LEAFY SPANNING ARBORESCENCES IN DAGS

CRISTINA G. FERNANDES AND CARLA N. LINTZMAYER

ABSTRACT. Broadcasting in a computer network is a method of transferring a message to all recipients simultaneously. It is common in this situation to use a tree with many leaves to perform the broadcast, as internal nodes have to forward the messages received, while leaves are only receptors. We consider the subjacent problem of, given a directed graph D, finding a spanning arborescence of D, if one exists, with the maximum number of leaves. In this paper, we concentrate on the class of rooted directed acyclic graphs, for which the problem is known to be MaxSNP-hard. A 2-approximation was previously known for this problem on this class of directed graphs. We improve on this result, presenting a $\frac{3}{2}$-approximation. We also adapt a result for the undirected case and derive an inapproximability result for the vertex-weighted version of MAXIMUM LEAF SPANNING ARBORESCENCE on rooted directed acyclic graphs.

1. INTRODUCTION

The problem of, given a connected undirected graph, finding a spanning tree with the maximum number of leaves is well known in the literature, appearing as one of the NP-hard problems in the classic book by Garey and Johnson [13]. With many applications in network design problems, the best known result for it is a long standing 2-approximation proposed by Solis-Oba [21, 22]. In the literature, a directed version of this problem has also been considered.

For network broadcast, one looks for a directed spanning tree rooted at a source node, in which all arcs are directed away from the source. Broadcast trees with many leaves are preferable in this situation [17, 19]. Internal nodes have to forward the messages received, while leaves are only receptors. Also, in some applications, it is interesting to build a more robust backbone tree, and possibly less expensive links to reach the endpoint clients. The cost of such a backbone tree is usually related to its number of arcs. By maximizing the number of leaves in a rooted directed spanning tree, we are minimizing the number of arcs in the tree obtained from removing the arcs incident to the leaves, which can be seen as a backbone tree for the network. To define the directed version of the problem precisely, we introduce some notation.

Let D be a directed graph. A vertex r in D is a *root* if there is a directed path in D from r to every vertex in D. If r is a root in D, then we say D is *r-rooted*, or simply *rooted*. We say D is *acyclic* if there is no directed cycle in D. A directed acyclic graph is called a *dag*, for short. Note

that any rooted dag has only one root. An *arborescence* is an r-rooted dag T for which there is a unique directed path from r to every vertex in T. The *out-degree* of a vertex in a directed graph is the number of arcs that start in that vertex, while the *in-degree* of a vertex is the number of arcs that end in that vertex. A vertex of out-degree 0 in an arborescence is called a *leaf*.

The Maximum Leaf Spanning Arborescence is the problem of, given a rooted directed graph D, finding a spanning arborescence of D with the maximum number of leaves. Let $\mathrm{opt}(D)$ denote the number of leaves in such an arborescence.

Given an undirected graph G, one can consider the digraph D obtained by substituting each edge by two arcs, one in each direction. With this construction, it is easy to deduce that the Maximum Leaf Spanning Arborescence is NP-hard, as its undirected version. Alon et al. [1] showed that the Maximum Leaf Spanning Arborescence remains NP-hard on dags. They were in fact investigating whether the Maximum Leaf Spanning Arborescence is fixed parameter tractable [8], and they gave a positive answer for strongly connected digraphs, as well as for dags. Binkele-Raible et al. [4] provided a cubic size kernel for the Maximum Leaf Spanning Arborescence, and Daligault and Thomassé [9] improved on this result, providing a quadratic size kernel. It is worth mentioning that a linear size kernel is known for the undirected version of the problem.

As a byproduct, Daligault and Thomassé [9] derived a 92-approximation for the Maximum Leaf Spanning Arborescence in general rooted directed graphs. This turns into a 24-approximation when the digraph has no digon (directed cycle of length two). More recently, Schwartges, Spoerhase, and Wolff [20] described a 2-approximation for the case in which the digraph is acyclic, and proved that this restricted version of the Maximum Leaf Spanning Arborescence is MaxSNP-hard. Their algorithm is inspired on a greedy 3-approximation by Lu and Ravi [18] for the undirected version of the problem.

Sections 2 and 3 present a $\frac{3}{2}$-approximation algorithm for the Maximum Leaf Spanning Arborescence on rooted dags. Our algorithm is somehow inspired on Solis-Oba's algorithm, in the sense that it prioritizes certain expansion rules. However, there is a key difference: in one of the rules, the number of expansions can be optimized. Section 4 explores the relation of our algorithm with matchings. Section 5 shows an inapproximability result for the vertex-weighted version of Maximum Leaf Spanning Arborescence on rooted dags. Some open problems are presented in Section 6.

2. The algorithm

A *branching* is a forest of arborescences. A vertex that is not a leaf in a branching is called *internal*. For a positive integer t, a *t-branching* is a branching all of whose internal vertices have out-degree at least t. See Figure 1.

For a directed graph D, we denote by $V(D)$ and $A(D)$ the set of vertices and arcs of D respectively. For a vertex v in $V(D)$, we denote by $d_D^+(v)$ its out-degree in D and by $d_D^-(v)$ its in-degree in D. The *out-neighbors* of v are the extreme vertices of arcs that start at v. We say a
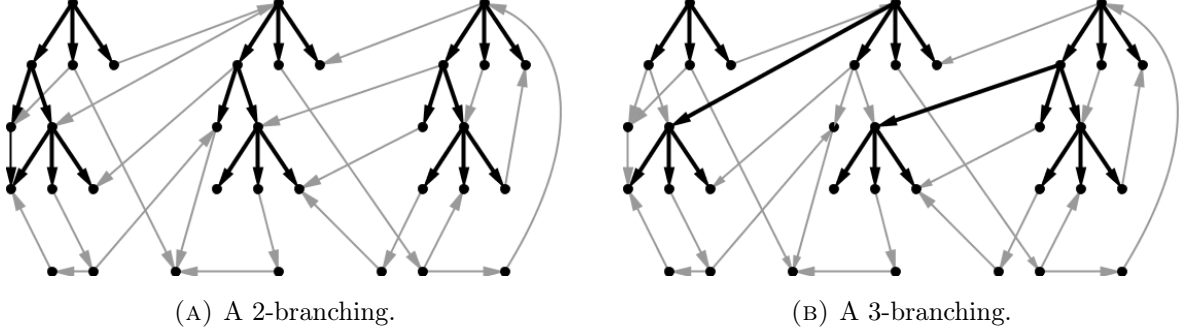
(A) A 2-branching.   (B) A 3-branching.

FIGURE 1. The bold arcs show two different branchings in a rooted dag.

spanning t-branching is maximal if, for any vertex of out-degree 0, its set of out-neighbors with in-degree 0 contains less than t vertices. The spanning branchings in Figure 1 are maximal.

Algorithm 1 presents GREEDYEXPAND(D, t, F), the heart of our approximation. Given a rooted dag D, a positive integer t, and a spanning $(t+1)$-branching F of D, it returns a maximal spanning t-branching of D containing F.

