Egerváry Research Group
on Combinatorial Optimization

Technical reportS

TR-2014-02. Published by the Egerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: www.cs.elte.hu/egres . ISSN 1587–4451.

Blocking unions of arborescences

Attila Bernáth and Gyula Pap

April 2014
Revised: May 2015

Blocking unions of arborescences

Attila Bernáth* and Gyula Pap**

Abstract: Given a digraph $D = (V, A)$ and a positive integer k, a subset $B \subseteq A$ is called a **k-union-arborescence**, if it is the disjoint union of k spanning arborescences. When also arc-costs $c : A \to \mathbb{R}$ are given, minimizing the cost of a k-union-arborescence is well-known to be tractable. In this paper we take on the following problem: what is the minimum cardinality of a set of arcs the removal of which destroys every minimum c-cost k-union-arborescence. Actually, the more general weighted problem is also considered, that is, arc weights $w : A \to \mathbb{R}_+$ (unrelated to c) are also given, and the goal is to find a minimum weight set of arcs the removal of which destroys every minimum c-cost k-union-arborescence. An equivalent version of this problem is where the roots of the arborescences are fixed in advance. In an earlier paper [2] we solved this problem for $k = 1$. This work reports on other partial results on the problem. We solve the case when both c and w are uniform – that is, find a minimum size set of arcs that covers all k-union-arbosercences. Our algorithm runs in polynomial time for this problem. The solution uses a result of Bárász, Becker and Frank [1] saying that the family of so-called insolid sets (sets with the property that every proper subset has a larger in-degree) satisfies the Helly-property, and thus can be (efficiently) represented as a subtree hypergraph. We also give an algorithm for the case when only c is uniform but w is not. This algorithm is only polynomial if k is not part of the input.

Keywords: arborescences, polynomial algorithm, covering

*MTA-ELTE Egerváry Research Group, Department of Operations Research, Eötvös University, Pázmány Péter sétány 1/C, Budapest, Hungary, H-1117. Part of the research was done while the author was at Warsaw University, Institute of Informatics, ul. Banacha 2, 02-097 Warsaw, Poland. Research was supported by the ERC StG project PAAl no. 259515. E-mail: `bernath@cs.elte.hu`.

**MTA-ELTE Egerváry Research Group, Department of Operations Research, Eötvös University, Pázmány Péter sétány 1/C, Budapest, Hungary, H-1117. Supported by OTKA grant no. K109240. E-mail: `gyuszko@cs.elte.hu`.

1 Introduction

Let $D = (V, A)$ be a digraph with vertex set V and arc set A. A **spanning arborescence** is a subset $B \subseteq A$ that is a spanning tree in the undirected sense, and every node has in-degree at most one. Thus there is exactly one node, the **root node**, with in-degree zero. Equivalently, a spanning arborescence is a subset $B \subseteq A$ with the property that there is a root node $r \in V$ such that $\varrho_B(r) = 0$, and $\varrho_B(v) = 1$ for $v \in V - r$, and B contains no cycle. We will also call a spanning arborescence as an **arborescence** for short, when the set of nodes is obvious from context. If $r \in V$ is the root of the spanning arborescence B then B is said to be an **r-arborescence**.

Given also a positive integer k, a subset $B \subseteq A$ is called a **k-union-arborescence**, if it is the arc-disjoint union of k spanning arborescences. In the special case when every arborescence has the same root r, we call B a **k-union-r-arborescence**.

Given $D = (V, A)$, k and a cost function $c : A \to \mathbb{R}$, it is well known how to find a **minimum cost k-union-r-arborescence** in polynomial time, and to find a **minimum cost k-union-arborescence** just as well. See [10], Chapter 53.8 for a reference, where several related problems are considered. The existence of a k-union-r-arborescence is characterized by Edmonds' Disjoint Arborescence Theorem (Theorem 2), while the existence of a k-union-arborescence is characterized by a theorem of Frank [4] (see Theorem 3 below). Frank also gave a linear programming description of the convex hull of k-union-arborescences, generalizing Edmonds' linear programming description of the convex hull of k-union-r-arborescences. The problem of finding a minimum cost k-union-arborescence may also be solved with the use of these results, either via a reduction to minimum cost k-union-r-arborescences, or minimum weight matroid intersection.

In this paper we consider the following covering problems, which are polynomial time equivalent.

Problem 1 (Blocking optimal k-union-arborescences). *Given a digraph $D = (V, A)$, a positive integer k, a cost function $c : A \to \mathbb{R}$ and a nonnegative weight function $w : A \to \mathbb{R}_+$, find a subset H of the arc set such that H intersects every minimum c-cost k-union-arborescence, and $w(H)$ is minimum.*

Here the expression "intersects" simply means that the two have nonempty intersection. We remark that Problem 1 is polynomially equivalent with the version where the root is also given in advance, that is, the problem of blocking optimal k-union-r-arborescences.

Problem 2 (Blocking optimal k-union-r-arborescences). *Given a digraph $D = (V, A)$, a node $r \in V$, a positive integer k, a cost function $c : A \to \mathbb{R}$ and a nonnegative weight function $w : A \to \mathbb{R}_+$, find a subset H of the arc set such that H intersects every minimum c-cost k-union-r-arborescence, and $w(H)$ is minimum.*

In section 3 we will show that the two problems are polynomial time equivalent.

