F_i is a nonempty forest. As Solis-Oba already observed, $G_i - V(F_i)$ is a set $\mathcal{P}_i$ of paths, because F_i is a maximal SO-forest in G_i. Moreover, all edges in G_i from $\mathcal{P}_i$ to F_i are incident to ends of paths in $\mathcal{P}_i$. Let $\mathcal{P}$ be the union of the sets $\mathcal{P}_i$. A component G_i where F_i is empty consists of a path or a cycle. Each such component contributes with at least one component in any spanning forest of G. Let $\mathcal{P}'$ be a set with one spanning path in each such path or cycle component of G.

From F, we create F' by adding the set $\mathcal{P}'$ to F. Our goal is to obtain from F' a spanning forest with at most k components, keeping as many leaves as possible. Then, if the number of components in F' is too small, its number of leaves might be small as well, so we try to augment it applying any of the two *improvement steps*:

Promotion step:	if there is a path P in $\mathcal{P}$, add P to F' and remove it from $\mathcal{P}$.
Division step:	if there is a degree-2 vertex v in F', remove an edge incident to v.

Note that each improvement step increases the number of components in F' by one and increases the number of leaves in F' by at least one. If, on the other hand, the number of components in $F' = F \cup \mathcal{P}'$ is too high, then we have to reduce it, which can be done by repeatedly connecting two components. After (possibly) applying these steps, the remaining paths in $\mathcal{P} = G - V(F')$ can be attached to F' to produce a spanning forest F'' with the same number of leaves and components as F'. The algorithm is formalized in Algorithm 3, and the next theorem guarantees its approximation ratio.

Theorem 3.2. LEAFYFOREST *is a 2-approximation for* MAXIMUM LEAF SPANNING k-FOREST.

Algorithm 3 LEAFYFOREST(G, k)

Input: a graph G and a positive integer $k \geq c(G)$
Output: a spanning k-forest with at least $\mathrm{opt}(G, k)/2$ leaves

$F \leftarrow$ SOFOREST(G)
let $\mathcal{P}'$ be a set which contains one spanning path in each path or cycle component of G
$F' \leftarrow F \cup \mathcal{P}'$
$\mathcal{P} \leftarrow G - V(F')$ ▷ $\mathcal{P}$ is a set of paths
$c \leftarrow c(F')$
if $c < 3k/4$ **then** ▷ improvement steps
    $p \leftarrow \min\{k - c, |\mathcal{P}|\}$
    add p paths from $\mathcal{P}$ to F' and remove them from $\mathcal{P}$
    $c \leftarrow c + p$
    while $c < k$ and there is a degree-2 vertex v in a component of F' **do**
        remove an edge incident to v from F'
        $c \leftarrow c + 1$
else ▷ components reduction steps
    while $c > k$ **do** ▷ $c = c(F') > c(G)$
        connect two components of F' from the same component of G through leaves
        $c \leftarrow c - 1$
let F'' be F' after attaching each path in $G - V(F')$ to a leaf of F'
return F''

4. Final remarks

For the alternative leaf definition, the difference occurs for the components that consist of a singleton or a single edge. Even though this difference seems small, so far we could not prove an approximation ratio better than 3 for Algorithm 3 with this different leaf definition, which would be an improvement of Reis, Felice, Lee, and Usberti [Reis et al. 2017]'s result. We are currently trying to adapt our algorithm to obtain a 2-approximation for their version of the problem.

Referências

Lu, H. and Ravi, R. (1998). Approximating maximum leaf spanning trees in almost linear time. *Journal of Algorithms*, 29(1):132–141.

Reis, M. F., Felice, M. C. S., Lee, O., and Usberti, F. L. (2017). A 3-approximation algorithm for the maximum leaf k-forest problem. *Electronic Notes in Discrete Mathematics*, 62:201–206.

Solis-Oba, R. (1998). 2-Approximation algorithm for finding a spanning tree with maximum number of leaves. In *Proceedings of the European Symposium on Algorithms (ESA)*, volume 1461 of *Lecture Notes in Computer Science*, pages 441–452.

Solis-Oba, R., Bonsma, P., and Lowski, S. (2017). A 2-approximation algorithm for finding a spanning tree with maximum number of leaves. *Algorithmica*, 77:374–388.