Algorithm 1 GREEDYEXPAND(D, t, F)

Input: rooted dag D, a positive integer t, and a spanning $(t+1)$-branching F of D

Output: a maximal spanning t-branching of D containing F

    $F' \leftarrow F$

    for each $v \in V(D)$ such that $d^+_{F'}(v) = 0$ **do**

        $U_v \leftarrow \{vu \in A(D) : d^-_{F'}(u) = 0\}$

        if $|U_v| \geq t$ **then**

            $F' \leftarrow F' + U_v$

    return F'

Let us argue that the call GREEDYEXPAND(D, t, F) produces a maximal t-branching. Indeed, the returned F' is spanning because F' contains the spanning branching F. Also, all internal vertices of F' have in-degree at most one and out-degree at least t. So F' is a t-branching and is clearly maximal.

The branchings in Figure 1 are possible outputs of the calls GREEDYEXPAND(D, 2, F) and GREEDYEXPAND(D, 3, F), respectively, when D is the depicted dag and F is the spanning branching of D with no arcs.

We observe that the GREEDYEXPAND is an extension of the EXPANSION algorithm by Schwartges, Spoerhase, and Wolff [20]. Particularly, if F is the spanning branching of D with no arcs, then GREEDYEXPAND(D, 2, F) behaves as EXPANSION(D) on any rooted dag D.

Algorithm 2 shows our approximation for the MAXIMUM LEAF SPANNING ARBORESCENCE on rooted dags, named MAXLEAVES. It uses twice the previously presented GREEDYEXPAND. Algorithm MAXLEAVES also uses an algorithm MAXEXPAND(D, F) that receives a rooted dag D

and a maximal spanning 3-branching F of D, and returns a maximum spanning 2-branching of D containing F. Algorithm $\mathrm{MAXEXPAND}(D, F)$ will be described after $\mathrm{MAXLEAVES}$.

Algorithm 2 $\mathrm{MAXLEAVES}(D)$

Input: rooted acyclic directed graph D
Output: spanning arborescence with at least $\frac{3}{2}\operatorname{opt}(D)$ leaves
  let F_0 be the spanning branching with no arcs
  $F_1 \leftarrow \mathrm{GREEDYEXPAND}(D, 3, F_0)$
  $F_2 \leftarrow \mathrm{MAXEXPAND}(D, F_1)$
  $T \leftarrow \mathrm{GREEDYEXPAND}(D, 1, F_2)$
  return T

The call $\mathrm{GREEDYEXPAND}(D, 1, F)$ returns a maximal 1-branching of the rooted dag D containing F, that is, a spanning arborescence of D containing F. So $\mathrm{MAXLEAVES}(D)$ indeed produces a spanning arborescence of D. See Figure 2. In the next section, we will prove that algorithm $\mathrm{MAXLEAVES}$ is a $\frac{3}{2}$-approximation for the MAXIMUM LEAF SPANNING ARBORESCENCE on rooted dags.
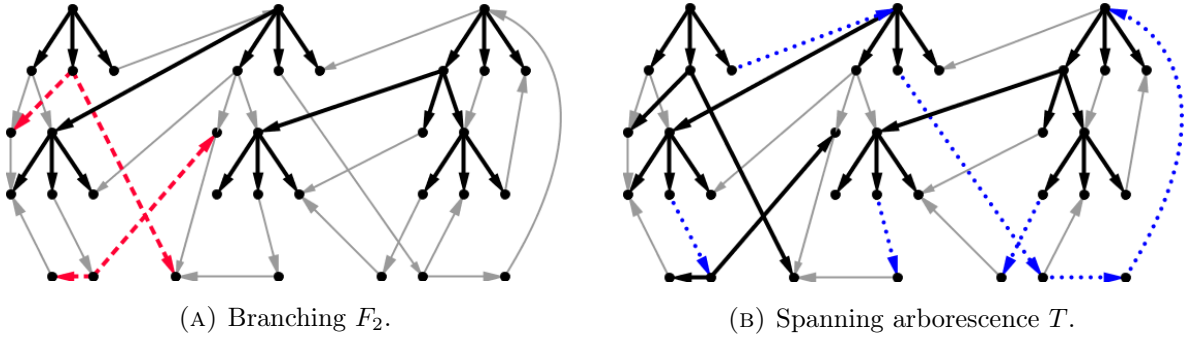

(A) Branching F_2.   (B) Spanning arborescence T.

FIGURE 2. In the left, the bold arcs represent a possible maximal 3-branching F_1 and the dashed arcs were added to obtain F_2. In the right, the bold arcs represent F_2 and the dotted arcs were added to obtain T.

The $\mathrm{MAXEXPAND}(D, F)$ procedure, presented in Algorithm 3, is an optimized version of $\mathrm{GREEDYEXPAND}(D, 2, F)$. It uses an algorithm $\mathrm{MAXIMUMMATCHING}$ that receives an undirected multigraph G and returns a maximum matching in G. Polynomial-time algorithms for this are known in the literature [10].

The call $\mathrm{MAXEXPAND}(D, F)$ produces a maximum spanning 2-branching of D containing F. It does this by constructing an undirected multigraph G whose vertices are vertices of in-degree 0 in F and an edge uw exists in G if u and w are the only out-neighbors of in-degree 0 in F of some vertex v of out-degree 0 in F. Thus, edge uw of G represents an expansion that can be performed on vertex v of D. The fact that more than one vertex of out-degree 0 in F may have vertices u and w of in-degree 0 in F as their out-neighbors shows the need for a multigraph. Independent

---

Algorithm 3 MAXEXPAND(D, F)

Input: rooted dag D and a maximal spanning 3-branching F of D

Output: a maximum spanning 2-branching of D containing F

    for each $v \in V(D)$ such that $d_F^+(v) = 0$ **do**

        $U_v \leftarrow \{vu \in A(D) : d_F^-(u) = 0\}$

    $Candidates \leftarrow \{v \in V(D) : d_F^+(v) = 0 \text{ and } |U_v| = 2\}$

    $V' \leftarrow \{u \in V(D) : d_F^-(u) = 0\}$

    $E' \leftarrow \{e_v = uw : v \in Candidates \text{ and } U_v = \{vu, vw\}\}$

    let G be the undirected multigraph (V', E')

    $M \leftarrow$ MAXIMUMMATCHING(G)

    $F' \leftarrow F$

    for each $e_v \in M$ **do**

        $F' \leftarrow F' + U_v$

    return F'

---

edges in this undirected multigraph correspond to compatible expansions, so a maximum matching gives the maximum number of expansions that can be performed in D. See Figure 3a.

Indeed, note that, for the returned F' to be a branching, the edges e_v corresponding to expanded vertices v must form a matching in the multigraph G. Otherwise, there would be vertices with in-degree greater than one. As F is a maximal 3-branching and D is acyclic, the returned F' is also a branching, and therefore a maximum 2-branching containing F. See Figure 3b.