In our previous paper [2] we have solved Problem 1 and Problem 2 in the special case when $k = 1$. Our conjecture is that the problem is also polynomial time solvable when k is not fixed. The main result of this paper is that the problem is polynomial

time solvable when k is part of the input, and both c and w are set to be constant – that is, when H is required to intersect *every* k-union-arborescence and we want to minimize $|H|$. We also give algorithms for Problems 1 and 2 when c is constant but w is not, but these algorithms are only polynomial if k is not part of the input.

We remark that the version of Problem 1 and Problem 2 where we set c to be constant are *not* easily seen to be equivalent: at least we did not find such reductions (find more details in Section 4). Thus it is important to note that in the sequel we will consider the version of Problem 1 (and not Problem 2 !) with constant cost function c.

The rest of the paper is organized as follows. In Section 2 we introduce some notation that we will use. In Section 3 we show that Problems 1 and 2 are polynomial-time equivalent. In Section 4 we provide general observations on the uniform cost version of Problems 1 and 2. In Section 5 we give a polynomial algorithm solving Problem 2 in the case when both c and w are uniform. Finally, in Section 6 we deal with the uniform cost general weight versions: in Section 6.1 we give an algorithm for the weighted blocking of k-union-r-arborescences, while in Section 6.2 we give an algorithm for the weighted blocking of k-union-arborescences. These algorithms are only polynomial if k is not part of the input.

2 Notation

Let us overview some of the notation and definitions used in the paper. The arc set of the digraph D will also be denoted by $A(D)$. Given a digraph $D = (V, A)$ and a node set $Z \subseteq V$, let $D[Z]$ be the digraph obtained from D by deleting the nodes of $V - Z$ (and all the arcs incident with them). If $B \subseteq A$ is a subset of the arc set, then we will identify B and the subgraph (V, B). Thus $B[Z]$ is obtained from (V, B) by deleting the nodes of $V - Z$ (and the arcs of B incident with them). The set of arcs of D entering Z is denoted $\delta^{in}_D(Z)$, the number of these arcs is $\varrho_D(Z) = |\delta^{in}_D(Z)|$. An **arc-weighted digraph** is a triple $D_w = (V, A, w)$ where (V, A) is a digraph and $w : A \to \mathbb{R}_+$ is a weight function. For an arc-weighted digraph $D_w = (V, A, w)$ and subset X of its node set we let $\varrho_{D_w}(X) = \sum\{w_a : a \text{ enters } X\}$ denote the weighted indegree. A **subpartition** of a subset X of V is a collection of pairwise disjoint non-empty subsets of X: note that $\emptyset$ cannot be a member of a subpartition, but $\emptyset$ is a valid subpartition, having no members at all. For a vector $x : A \to \mathbb{R}$ and subset $E \subseteq A$ we let $x(E) = \sum_{a \in E} x_a$.

3 Equivalence of versions

Theorem 1. *The following problems are polynomially equivalent (where $D_i = (V_i, A_i)$ is a digraph, k a positive integer, $c_i : A_i \to \mathbb{R}$ and $w_i : A_i \to \mathbb{R}_+$ for $i = 1, 2$).*

1. ***Blocking optimal k-union-arborescences (Problem 1)**: Given D_1, k, c_1, w_1, find $H \subseteq A_1$ so that H intersects every minimum c_1-cost k-union-arborescence in D_1, and $w_1(H)$ is minimum.*

2. ***Blocking optimal k-union-r-arborescences (Problem 2):*** *Given D_2, k, c_2, w_2 and a node $r \in V_2$, find $H \subseteq A_2$ so that H intersects every minimum c_2-cost k-union-r-arborescence in D_2, and $w_2(H)$ is minimum.*

Proof. Problem 2 reduces to Probem 1 by deleting all arcs entering node r from the input digraph. For the reverse reduction, consider an instance D_1, k, c_1, w_1 of the Problem 1, and define an instance of Problem 2 as follows. Let $V_2 = V_1 + r$ with a new node r, and let $D_2 = (V_2, A_1 \cup \{rv : v \in V_1\})$. Let the costs defined as $c_2(a) = c_1(a)$ for every $a \in A_1$ and $c_2(rv) = C$ for every new arc rv $(v \in V_1)$ where $C = \sum_{a \in A_1} c_1(a) + 1$. Finally, the weights are defined as follows: $w_2(a) = w_1(a)$ for every $a \in A_1$ and $w_2(rv) = W$ for every new arc rv $(v \in V_1)$ where $W = \sum_{a \in A_1} w_1(a) + 1$. By these choices, in the instance to Problem 2 given by D_2, k, c_2, w_2 and r, the minimum c_2-cost k-union-r-arborescences naturally correspond to minimum c_1-cost k-arborescences in D_1 (since they will use exactly k arcs leaving r), and for blocking these we will not use the new arcs because of their large weight. □

4 General observations on the uniform cost case

In this section we consider Problem 1 and Problem 2 in the case when c is uniform. For sake of clarity, we explicitly restate these problems.