We observe that, in the dag shown in Figure 3a, our algorithm produces the best arborescence possible, with roughly half of the vertices of the dag as leaves. Meanwhile, the algorithm due to Schwartges, Spoerhase, and Wolff [20] could have produced an arborescence with only one forth of the vertices as leaves.

3. APPROXIMATION RATIO

Let F_1, F_2, and T be the branchings produced in the call MAXLEAVES(D). For $i = 1, 2$, let k_i be the number of non-trivial components of F_i and N_i be the number of vertices in such components. We denote by $\ell(F)$ the number of leaves in any branching F.

For example, if D is the dag depicted in Figure 1, then F_1 could be the spanning 3-branching depicted in Figure 1b, F_2 could be the spanning 2-branching depicted in Figure 2a, and T could be the arborescence in Figure 2b. In this example, we have $k_1 = 3$, $N_1 = 25$, $k_2 = 4$, and $N_2 = 30$.

Lemma 3.1. *Let T be the arborescence produced by* MAXLEAVES(D). *Then*

$$\ell(T) \geq \frac{N_1 - k_1}{6} + \frac{N_2 - k_2}{2} + 1 .$$

Proof. Let n be the number of vertices of D. Let $T_1, \ldots, T_{k_1}$ be the non-trivial arborescences in F_1. Note that $\ell(T_j) \geq \frac{1 + 2|V(T_j)|}{3}$ because all internal vertices of T_j have out-degree at least 3.
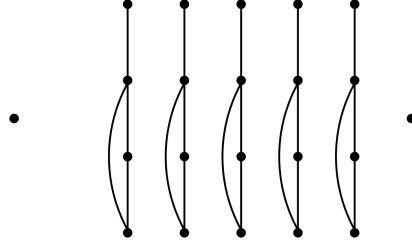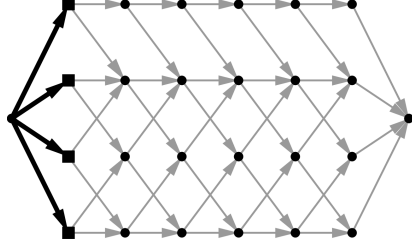
(A) The bold arcs incident to the root of the dag are a maximal 3-branching. The round vertices form the set V, and the corresponding multigraph G is in the right.
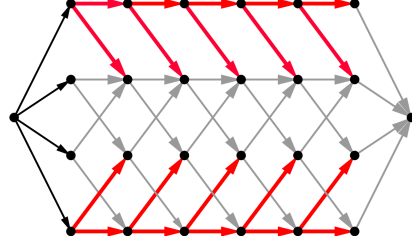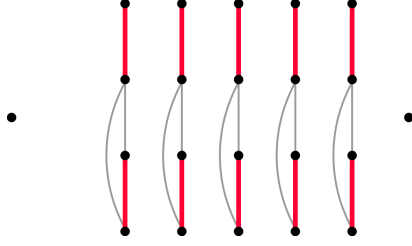
(B) In the left, a maximum matching in red and bold. In the right, the corresponding expansions in red and bold.

FIGURE 3. Example of an execution of MAXEXPAND.

Therefore,

$$\ell(F_1) \;=\; n - N_1 + \sum_{j=1}^{k_1} \ell(T_j) \;\geq\; n - N_1 + \sum_{j=1}^{k_1} \frac{1 + 2|V(T_j)|}{3}$$

$$=\; n - N_1 + \frac{2N_1}{3} + \frac{k_1}{3} \;=\; n - \frac{N_1 - k_1}{3} \;.$$

The number of components in F_i is $n - N_i + k_i$ for $i = 1, 2$. Hence, the number of leaves lost from F_1 to F_2 is exactly

$$\frac{(n - N_1 + k_1) - (n - N_2 + k_2)}{2} \;=\; \frac{N_2 - k_2}{2} - \frac{N_1 - k_1}{2} \;. \tag{1}$$

Also, the number of leaves lost from F_2 to T is exactly $n - N_2 + k_2 - 1 = n - (N_2 - k_2) - 1$. Thus

$$\ell(T) \;\geq\; n - \frac{N_1 - k_1}{3} - \left(\frac{N_2 - k_2}{2} - \frac{N_1 - k_1}{2} \right) - (n - (N_2 - k_2) - 1)$$

$$=\; \frac{N_1 - k_1}{6} + \frac{N_2 - k_2}{2} + 1 \;.$$

$\square$

For D, F_1, F_2, and T as in Figures 1 and 2, we have that $\ell(T) = 18$ and Lemma 3.1 gives as lower bound on $\ell(T)$

$$\frac{N_1 - k_1}{6} + \frac{N_2 - k_2}{2} + 1 \;=\; \frac{25 - 3}{6} + \frac{30 - 4}{2} + 1 \;=\; \frac{11}{3} + 14 = 17.666\ldots$$

Now we are going to present two upper bounds on $\mathrm{opt}(D)$. The following upper bound holds because the branching F_2 could be produced as output of the EXPANSION algorithm from Schwartges, Spoerhase, and Wolff [20].

Lemma 3.2 (Lemma 5 [20]). *It holds that* $\mathrm{opt}(D) \leq N_2 - k_2 + 1$.

The next lemma is the key for the approximation ratio analysis.

Lemma 3.3. *It holds that* $\mathrm{opt}(D) \leq \dfrac{N_1 - k_1}{2} + \dfrac{N_2 - k_2}{2} + 1$.

Proof. We apply on F_1 the same definition of witness that Schwartges, Spoerhase, and Wolff [20] used in their proof of Lemma 3.2. Let T^* be a spanning arborescence of D with the maximum number of leaves. Call R the set of all roots of non-trivial components of F_1. Call L the set of leaves of T^* that are isolated vertices of F_1. Let $Z := L \cup R \setminus \{r\}$, where r is the root of D. See Figure 4a. The witness of a vertex $z \in Z$ is the closest proper predecessor $q(z)$ of z in T^* which is in a non-trivial component of F_1. Note that each witness is an internal vertex of T^*. These witnesses will not necessarily be pairwise distinct, as in [20]. See Figure 4b.

(A) The red and bold arcs show T^*, and blue and square vertices show the set Z.

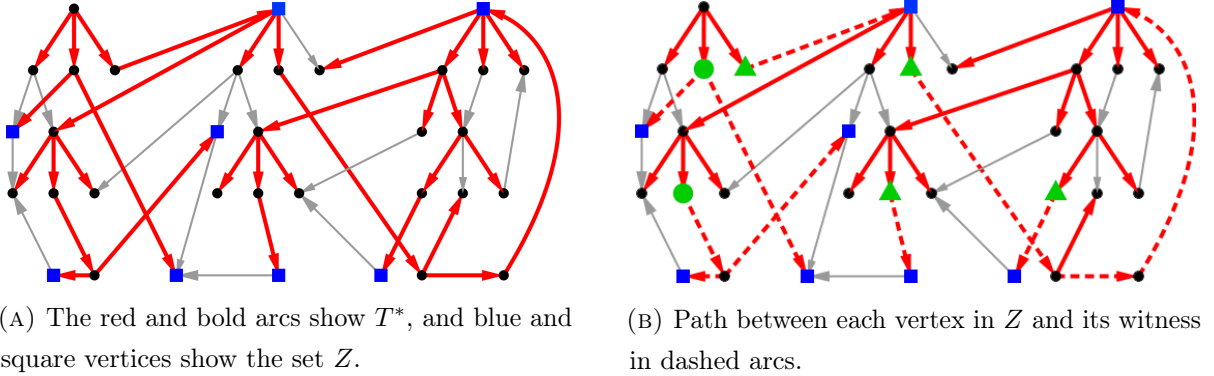(B) Path between each vertex in Z and its witness in dashed arcs.

FIGURE 4. The green triangular vertices are the witnesses for a vertex in Z and the green big circles mark two vertices that are the witnesses for two vertices in Z.

We will prove that the number w of distinct witnesses is

$$w \;\geq\; |Z| - \frac{N_2 - k_2}{2} + \frac{N_1 - k_1}{2} \;=\; k_1 - 1 + |L| - \frac{N_2 - k_2}{2} + \frac{N_1 - k_1}{2} \,. \qquad (2)$$

From this, because each witness lies in a non-trivial component of F_1 and is internal in T^*, we deduce that

$$\begin{aligned}
\mathrm{opt}(D) \;&\leq\; N_1 - \left(k_1 - 1 + |L| - \frac{N_2 - k_2}{2} + \frac{N_1 - k_1}{2} \right) + |L| \\
&=\; N_1 - k_1 + \frac{N_2 - k_2}{2} - \frac{N_1 - k_1}{2} + 1 \;=\; \frac{N_1 - k_1}{2} + \frac{N_2 - k_2}{2} + 1 \,.
\end{aligned}$$

It remains to prove (2).

For a witness s, let $Z_s := \{z \in Z : q(z) = s\}$. Let T_s^* be the subarborescence of T^* induced by the union of all paths in T^* from s to each vertex in Z_s. The number of such arborescences T_s^* is

exactly w. Note that the only internal vertex of T_s^* that is in a non-trivial component of F_1 is its root s, which is necessarily a leaf of F_1 (because there is no arc from an internal vertex of F_1 to vertices out of F_1). Thus the maximum out-degree in T_s^* is at most two.

First let us argue that no z in Z_s is a predecessor in T_s^* of another z' in Z_s. Suppose by contradiction that z is in the path from s to z'. Then z is not a leaf of T^* and therefore z is in R, and thus is in a non-trivial component of F_1. This leads to a contradiction because z, and not s, would be the witness for z'. Therefore T_s^* has exactly $|Z_s|$ leaves.

Let B be the set of vertices v such that $e_v \in M$, where M is the maximum matching in the multigraph G computed during the execution of $\text{MAXEXPAND}(D, F_1)$. Observe that $|M|$ is exactly the number of leaves lost from branching F_1 to F_2, given by (1), so

$$|M| \;=\; |B| \;=\; \frac{N_2 - k_2}{2} - \frac{N_1 - k_1}{2} \;. \tag{3}$$

Now let us argue that the vertices with out-degree two in T_s^* are all in the set *Candidates*, defined in Algorithm 3. Let v be one such vertex. Either v is an isolated vertex or v is a leaf of a non-trivial component of F_1. Therefore $d_{F_1}^+(v) = 0$. As the two children of v in T_s^* have in-degree 0 in F_1, both are in U_v. Hence $v \in$ *Candidates*.

Let C_s be the set of vertices of *Candidates* with out-degree two in T_s^* and $C = \cup_s C_s$. Then the number of leaves in T_s^* is $|Z_s| = |C_s| + 1$. The set of internal vertices of T_s^* and of $T_{s'}^*$ are disjoint for distinct witnesses s and s'. Thus the sets C_s and $C_{s'}$ are disjoint. Let M_C be the set of edges of G corresponding to the vertices in C. Note that M_C is a matching, so $|C| = |M_C| \le |M| = |B|$. Hence

$$|Z| \;=\; \sum_s |Z_s| \;=\; \sum_s (|C_s| + 1) \;=\; |C| + w \;\le\; |B| + w \;.$$

Therefore $w \ge |Z| - |B| = k_1 - 1 + |L| - \left(\frac{N_2 - k_2}{2} - \frac{N_1 - k_1}{2}\right)$, as in (2). $\qquad\square$

Continuing with our example, if D is the dag depicted in Figure 1, then Lemma 3.2 implies that $\text{opt}(D) \le 27$, while Lemma 3.3 implies that $\text{opt}(D) \le 25$.

Theorem 3.4. *Algorithm* MAXLEAVES *is a $\frac{3}{2}$-approximation for the* MAXIMUM LEAF SPANNING ARBORESCENCE *on rooted directed acyclic graphs.*

Proof. For a rooted dag D, let T be the output of MAXLEAVES(D). Then

$$\ell(T) \;\ge\; \frac{N_1 - k_1}{6} + \frac{N_2 - k_2}{2} + 1 \tag{4}$$

$$= \frac{N_1 - k_1}{6} + \frac{N_2 - k_2}{6} + \frac{N_2 - k_2}{3} + 1$$

$$\ge \frac{\text{opt}(D) - 1}{3} + \frac{\text{opt}(D) - 1}{3} + 1 \tag{5}$$

$$> 2\frac{\text{opt}(D)}{3} \;,$$

where (4) holds by Lemma 3.1 and (5) holds by Lemmas 3.2 and 3.3. $\qquad\square$

The bound given in Theorem 3.4 is tight. Indeed, an example similar to the one by Schwartges, Spoerhase, and Wolff [20] for their algorithm proves that algorithm MAXLEAVES can achieve ratios arbitrarily close to 3/2. See Figure 5.
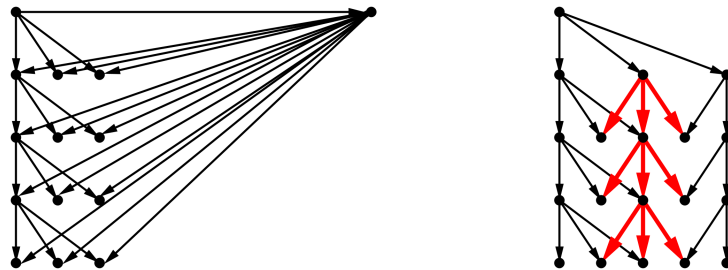

FIGURE 5. In the left, a rooted dag with $n = 3k + 2$ vertices, for $k = 4$, where MAXLEAVES can produce an arborescence with $2k + 2$ leaves and opt $= 3k$ leaves. In the right, an example where the 3-expansions in red and bold damage all possible 2-expansions that exist in the optimal arborecence.

4. USING APPROXIMATIONS FOR 3-DIMENSIONAL MATCHING

The problem known as 3D-MATCHING, from 3-dimensional matching, consists in the following. Given a finite set U and a collection $\mathcal{S}$ of subsets of U with three elements each, find a collection $\mathcal{S}' \subseteq \mathcal{S}$ of pairwise disjoint sets with as many sets as possible. The name of the problem comes from the fact that one can define a 3-regular hypergraph on the vertex set U whose edges are the sets in $\mathcal{S}$, and the collection $\mathcal{S}'$ is a maximum matching in such hypergraph.