Problem 3 (**Blocking k-union-arborescences**). *Given a digraph $D = (V, A)$, a positive integer k, and a nonnegative weight function $w : A \to \mathbb{R}_+$, find a subset H of the arc set such that H intersects every k-union-arborescence, and $w(H)$ is minimum.*

Problem 4 (**Blocking k-union-r-arborescences**). *Given a digraph $D = (V, A)$, a node $r \in V$, a positive integer k, and a nonnegative weight function $w : A \to \mathbb{R}_+$, find a subset H of the arc set such that H intersects every k-union-r-arborescence, and $w(H)$ is minimum.*

Note that the reduction given in Section 3 shows that Problem 4 is polynomial time reducible to Problem 3. On the other hand we did not find a reduction in the other direction: Problem 4 seems to be easier than Problem 3.

We first recall some definitions and results to be used later. First we recall the fundamental result of Edmonds' characterizing the existence of a k-union-r-arborescence in a digraph.

Theorem 2 (Edmonds' Disjoint Arborescence Theorem, [3]). *Given a digraph $D = (V, A)$, a node $r \in V$ and a positive integer k, there exists a k-union-r-arborescence in D if and only if $\varrho_D(X) \geq k$ for every non-empty $X \subseteq V - r$.*

On the other hand the existence of a k-union-r-arborescence in a digraph is characterized by the following result due to Frank.

Theorem 3 (Frank, [6, 4]). *Given a digraph $D = (V, A)$ and a positive integer k, there exists a k-union-arborescence in D if and only if $\sum_{X \in \mathcal{X}} \varrho_D(X) \geq k(|\mathcal{X}| - 1)$ for every subpartition $\mathcal{X}$ of V.*

Note that the condition in the theorem above always (trivially) holds for subpartitions having only one member, thus we may only need to check for subpartitions having at least two members. Furthermore, we may also narrow down to subpartitions in which every member is an insolid set of the digraph - the notion of insolid sets is defined as follows.

Definition 1. *Given a digraph $D = (V, A)$, a non-empty subset of nodes $X \subseteq V$ is called* in-solid, *if $\varrho(Y) > \varrho(X)$ holds for every nonempty $Y \subsetneq X$.*

A simple but useful corollary of the definition is the following fact.

Claim 1. *Given a digraph $D = (V, A)$ and an arbitrary non-empty subset $X \subseteq V$, there exists an in-solid set $X' \subseteq X$ such that $\varrho_D(X') \leq \varrho_D(X)$.*

Note that the insolid set X' in this claim can be found in polynomial time (given D and X), but we will not rely on this fact here. Claim 1 implies that Frank's theorem can be formulated with insolid sets. We say that a subpartition $\mathcal{X}$ is an **insolid subpartition** if every member of $\mathcal{X}$ is insolid.

Theorem 4 (Equivalent form of Theorem 3)**.** *Given a digraph $D = (V, A)$ and a positive integer k, there exists a k-union-arborescence in D if and only if $\sum_{X\in\mathcal{X}} \varrho_D(X) \geq k(|\mathcal{X}| - 1)$ for every insolid subpartition $\mathcal{X}$ of V with $|\mathcal{X}| \geq 2$.*

5 The cardinality case

In this section we solve Problem 3 in the case when $w \equiv 1$ is uniform, too. Note that the solution of Problem 4 when $w \equiv 1$ is easy: by Edmonds' Disjoint Arborescence Theorem the task is to remove all but $k - 1$ arcs of a minimum cut.

For sake of clarity we state the problem explicitly.

Problem 5. *Given a digraph $D = (V, A)$ and a positive integer k, find a minimum size subset H of the arc set such that $D - H$ does not contain a k-union-arborescence.*

By Theorem 3, our Problem 5 reduces to finding a smallest cardinality subset of arcs which, when removed, will create a violating subpartition. A subpartition $\mathcal{X}$ becomes a violating subpartition if we remove at least $\sum_{X\in\mathcal{X}} \varrho_D(X) - k(|\mathcal{X}| - 1) + 1$ arcs from $\cup\{\delta^{in}(X) : X \in \mathcal{X}\}$. Note that this is only possible if $\sum_{X\in\mathcal{X}} \varrho_D(X) - k(|\mathcal{X}| - 1) + 1 \leq \sum_{X\in\mathcal{X}} \varrho_D(X)$, which is equivalent to $|\mathcal{X}| \geq 2$. Furthermore, by Theorem 4, we can narrow down to insolid subpartitions. Therefore, Problem 5 is equivalent to the following problem.

Problem 6. *Given a digraph $D = (V, A)$ and a positive integer k, find an insolid subpartition $\mathcal{X}$ with $|\mathcal{X}| \geq 2$ maximizing $\sum_{X\in\mathcal{X}}(k - \varrho_D(X))$.*

We solve this problem in two steps: first we show how to solve it without the requirement on the size of the subpartition, and then we show how to force that requirement. Note that a subpartition $\mathcal{X}$ maximizing $\sum_{X\in\mathcal{X}}(k - \varrho_D(X))$ does not automatically satisfy the requirement $|\mathcal{X}| \geq 2$: for example in a k-arc-connected digraph $\mathcal{X} = \{V\}$ is an optimal subpartition. Note that the problem without the size requirement is a maximum weight matching problem in the hypergraph of insolid sets, but with the special weight function $k - \varrho$.