This problem is NP-hard [13], and there exists a $\frac{4}{3}$-approximation for it [7, 12] as well as a $(2 + \epsilon)$-approximation for any $\epsilon > 0$, for its weighted version [2, 3].

The strategy in MAXLEAVES can be generalized by using an approximation algorithm for 3D-MATCHING. One possibility is, for a rooted dag D, to call GREEDYEXPAND(D, 4, F_0) with the empty spanning branching F_0, obtaining F_1, then to use the $\frac{4}{3}$-approximation for 3D-MATCHING to expand F_1 with a good set of 3-expansions, resulting in a branching F_2. Then we can proceed as in MAXLEAVES, calling MAXEXPAND(D, F_2) to obtain F_3, and GREEDYEXPAND(D, 1, F_3) to obtain the final arborescence T. This idea however will not give an approximation better than $\frac{3}{2}$. See the example to the right in Figure 5. It shows that, to achieve a ratio better than $\frac{3}{2}$, the choice of a good 3-dimensional matching has to somehow take into consideration the 2-expansions.

The following weighted variant of the previous idea takes into account 2-expansions and 3-expansions simultaneously. This variant does not give an improvement, but it can have implications for MAXIMUM LEAF SPANNING ARBORESCENCE on rooted dags if a better approximation for the weighted variant of 3D-MATCHING is designed. So we describe it ahead.

The second possibility we investigated makes use of weights in the following way. We start by calling GREEDYEXPAND(D, 4, F_0^w) with the empty spanning branching F_0^w, obtaining F_1^w.

After that, we create an instance of the weighted 3D-MATCHING where feasible 3-expansions turn into sets of weight two and feasible 2-expansions turn into sets of weight one. We then use an approximation for the weighted 3D-MATCHING to obtain a branching F_3^w from F_1^w. We finish by calling GREEDYEXPAND(D, 1, F_3^w) to obtain an arborescence. This algorithm, named MAXLEAVES-W3DM, is formalized in Algorithm 4. It makes use of a procedure named APPROXMAXWEIGHTED3DMATCHING($\mathcal{S}$, w), that receives a collection $\mathcal{S}$ of sets of size two or three, and a weight function w defined on $\mathcal{S}$. The procedure returns a subcollection of $\mathcal{S}$ that consists of pairwise disjoint sets. The weight of this subcollection depends on the approximation guarantee of the procedure.

To convert $\mathcal{S}$ into a collection of sets of size three, as in an instance of 3D-MATCHING, one can add a new element to each set of size two, keeping the same weights. For simplicity, we refrain from doing this, and in the proof of Lemma 4.2 ahead, we abuse notation and refer to subcollections of $\mathcal{S}$ of pairwise disjoint sets as 3D-matchings of $\mathcal{S}$.

Next lemma corresponds to Lemma 3.1, and their proofs are quite similar. Let F_1^w, F_2^w, F_3^w, and T be the branchings produced during the call MAXLEAVES-W3DM(D). For $i = 1, 2, 3$, let k_i^w be the number of non-trivial components of F_i^w and N_i^w be the number of vertices in such components.

Algorithm 4 MAXLEAVES-W3DM(D)

Input: rooted acyclic directed graph D

Output: spanning arborescence of D

   let F_0^w be the spanning branching with no arcs

   $F_1^w \leftarrow$ GREEDYEXPAND(D, 4, F_0^w)

   for each $v \in V(D)$ such that $d_{F_1^w}^+(v) = 0$ **do**

      $U_v \leftarrow \{vu \in A(D) : d_{F_1^w}^-(u) = 0\}$

   $Candidates \leftarrow \{v \in V(D) : d_{F_1^w}^+(v) = 0 \text{ and } 2 \leq |U_v| \leq 3\}$

   $\mathcal{S} \leftarrow \{U_v : v \in Candidates\}$

   $\mathcal{S} \leftarrow \mathcal{S} \cup \{\{a, b\}, \{a, c\}, \{b, c\} : v \in Candidates \text{ and } |U_v| = \{a, b, c\}\}$

   $w(U) \leftarrow |U| - 1$ for each $U \in \mathcal{S}$

   $M \leftarrow$ APPROXMAXWEIGHTED3DMATCHING($\mathcal{S}$, w)

   $F_2^w \leftarrow F_1^w$

   for each $U \in M$ such that $|U| = 3$ **do**

      $F_2^w \leftarrow F_2^w + U$

   $F_3^w \leftarrow F_2^w$

   for each $U \in M$ such that $|U| = 2$ **do**

      $F_3^w \leftarrow F_3^w + U$

   $T \leftarrow$ GREEDYEXPAND(D, 1, F_3^w)

   return T

Lemma 4.1. *Let T be the arborescence produced by* MaxLeaves-W3DM*(D). Then*

$$\ell(T) \;\geq\; \frac{N_1^w - k_1^w}{12} + \frac{N_2^w - k_2^w}{6} + \frac{N_3^w - k_3^w}{2} + 1 \;.$$

Proof. Let n be the number of vertices of D. Let $T_1, \ldots, T_{k_1^w}$ be the non-trivial arborescences in F_1^w. Note that $\ell(T_j) \geq \frac{1+3|V(T_j)|}{4}$ because all internal vertices of T_j have out-degree at least 4. Therefore,

$$\ell(F_1^w) \;=\; n - N_1^w + \sum_{j=1}^{k_1^w} \ell(T_j) \;\geq\; n - N_1^w + \sum_{j=1}^{k_1^w} \frac{1 + 3|V(T_j)|}{4}$$

$$= \; n - N_1^w + \frac{3N_1^w}{4} + \frac{k_1^w}{4} \;=\; n - \frac{N_1^w - k_1^w}{4} \;.$$

The number of components in F_i^w is $n - N_i^w + k_i^w$ for $i = 1, 2, 3$. Hence, the number of leaves lost from F_1^w to F_2^w is exactly

$$\frac{(n - N_1^w + k_1^w) - (n - N_2^w + k_2^w)}{3} \;=\; \frac{N_2^w - k_2^w}{3} - \frac{N_1^w - k_1^w}{3} \;.$$

Similarly, the number of leaves lost from F_2^w to F_3^w is exactly

$$\frac{(n - N_2^w + k_2^w) - (n - N_3^w + k_3^w)}{2} \;=\; \frac{N_3^w - k_3^w}{2} - \frac{N_2^w - k_2^w}{2} \;.$$

Also, the number of leaves lost from F_3^w to T is exactly $n - N_3^w + k_3^w - 1 = n - (N_3^w - k_3^w) - 1$. Thus

$$\ell(T) \;\geq\; n - \frac{N_1^w - k_1^w}{4} - \left(\frac{N_2^w - k_2^w}{3} - \frac{N_1^w - k_1^w}{3} \right)$$

$$- \left(\frac{N_3^w - k_3^w}{2} - \frac{N_2^w - k_2^w}{2} \right) - (n - (N_3^w - k_3^w) - 1)$$

$$= \; \frac{1}{12}(N_1^w - k_1^w) + \frac{1}{6}(N_2^w - k_2^w) + \frac{1}{2}(N_3^w - k_3^w) + 1 \;.$$

$\square$

Now we present an upper bound on $\mathrm{opt}(D)$ that relates to the lower bound presented in Lemma 4.1. Its proof follows the lines of the proof of Lemma 3.3.