5.1 Finding an optimal subpartition of unconstrained size

In this section we solve the variant of Problem 6 that comes without the requirement on the size of the subpartition. In fact we will need the solution of a little more general problem, namely the following one.

Problem 7. *Given a digraph $D = (V, A)$, a nonempty subset $V' \subseteq V$ and a positive integer k, find a subpartition $\mathcal{X}$ of V' maximizing $\sum_{X \in \mathcal{X}}(k - \varrho_D(X))$.*

Note that we could restrict ourselves to insolid subpartitions in the problem. The following simple observation will be useful later.

Claim 2. *Given an instance to Problem 7, the subpartition $\mathcal{X} = \emptyset$ is an optimal solution if and only if $\varrho(X) \geq k$ for every nonempty $X \subseteq V'$.*

For an arbitrary digraph $D = (V, A)$, nonempty subset $V' \subseteq V$ and positive integer k, let $BestSubpart(V')$ denote an optimum solution of our unconstrained Problem 7 (that is, a maximizer of $\max\{\sum_{X \in \mathcal{X}}(k - \varrho(X)) : \mathcal{X}$ is a subpartition of $V'\}$). Note that $BestSubpart(V)$ always has at least one member (since the subpartition $\{V\}$ is better than the empty subpartition).

Fortunately, the family of insolid sets has a nice structure: as observed in [1], the family of insolid sets forms a subtree hypergraph, defined as follows. (Actually, the authors of [1] proved that the family of solid sets - the family of all insolid or outsolid sets - forms a subtree hypergraph, and the subtree representation can be found in polynomial time. Here we only need the property for the family of insolid sets.)

Definition 2. *A hypergraph $H = (V, \mathcal{E})$ is called a* ***subtree hypergraph*** *if there exists a tree T spanning the node set V such that every hyperedge in $\mathcal{E}$ induces a subtree of T. The tree T is called a* **basic tree** *(or representative tree) for the hypergraph H.*

Bárász, Becker and Frank proved that a representative tree for insolid sets exists, and can be found in polynomial time. The subtree hypergraph property and its polynomial time construction is quite surprising, given the fact that the number of insolid sets might be exponential.

Theorem 5 (Bárász, Becker, Frank [1])**.** *The family $\mathcal{F}_{in} = \mathcal{F}_{in}(D)$ of in-solid sets of a digraph $D = (V, A)$ is a subtree hypergraph. The representative tree can be found in polynomial time in the size of the digraph.*

For a digraph D, let $T_{in} = T_{in}(D)$ denote a representative tree for the family $\mathcal{F}_{in}$ of insolid sets. In fact we will only need to solve Problem 7 for a set V' that induces a subtree of T_{in}. Observe however that this is not a restriction: if $T_{in}[V']$ is not connected then solving Problem 7 for the components of $T_{in}[V']$ and taking the union of the obtained subpartitions gives a solution for V'. Therefore we may assume in Problem 7 that V' induces a subtree of T_{in}.

Unfortunately, enumerating all the insolid sets is not possible, because there can be exponentially many of them (an example can be found in [8]). However, in the case of insolid sets, the digraph itself provides a succint representation, thus we can query $\mathcal{F}_{in}$ through certain oracles using the likes of minimum cut algorithms.

We cite the following theorem that characterizes the optimum in Problem 7 for an arbitrary subtree hypergraph. (Note that the notion of subtree hypergraphs is very general, and thus this min-max is outside of the realm of efficient algorithms, because a subtree hypergraph may not be given as part of the input. The theorem holds anyway, and we will apply in a way that it also implies a polynomial running time.) For sake of completeness, we also provide a proof and algorithm here.

Theorem 6 (Frank [5]). *If $\mathcal{E} \subseteq 2^{V'}$ is a subtree hypergraph and $val : \mathcal{E} \to \mathbb{R}_+$ is a weight function then*

$$\max\{\sum_{e\in\mathcal{E}'} val_e : \mathcal{E}' \text{ is a collection of disjoint members of } \mathcal{E}\} = \tag{1}$$

$$\min\{\sum_{v\in V'} y_v : y : V' \to \mathbb{R}, y \geq 0, \sum_{v\in e} y_v \geq val_e \text{ for every } e \in \mathcal{E}\}. \tag{2}$$

Proof. We give an algorithmic proof of this theorem. Consider the following LP problem.

$$\max \sum_{e\in\mathcal{E}} val_e x_e \tag{3}$$

$$x : \mathcal{E} \to \mathbb{R}, x \geq 0, \tag{4}$$

$$\sum_{e:v\in e} x_e \leq 1 \text{ for every } v \in V'. \tag{5}$$

This LP is a relaxation of the maximum weight matching problem given in (1), while its LP dual is the minimization problem (2). Weak duality shows that the maximum in the theorem cannot be larger than the minimum. For the proof we need to show that the primal LP has an integer optimum solution for every weight function val. This is shown by the following dynamic programming algorithm. A sketch of the below algorithm goes as follows: Initially, we set node weights y_v to be very large. We specify a node r to be the root of the representative tree T, and start scanning the tree taking the leaves first. When a node v is scanned, its weight y_v is lowered as much as possible to maintain dual feasibility. When all nodes are scanned, $\mathcal{E}'$ is set up by tight sets, starting with one covering the node with positive weight closest to the root node, and recursively, adding tight sets for the remaining components. (A set $e \in \mathcal{E}$ is said to be **tight** with respect to the given feasible dual solution y if $\sum_{v:v\in e} y_v = val_e$.) The algorithm is detailed below.

Algorithm MaxWeightMatching($V', T, \mathcal{E}, val$)

begin

INPUT: A tree T on node set V', a family $\mathcal{E} \subseteq 2^{V'}$ of subtrees of T, and a weight function $val : \mathcal{E} \to \mathbb{R}$ ($\mathcal{E}$ and val are available via certain oracles: see later).

OUTPUT: A collection $\mathcal{E}'$ of disjoint members of $\mathcal{E}$ and a dual solution y of the LP Problem (3)-(5) so that $\sum_{e\in\mathcal{E}'} val_e = \sum_{v\in V'} y_v$.

1.1. Find some value M with $M > \max_{e\in\mathcal{E}} val_e$.

1.2. Initialize $y_v = M$ for every $v \in V'$.

1.3. Fix an arbitrary node $r \in V'$ and orient T out of r to get $\vec{T}$.

1.4. Let $r = v_1, v_2, \ldots, v_n$ be an order of the nodes of V so that $v_iv_j \in \vec{T}$ implies that $i < j$. Let V_i be the node set of the subtree of $\vec{T}$ rooted at v_i for every i.
1.5. For every $i = n, n-1, \ldots, 1$
1.6. Decrease y_{v_i} as much as possible so that y remains feasible for Problem (2): let e_i be a minimizer of $\min\{y(e) - val_e : v_i \in e \in \mathcal{E}, e \subseteq V_i\}$ and let $y_{v_i} = \max(val_{e_i} - y(e_i - v_i), 0)$ (note that e_i becomes tight, if y_{v_i} stays positive).
1.7. Let $i = 1$, $S = \emptyset$ and $\mathcal{E}' = \emptyset$. //Throughout $S \subseteq V$ and $\mathcal{E}' \subseteq \mathcal{E}$
1.8. For $i = 1, 2, \ldots, n$ do
1.9. If $y_{v_i} > 0$ and $v_i \notin S$ then
1.10. Let $\mathcal{E}' := \mathcal{E}' + \{e_i\}$ and $S := S \cup e_i$
1.11. Output $\mathcal{E}'$ and y.
end

The algorithm clearly proves the minmax theorem. In order to implement it in polynomial time in $n = |V'|$ we have to provide the subroutines needed in Steps 1.1 and 1.6. □

Oracles for insolid sets Next, we analyze this algorithm for the solution of Problem 7 to show that it can be implemented in polynomial time when $\mathcal{E}$ is the family of insolid sets and $val_e = k - \varrho_D(e)$ (for this weight function $y(e) \geq val_e \ \forall e \in \mathcal{E}$ implies $y(Z) \geq k - \varrho_D(Z)$ for every non-empty $Z \subseteq V$). For that, we need to establish subroutines for Steps 1.1 and 1.6 that have running time polynomial in the size of the graph.

It is easy to realize Step 1.1 of Algorithm MaxWeightMatching: $M = k + 1$ will be a good choice.

Step 1.6 can be realized with minimum cut computation as follows. Let $V_i = V(T_{v_i})$ be the node set of the subtree of T rooted at v_i and define a digraph $D' = (V + s, A')$ as follows: add a new node s to D, introduce an arc sv from s to every $v \in V_i$ of multiplicity y_v (note that y stays integer during the algorithm). Then by a minimum cut algorithm we find an inclusionwise minimal minimizer Z of $\min\{\varrho_{D'}(Z) : v_i \in Z \subseteq V_i\}$. We claim that if $y(Z - v_i) < k - \varrho(Z)$ (that is, $y(v_i)$ will not decrease to 0) then Z is in-solid in D. By Claim 1, there is an insolid (D-insolid, to be precise) set $Z' \subseteq Z$ such that $\varrho_D(Z') \leq \varrho_D(Z)$. If $v_i \in Z'$ then $Z' = Z$ by our choice of Z. If $v_i \notin Z'$, then $y(Z') + \varrho_D(Z') - k \leq y(Z - v_i) + \varrho_D(Z) - k < 0$, contradicting with $y(Z') \geq k - \varrho_D(Z')$. In other words, we in fact maintain $y(Z) \geq k - \varrho_D(Z)$ for every non-empty $Z \subseteq V'$ (besides non-negativity of y), and in Step 1.6 the set $Z = e_i$ we find is insolid only if $y(v_i)$ does not decrease to zero.

This finally proves that Problem 7 can be solved in polynomial time.