Lemma 4.2. *If algorithm* ApproxMaxWeighted3DMatching *is an α-approximation for the weighted* 3D-Matching*, then*

$$\mathrm{opt}(D) \;\leq\; \frac{3 - 2\alpha}{3}(N_1^w - k_1^w) + \frac{\alpha}{6}(N_2^w - k_2^w) + \frac{\alpha}{2}(N_3^w - k_3^w) + 1.$$

Proof. Let T^* be a spanning arborescence of D with the maximum number of leaves. Call R the set of all roots of non-trivial components of F_1^w. Call L the set of leaves of T^* that are isolated vertices of F_1^w. Let $Z := L \cup R \setminus \{r\}$, where r is the root of D. The witness of a vertex $z \in Z$ is the closest proper predecessor $q(z)$ of z in T^* which is in a non-trivial component of F_1^w. Note that each witness is an internal vertex of T^*.

We will prove that the number ψ of distinct witnesses is

$$\psi \geq |Z| - 2\alpha \left(\frac{N_2^w - k_2^w}{3} - \frac{N_1^w - k_1^w}{3} \right) - \alpha \left(\frac{N_3^w - k_3^w}{2} - \frac{N_2^w - k_2^w}{2} \right) \tag{6}$$

$$= |Z| + 2\alpha \frac{N_1^w - k_1^w}{3} - \alpha \frac{N_2^w - k_2^w}{6} - \alpha \frac{N_3^w - k_3^w}{2} \ .$$

Because $|Z| = k_1^w - 1 + |L|$ and each witness lies in a non-trivial component of F_1^w and is internal in T^*, we deduce that

$$
\begin{aligned}
\mathrm{opt}(D) \ &\leq \ N_1^w - \psi + |L| \\
&\leq \ N_1^w - |Z| - 2\alpha \frac{N_1^w - k_1^w}{3} + \alpha \frac{N_2^w - k_2^w}{6} + \alpha \frac{N_3^w - k_3^w}{2} + |L| \\
&= \ N_1^w - k_1^w - 2\alpha \frac{N_1^w - k_1^w}{3} + \alpha \frac{N_2^w - k_2^w}{6} + \alpha \frac{N_3^w - k_3^w}{2} + 1 \\
&= \ \frac{3 - 2\alpha}{3}(N_1^w - k_1^w) + \frac{\alpha}{6}(N_2^w - k_2^w) + \frac{\alpha}{2}(N_3^w - k_3^w) + 1 \ .
\end{aligned}
$$

It remains to prove (6). The proof follows closely to that of (2).

For a witness s, let $Z_s := \{z \in Z : q(z) = s\}$ and let T_s^* be the subarborescence of T^* induced by the union of all paths in T^* from s to each vertex in Z_s. The number of such arborescences T_s^* is exactly ψ. The only internal vertex of T_s^* that is in a non-trivial component of F_1^w is its root s, which is necessarily a leaf of F_1^w. So the maximum out-degree in T_s^* is at most three.

Again, no $z \in Z_s$ is a predecessor in T_s^* of another $z' \in Z_s$. Indeed, suppose by contradiction that z is in the path from s to z'. Then z is not a leaf of T^*, and is in R, thus being in a non-trivial component of F_1^w, which is a contradiction, because z, and not s, would be the witness for z'. Hence T_s^* has exactly $|Z_s|$ leaves.

When $|U_v| = 3$, the algorithm includes in $\mathcal{S}$ also the subsets of U_v with two elements. At most one among U_v and these subsets is included in M, where M is the output of Approx-MaxWeighted3DMatching$(\mathcal{S}, w)$, computed during the execution of MaxLeaves-W3DM(D). Thus, in what follows, we abuse notation and refer to each such subset of U_v also as U_v. Let B_i be the set of vertices v such that $U_v \in M$ and $|U_v| = i$, for $i = 2, 3$. Note that $|B_3|$ is exactly the number of leaves lost from branching F_1^w to F_2^w, so

$$|B_3| \ = \ \frac{N_2^w - k_2^w}{3} - \frac{N_1^w - k_1^w}{3} \ . \tag{7}$$

Also, $|B_2|$ is exactly the number of leaves lost from branching F_2^w to F_3^w, so

$$|B_2| \ = \ \frac{N_3^w - k_3^w}{2} - \frac{N_2^w - k_2^w}{2} \ . \tag{8}$$

Finally, $|M| = |B_3| + |B_2|$ and $w(M) = 2|B_3| + |B_2|$.

Vertices with out-degree two and three in T_s^* are all in the set *Candidates*. Indeed, let v be one such vertex. Either v is an isolated vertex or v is a leaf of a non-trivial component of F_1^w. So $d_{F_1^w}^+(v) = 0$. As the children of v in T_s^* have in-degree 0 in F_1^w, they are all in U_v. Hence $v \in$ *Candidates*.

For $i = 2, 3$, let C_s^i be the set of vertices of *Candidates* with out-degree i in T_s^*, and let $C = \cup_s C_s^i$. The number of leaves in T_s^* is $|Z_s| = 2|C_s^3| + |C_s^2| + 1$. The set of internal vertices of T_s^* and of $T_{s'}^*$ are disjoint for distinct witnesses s and s'. Thus the sets C_s^i and $C_{s'}^i$ are disjoint. Let M_C be the subset of $\mathcal{S}$ corresponding to the vertices in C. Note that M_C is a 3D-matching of $\mathcal{S}$, so $w(M_C) = 2|C_s^3| + |C_s^2| \leq w(M^*) \leq \alpha \, w(M)$, where M^* is a maximum weighted 3D-matching of $\mathcal{S}$. Hence

$$|Z| \;=\; \sum_s |Z_s| \;=\; \sum_s (2|C_s^3| + |C_s^2| + 1) \;=\; w(M_C) + \psi$$

$$\leq \; \alpha \, w(M) + \psi \;=\; 2\alpha|B_3| + \alpha|B_2| + \psi \, .$$

Therefore,

$$\psi \;\geq\; |Z| - 2\alpha|B_3| - \alpha|B_2|$$

$$=\; |Z| - 2\alpha \left(\frac{N_2^w - k_2^w}{3} - \frac{N_1^w - k_1^w}{3} \right) - \alpha \left(\frac{N_3^w - k_3^w}{2} - \frac{N_2^w - k_2^w}{2} \right) \, ,$$

which completes the proof of (6). $\qquad\square$

Theorem 4.3. *If algorithm* ApproxMaxWeighted3DMatching *is an α-approximation for the weighted* 3D-Matching, *then algorithm* MaxLeaves-W3DM *is a $\max\{\frac{4}{3}, \alpha\}$-approximation for the* Maximum Leaf Spanning Arborescence *on rooted directed acyclic graphs.*