5.2 Enforcing the size requirement

For an arbitrary digraph $D = (V, A)$, nonempty subset $V' \subseteq V$ and positive integer k, recall that $BestSubpart(V')$ denotes an optimum solution of our unconstrained Problem 7 (that is, a maximizer of $\max\{\sum_{X \in \mathcal{X}}(k - \varrho(X)) : \mathcal{X}$ is a subpartition

of V'}). This can be found in polynomial time by the previous section: for a set V' that induces a subtree of T_{in} we have $BestSubpart(V')$ directly in the output of $MaxWeightMatching(V', T_{in}[V'], \mathcal{F}_{in}[V'], k - \varrho_D)$, where $\mathcal{F}_{in}[V']$ denotes the family of insolid sets contained in V'.

Algorithm $BestConstrSubpart(D, k)$
begin
INPUT: A digraph $D = (V, A)$ and a positive integer k.
OUTPUT: A maximizer of $\max\{\sum_{X\in\mathcal{X}}(k - \varrho(X)) : |\mathcal{X}| \geq 2, \mathcal{X}$ is a subpartition of $V\}$
1.1. If the subpartition $BestSubpart(V)$ has at least 2 members then output this and STOP.
1.2. Let $T = T_{in}$ be a basic tree of D.
1.3. For every edge e of T consider at most 3 candidates defined as follows.
1.4. Let V_1 and V_2 be the node sets of the 2 components of $T - e$.
1.5. For $i = 1, 2$, let X_i be an inclusionwise minimal minimizer of $\min\{\varrho_D(X) : \emptyset \neq X \subseteq V_i\}$ (which can be found with minimum cut computations).
1.6. For both $i = 1, 2$, let $\mathcal{P}_i = BestSubpart(V_i)$.
1.7. Candidate 0 is $\{X_1, X_2\}$.
1.8. If $|\mathcal{P}_1| \geq 1$ then Candidate 1 is $\mathcal{P}_1 \cup \{X_2\}$.
1.9. If $|\mathcal{P}_2| \geq 1$ then Candidate 2 is $\mathcal{P}_2 \cup \{X_1\}$.
1.10. Output the best of the candidates above.
end

Theorem 7. *Algorithm BestConstrSubpart is correct.*

Proof. Clearly, the Algorithm outputs a feasible solution (that is, a subpartition with at least 2 members), since all its candidates are feasible. Let $\mathcal{P}$ be an optimal insolid subpartition. The proof is complete if we show that one of the candidates is at least as good as $\mathcal{P}$. Choose an edge e of T_{in} so that one of the components (call it V_1) of $T - e$ contains only one member of $\mathcal{P}$. If $\mathcal{P}$ has only 2 members then clearly Candidate 0 for the edge e is not worse than $\mathcal{P}$, so we can assume that $|\mathcal{P}| \geq 3$. Let V_2 be the other component of $T - e$. If $BestSubpart(V_2)$ has at least 1 member then Candidate 2 is not worse than $\mathcal{P}$. But if $BestSubpart(V_2)$ has no members then Claim 2 gives that $\varrho_{D_w}(X) \geq k$ for every nonempty $X \subseteq V_2$, therefore Candidate 0 should not be worse than $\mathcal{P}$, finishing the proof. □

The following example shows that even if $BestSubpart(V)$ is not feasible (that is, it has only 1 member), $BestConstrSubpart(V)$ might have more than 2 members: let $k = 4$ and the digraph be directed circuit of size 3, where every arc has multiplicity 3.

5.3 Analysis

In the algorithm designed above to solve Problem 6, we apply two steps. In the first step (Step 1.2) we determine a basic tree representation of the in-solid sets, which according to [1] can be done in $n^3S(n, m)$ time, where n is the number of nodes and

m is the number of edges in D, and $S(n,m)$ denotes the time complexity of finding a minimum $s-t$-cut in a digraph with n nodes and m arcs. In the second step (Step 1.6), we apply our algorithm for Problem 7 for $O(n)$ different subsets U of V, which are determined by removing an edge from the basic tree. For any given subset U, Problem 7 is solved in time $nS(n+1,3m)$, since we apply a minimum cut algorithm to determine the value of $y(v)$ for all nodes v in U by a minimum cut computation for graphs of at most $n+1$ nodes and at most $m+kn$ arcs. (To see this, note that only one node s is added to the graph, and the number of arcs added is equal to the sum of $y(v)$'s, which is equal to the sum $\sum_{X\in\pi}(k-\varrho(X))$ over a partition π of $U+s$.) Note that $kn \leq 2m$, since otherwise there may be no k disjoint arborescences. The total running time for the algorithm amounts to $n^2S(n+1,3m)$. Thus, by using Orlin's algorithm [9] for minimum cut computations, the running time is bounded by $O(n^4m)$ (the bottleneck being Step 1.2).

6 The weighted case

In this section we solve Problems 3 and 4. Our algorithms are only polynomial if k is fixed (not part of the input).

6.1 Weighted blocking of k-union-r-arborescences

First we give an algorithm solving Problem 4. By Edmonds' disjoint arborescence theorem, the optimum solution will be all but $k-1$ arcs entering a nonempty subset $X \subseteq V-r$. Thus what we do is that we guess which $k-1$ arcs remain.