Proof. For a rooted dag D, let T be the output of MaxLeaves-W3DM(D). First, suppose $\alpha \geq \frac{4}{3}$. In this case, $\frac{3-2\alpha}{3} \leq \frac{\alpha}{12}$ and, by Lemmas 4.1 and 4.2,

$$\text{opt}(D) \;\leq\; \frac{3 - 2\alpha}{3}(N_1^w - k_1^w) + \frac{\alpha}{6}(N_2^w - k_2^w) + \frac{\alpha}{2}(N_3^w - k_3^w) + 1$$

$$\leq\; \frac{\alpha}{12}(N_1^w - k_1^w) + \frac{\alpha}{6}(N_2^w - k_2^w) + \frac{\alpha}{2}(N_3^w - k_3^w) + \alpha$$

$$\leq\; \alpha \, \ell(T) \, .$$

Now, suppose $\alpha < \frac{4}{3}$, and let $\beta = \frac{4}{3} - \alpha$. By Lemma 4.2,

$$\text{opt}(D) \;\leq\; \frac{3 - 2\alpha}{3}(N_1^w - k_1^w) + \frac{\alpha}{6}(N_2^w - k_2^w) + \frac{\alpha}{2}(N_3^w - k_3^w) + 1$$

$$\leq\; \left(\frac{1}{9} + \frac{2}{3}\beta \right)(N_1^w - k_1^w) + \left(\frac{2}{9} - \frac{1}{6}\beta \right)(N_2^w - k_2^w)$$

$$+ \left(\frac{2}{3} - \frac{1}{2}\beta \right)(N_3^w - k_3^w) + 1$$

$$=\; \frac{1}{9}(N_1^w - k_1^w) + \frac{2}{9}(N_2^w - k_2^w) + \frac{2}{3}(N_3^w - k_3^w) + \frac{4}{3}$$

$$+ \frac{2}{3}\beta(N_1^w - k_1^w) - \frac{1}{6}\beta(N_2^w - k_2^w) - \frac{1}{2}\beta(N_3^w - k_3^w) - \frac{1}{3}$$

$$\leq\; \frac{4}{3}\ell(T) + \frac{2}{3}\beta \left((N_1^w - k_1^w) - \frac{1}{4}(N_2^w - k_2^w) - \frac{3}{4}(N_3^w - k_3^w) \right) - \frac{1}{3} \qquad (9)$$

$$\leq\; \frac{4}{3}\ell(T) \, , \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (10)$$

where (9) holds by Lemma 4.1 and (10) holds because the number of components in F_1^w, F_2^w, and F_3^w is so that $n - N_1^w + k_1^w \geq n - N_2^w + k_2^w \geq n - N_3^w + k_3^w$, and this implies that $N_1^w - k_1^w \leq N_2^w - k_2^w \leq N_3^w - k_3^w$, and therefore $N_1^w - k_1^w \leq \frac{1}{4}(N_2^w - k_2^w) + \frac{3}{4}(N_3^w - k_3^w)$. $\qquad\square$

Note that the weighted instance of 3D-MATCHING we used has only weights 1 and 2. So a good approximation even for this more restricted weighted version of 3D-MATCHING would be of interest.

At the moment, because the best approximation for the weighted 3D-MATCHING has ratio greater than 2, this does not provide an improvement on the previously best known ratio for MAXIMUM LEAF SPANNING ARBORESCENCE. Now, only a ratio better than 3/2 for the weighted 3D-MATCHING would provide an improvement.

5. INAPPROXIMABILITY OF THE VERTEX-WEIGHTED VERSION

A vertex-weighted generalization of the maximum leaf spanning tree (the undirected version of our problem) was considered in the literature. In such generalization, one is given a connected vertex-weighted graph and the goal is to find a spanning tree whose sum of leaf weights is maximum.

Jansen [15] proved that, unless P = NP, this version of the problem does not admit a polynomial-time ratio $O(n^{\frac{1}{2}-\epsilon})$ or a $O(\text{opt}^{\frac{1}{3}-\epsilon})$-approximation for any $\epsilon > 0$, where n is the number of vertices of the given graph. His reduction is from the INDEPENDENT SET problem. A straightforward modification of his reduction shows the same inapproximability results for the vertex-weighted version of MAXIMUM LEAF SPANNING ARBORESCENCE on rooted dags. Next we describe his reduction adapted to produce rooted dags with binary weights.

The INDEPENDENT SET problem consists of the following: given a graph G, find an independent set in G with as many vertices as possible.

Let G be an instance of the INDEPENDENT SET problem. Let D be the rooted dag that has as vertices the vertices of G, a new vertex r as its root, and a vertex e for each edge e of G. There is an arc from r to each vertex of G in D. For each edge $e = uv$ of G, there is an arc from u to e and an arc from v to e in D. So, if G has n vertices and m edges, D has $n + m + 1$ vertices and $n + 2m$ arcs. See Figure 6. Note that D is r-rooted and acyclic and that, in any spanning arborescence in D, the vertices corresponding to edges of G are leaves, because they have out-degree 0 in D. Because the complement of an independent set is an edge cover, the following holds.

Lemma 5.1. *Set S is an independent set in G if and only if there is a spanning arborescence in D that has $S \cup E(G)$ as leaves.*

Proof. Let S be an independent set in G. Initialize an arborescence T by taking the root of D and adding arcs to every vertex of $V(G)$. Since $V(G) \setminus S$ is an edge cover in G, we can augment T to a spanning arborescence by connecting vertices of D in $V(G) \setminus S$ to vertices of D in $E(G)$. Thus all vertices in $S \cup E(G)$ are leaves of T.
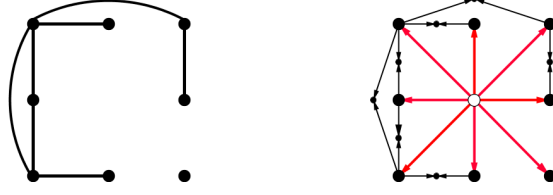
FIGURE 6. An instance of the INDEPENDENT SET problem and the corresponding rooted dag. The white vertex is the root of the dag.

Now let T be a spanning arborescence of D that has $S \cup E(G)$ as leaves, with $S \subseteq V(G)$. (Recall that any spanning arborescence of D has all vertices in $E(G)$ as leaves.) Thus, for $u, v \in S$, if $uv \in E(G)$, then $e = uv \in V(D)$ would be an isolated vertex in T because both of its in-neighbors u and v are leaves in T, a contradiction. Therefore S is an independent set of G. $\quad\square$

For any $\epsilon > 0$, there is no polinomial-time $O(n^{1-\epsilon})$-approximation for INDEPENDENT SET unless P = NP [14], where n is the number of vertices of the given graph G. Using this and Lemma 5.1, we derive the following.