Algorithm BLOCKING-k-UNION-r-ARBORESCENCES
begin
INPUT: A digraph $D=(V,A)$, a weight function $w: A \to \mathbb{R}_+$, a node $r \in V$, and a positive integer k. (We assume that $\varrho(r)=0$.)
OUTPUT: A subset H of the arcs so that there is no k-union-r-arborescence in $D-H$ and $w(H)$ is minimum.
1.1. Let $best=\infty$.
1.2. For any subset E of A with $|E|=k-1$ do
1.3. Find a minimizer X_0 of $\min\{\varrho_{D_w-E}(X) : \emptyset \neq X \subseteq V-r\}$.
1.4. If $\varrho_{D_w-E}(X_0) < best$ then let $best=\varrho_{D_w-E}(X_0)$ and $H=\delta^{in}_{D-E}(X_0)$
1.5. Output H.
end

This algorithm runs in time $O(m^k HO(n,m))$ time where $HO(n,m)$ denotes the time complexity of determining $\min\{\varrho_{D_w}(X) : \emptyset \neq X \subseteq V-r\}$ in an arc-weighted digraph D_w with n nodes and m arcs. Using the algorithm of Hao and Orlin [7] we have $HO(n,m)=O(nm\log(n^2/m))$, giving that Algorithm BLOCKING-k-UNION-r-ARBORESCENCES has running time $O(m^k nm\log(n^2/m))$.

6.2 Weighted blocking of k-union-arborescences

Now we turn to Problem 3.

Claim 3. *Given a digraph $D = (V, A)$ that contains a k-union-arborescence, let $H \subseteq A$ such that $D - H$ does not contain a k-union-arborescence, and H is inclusionwise minimal to this property. Then there exists a subpartition $\mathcal{X}$ of V such that $2 \leq |\mathcal{X}| \leq k+1$ and $\sum_{X\in\mathcal{X}} \varrho_{D-H}(X) = k(|\mathcal{X}|-1)-1$.*

Proof. By Theorem 3, there exists a subpartition $\mathcal{X}$ of V such that $\sum_{X\in\mathcal{X}} \varrho_{D-H}(X) < k(|\mathcal{X}|-1)$: choose such an $\mathcal{X}$ with $|\mathcal{X}|$ smallest possible. By this minimal choice of $\mathcal{X}$, $\varrho_{D-H}(X) < k$ for every $X \in \mathcal{X}$ (if $\varrho_{D-H}(X) \geq k$ for some $X \in \mathcal{X}$ then $\mathcal{X} - \{X\}$ would just as well do). If $|\mathcal{X}| > k+1$ then let $\mathcal{X}' \subseteq \mathcal{X}$ be arbitrary with $|\mathcal{X}'| = k+1$ and observe that $\sum_{X\in\mathcal{X}'} \varrho_{D-H}(X) \leq (k+1)(k-1) < k^2$, contradicting the minimal choice of $\mathcal{X}$. Therefore $|\mathcal{X}| \leq k+1$ indeed holds ($|\mathcal{X}| \geq 2$ is straightforward). By the minimality of H, $H \subseteq \cup_{X\in\mathcal{X}} \delta^{in}_D(X)$ and $\sum_{X\in\mathcal{X}} \varrho_{D-H}(X) = k(|\mathcal{X}|-1)-1$. □

By Claim 3, the optimum solution of Problem 3 will be all but $k(|\mathcal{X}|-1)-1$ arcs from a set $\cup_{X\in\mathcal{X}} \delta^{in}(X)$ for some subpartition $\mathcal{X}$ with $2 \leq |\mathcal{X}| \leq k+1$. Our algorithm below guesses this set of remaining arcs. As a subroutine we need an algorithm solving the following problem.

Problem 8. *Given a digraph $D = (V, A)$, a weight function $w : A \to \mathbb{R}_+$ and a positive integer t, determine $\min\{\sum_{X\in\mathcal{X}} \varrho_{D_w}(X) : \mathcal{X}$ is a subpartition of V and $|\mathcal{X}| = t\}$.*

This problem will be solved in Section 6.2.1.

Algorithm BLOCKING-k-UNION-ARBORESCENCES
begin
INPUT: A digraph $D = (V, A)$, a weight function $w : A \to \mathbb{R}_+$, and a positive integer k.
OUTPUT: $\min\{w(H)$: there is no k-union-arborescence in $D - H\}$.
1.1. If there is no k-union-arborescence in D then output 0 and STOP.
1.2. Let $best = \infty$.
1.3. For $t = 2, 3, \ldots, k+1$ do
1.4. For every $E \subseteq A$ of size $k(t-1)-1$ do
1.5. Let $candidate = \min\{\sum_{X\in\mathcal{X}} \varrho_{D_w-E}(X) : \mathcal{X}$ is a subpartition of V and $|\mathcal{X}| = t\}$.
1.6. If $candidate < best$ then $best = candidate$.
1.7. Output $best$.
end

For sake of simplicity we formulated Algorithm BLOCKING-k-UNION-ARBORESCENCES so that it outputs the weight of the optimal arc set that blocks all k-union-arborescences: the algorithm can be obviously modified to return the optimal arc set instead. The running time of Algorithm BLOCKING-k-UNION-ARBORESCENCES will be analyzed in Section 6.2.2. The proof of correctness is as follows. Let alg be the

output of Algorithm BLOCKING-k-UNION-ARBORESCENCES and $opt = w(OPT)$ be the optimum solution. Clearly, $opt \leq alg$. On the other hand, by Claim 3, there exists a subpartition $\mathcal{X}$ such that $\sum_{X\in\mathcal{X}} \varrho_{D-OPT}(X) = k(|\mathcal{X}|-1)-1$ and $2 \leq |\mathcal{X}| \leq k+1$; for $E = \cup_{X\in\mathcal{X}} \delta^{in}_{D-OPT}(X)$ the algorithm will find a candidate that is not worse than opt.

6.2.1 Solution of Problem 8

In this section we solve Problem 8 that is used in Step 1.5 of Algorithm BLOCKING-k-UNION-ARBORESCENCES. Clearly, we can assume that the optimal soltution is an insolid subpartition. Note the difference between this problem and Problem 6: here we have exact restriction on the size of the subpartition to be found, not just a lower bound.

Algorithm BEST-FIXED-SUBPART
begin
INPUT: A digraph $D = (V, A)$, a weight function $w : A \to \mathbb{R}_+$, and a positive integer t.
OUTPUT: $\min\{\sum_{X\in\mathcal{X}} \varrho_{D_w}(X) : \mathcal{X}$ is a subpartition of V and $|\mathcal{X}| = t\}$.
1.1. Let T be a representative tree for the insolid sets of the weighted digraph D_w.
1.2. Let $best = \infty$
1.3. For every $F \subseteq E(T)$ of size $t-1$ do
1.4. Let $Z_1, Z_2, \ldots, Z_t$ be the node sets of the connected components of $T - F$.
1.5. Let $candidate = 0$
1.6. For $i = 1, 2 \ldots, t$ do
1.7. $candidate \mathrel{+}= \min\{\varrho_{D_w}(X) : \emptyset \neq X \subseteq Z_i\}$.
1.8. If $candidate < best$ then $best = candidate$.
1.9. Output $best$.
end

The proof of correctness of Algorithm BEST-FIXED-SUBPART is left to the reader.

6.2.2 Running time

In this section we analyze the running time of Algorithm BLOCKING-k-UNION-ARBORESCENCES. Let n and m denote the number of nodes and arcs of the input digraph D. Recall that $S(n,m)$ denotes the time complexity of finding a minimum $s-t$-cut in a weighted digraph with n nodes and m arcs, and similarly $HO(n,m)$ denotes the time complexity of determining $\min\{\varrho_{D_w}(X) : \emptyset \neq X \subseteq V'\}$, where $V' \subseteq V$. The running time of Algorithm BEST-FIXED-SUBPART is $n^{t-1}HO(n,m)$ plus the time needed to determine the representative tree T, which can be done in time $O(n^3 S(n,m))$, as mentioned in Section 5.3. Thus we get $O(m^{k^2}(n^k HO(n,m) + n^3 S(n,m)))$ as an overall complexity for Algorithm BLOCKING-k-UNION-ARBORESCENCES. Substituting $S(n,m) = O(nm)$ ([9]) and $HO(n,m) = O(nm\log(n^2/m))$ ([7]) we get $O(m^{k^2}(n^{k+1}m\log(n^2/m) + n^4 m))$ for the running time.

7 Acknowledgements

The authors wish to thank Kristóf Bérczi and Tamás Király for observing the flaw in a previous version and for useful discussions on the topic.

References

[1] Mihály Bárász, Johanna Becker, and András Frank, *An algorithm for source location in directed graphs*, Oper. Res. Lett. **33** (2005), no. 3, 221–230.

[2] Attila Bernáth and Gyula Pap, *Blocking optimal arborescences*, Integer Programming and Combinatorial Optimization, Springer, 2013, pp. 74–85.

[3] Jack Edmonds, *Edge-disjoint branchings*, Combinatorial algorithms **9** (1973), 91–96.

[4] A Frank, *On disjoint trees and arborescences*, Algebraic Methods in Graph Theory, Colloquia Mathematica Soc. J. Bolyai, vol. 25, 1978, pp. 159–169.

[5] András Frank, *Some polynomial algorithms for certain graphs and hypergraphs*, Proceedings of the Fifth British Combinatorial Conference, 1975, pp. 211–226.

[6] Andras Frank, *Covering branchings*, Acta Scientiarum Mathematicarum (Szeged) **41** (1979), no. 77-81, 3.

[7] Jianxiu Hao and James B. Orlin, *A faster algorithm for finding the minimum cut in a directed graph*, JOURNAL OF ALGORITHMS **17** (1994), 424–446.

[8] Hiro Ito, Kazuhisa Makino, Kouji Arata, Shoji Honami, Yuichiro Itatsu, and Satoru Fujishige, *Source location problem with flow requirements in directed networks*, Optimization Methods and Software **18** (2003), no. 4, 427–435.

[9] James B. Orlin, *Max flows in O(nm) time, or better*, Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '13, ACM, 2013, pp. 765–774.

[10] Alexander Schrijver, *Combinatorial optimization: polyhedra and efficiency*, vol. 24, Springer Verlag, 2003.