Theorem 5.2. *The vertex-weighted* MAXIMUM LEAF SPANNING ARBORESCENCE *on directed acyclic graphs with binary weights and maximum in-degree 2 does not have a polynomial-time* $O(n^{1-\epsilon})$-*approximation for any* $\epsilon > 0$, *unless* P = NP, *where* n *is the number of weight-one vertices of the given directed graph.*

Proof. We will describe an approximation-preserving reduction from INDEPENDENT SET to the vertex-weighted MAXIMUM LEAF SPANNING ARBORESCENCE.

Let G be an instance of the INDEPENDENT SET problem. Let D be the rooted dag defined from G as before Lemma 5.1. Assign weights to the vertices of D as follows: vertices of G have weight 1 while vertices corresponding to edges of G have weight 0. The root r may have an arbitrary weight, because it will never be a leaf in a spanning arborescence of D. Note that the weights are binary and that the maximum in-degree in D is 2.

Let T^* be a maximum leaf weighted arborescence of D and S^* be a maximum independent set in G. Note that $w(T^*) = |S^*|$ by Lemma 5.1.

Suppose that A is an $O(n^{1-\epsilon})$-approximation, for some $\epsilon > 0$, for the vertex-weighted MAXIMUM LEAF SPANNING ARBORESCENCE on dags with binary weights and maximum in-degree 2, where n is the number of weight-one vertices of the given directed graph. Let T be the spanning arborescence of D obtained from applying A to D with weights w. Then, for some constant $c > 0$, we have that $w(T^*) \leq c\,n^{1-\epsilon}\,w(T)$, where $n = |V(G)|$. Let S be the set of $w(T)$ leaves of T in $V(G)$ which, by Lemma 5.1, form an independent set in G. Hence,

$$|S| \;=\; w(T) \;\geq\; \frac{w(T^*)}{c\,n^{1-\epsilon}} \;=\; \frac{|S^*|}{c\,n^{1-\epsilon}} \;. \tag{11}$$

This would then be an $O(n^{1-\epsilon})$-approximation for the INDEPENDENT SET, which exists only if P = NP by [14]. $\quad\square$

To avoid using weight zero, a similar result can be obtained by assigning weights m and 1 instead of 1 and 0, respectively, where m is the number of edges in G. For this case, Lemma 5.1 implies that an independent set of size t in G corresponds to a spanning arborescence of leaf weight $(t+1)m$, and a similar inapproximability result holds, as Jansen [15] proved for the undirected version.

6. Future directions

Improving on the 92-approximation for the general directed case would be very interesting. A major difficulty is that greedy strategies do not apply so easily, because not every branching can be extended to a spanning branching in an arbitrary rooted digraph. The strategy used by Daligault and Thomassé [9] consists of a series of reductions, and some of them end up with a dag. It is tempting to try to use an approximation for dags within their algorithm to achieve an improved ratio, however we did not succeed in doing that so far.

Directed acyclic graphs have directed tree width zero [16]. Is it possible to extend our approximation or any greedy algorithm for Maximum Leaf Spanning Arborescence to address directed graphs with bounded directed tree width?

It is natural to wonder if there is a way to optimize one of the expansions used in Solis-Oba's algorithm to achieve a better approximation ratio for the undirected case. Also, for the undirected case, there are better approximations for cubic graphs [5, 6]. Maybe one can obtain better bounds on the approximation ratio for dags with out-degree bounded by three or two.

References

[1] N. Alon, F. Fomin, G. Gutin, M. Krivelevich, and S. Saurabh. Spanning directed trees with many leaves. *SIAM Journal on Discrete Mathematics*, 23(1):466–476, 2009.

[2] E. M. Arkin and R. Hassin. On local search for weighted k-set packing. *Mathematics of Operations Research*, 23(3):640–648, 1998.

[3] P. Berman. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. *Nord. J. Comput.*, 7(3):178–184, 2000.

[4] D. Binkele-Raible, H. Fernau, F. V. Fomin, D. Lokshtanov, S. Saurabh, and Y. Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Transactions on Algorithms*, 8(4), 2012. article 38.

[5] P. Bonsma and F. Zickfeld. A 3/2-approximation algorithm for finding spanning trees with many leaves in cubic graphs. *SIAM Journal on Discrete Mathematics*, 25(4):1652–1666, 2011.

[6] J.R. Correa, C. G. Fernandes, M. Matamala, and Y. Wakabayashi. A 5/3-approximation for finding spanning trees with many leaves in cubic graphs. In *Proceedings of the 5th International Workshop on Approximation and Online Algorithms (WAOA 2007)*, volume 4927 of *Lecture Notes in Computer Science*, pages 184–192, 2008.

[7] M. Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *Proceedings of the IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 509–518, 2013.

[8] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2016.

[9] J. Daligault and S. Thomassé. On finding directed trees with many leaves. In *International Workshop on Parameterized and Exact Computation*, volume 5917 of *Lecture Notes in Computer Science*, pages 86–97, 2009.

[10] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[11] C. G. Fernandes and C. N. Lintzmayer. Leafy spanning arborescences in DAGs. In *Proceedings of the 14th Latin American Symposium on Theoretical Informatics (LATIN)*, Lecture Notes in Computer Science, 2020. to appear.

[12] M. Fürer and H. Yu. Approximating the k-set packing problem by local improvements. In *Proceedings of the International Symposium on Combinatorial Optimization (ISCO)*, volume 8596 of *Lecture Notes in Computer Science*, pages 408–420, 2014.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W.H. Freeman and Co., New York, 1979.

[14] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182(1):105–142, 1999.

[15] B. M. P. Jansen. Kernelization for maximum leaf spanning tree with positive vertex weights. *Journal of Graph Algorithms and Applications*, 16(4):811–846, 2012.

[16] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed tree-width. *Journal of Combinatorial Theory, Series B*, 82:138–154, 2001.

[17] A. Jüttner and A. Magi. Tree based broadcast in ad hoc networks. *Mobile Networks and Applications*, 10(5):753–762, 2005.

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

[19] J. Pope and R. Simon. Efficient one-to-many broadcasting for resource-constrained wireless networks. In *Proceedings of the 40th Annual IEEE Conference on Local Computer Networks (LCN)*, page 8 pp., 2015.

[20] N. Schwartges, J. Spoerhase, and A. Wolff. Approximation algorithms for the Maximum Leaf Spanning Tree Problem on acyclic digraphs. In *9th Workshop on Approximation and Online Algorithms (WAOA)*, volume 7164 of *Lecture Notes in Computer Science*, pages 77–88, 2011.

[21] R. Solis-Oba. 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, 1998.

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

Institute of Mathematics and Statistics. University of São Paulo. São Paulo, Brazil
E-mail address: `cris@ime.usp.br`

Center for Mathematics. Computing and Cognition. Federal University of ABC. Santo André, São Paulo, Brazil
E-mail address: `carla.negri@ufabc.edu.br`