The complexity of finding arc-disjoint branching flows

Jørgen Bang-Jensen, Frédéric Havet, Anders Yeo

▶ **To cite this version:**

Jørgen Bang-Jensen, Frédéric Havet, Anders Yeo. The complexity of finding arc-disjoint branching flows. Discrete Applied Mathematics, 2016, 209, pp.16-26. 10.1016/j.dam.2015.10.012 . hal-01360910

HAL Id: hal-01360910
https://hal.inria.fr/hal-01360910

Submitted on 6 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The complexity of finding arc-disjoint branching flows

J. Bang-Jensen* Frédéric Havet† Anders Yeo‡

September 2, 2014

Abstract

The concept of arc-disjoint flows in networks was recently introduced in [2]. This is a very general framework within which many well-known and important problems can be formulated. In particular, the existence of arc-disjoint branching flows, that is, flows which send one unit of flow from a given source s to all other vertices, generalizes the concept of arc-disjoint out-branchings (spanning out-trees) in a digraph. A pair of out-branchings $B^+_{s,1}, B^+_{s,2}$ from a root s in a digraph $D = (V, A)$ on n vertices corresponds to arc-disjoint branching flows x_1, x_2 (the arcs carrying flow in x_i are those used in $B^+_{s,i}$, $i = 1, 2$) in the network that we obtain from D by giving all arcs capacity $n - 1$. It is then a natural question to ask how much we can lower the capacities on the arcs and still have, say, two arc-disjoint branching flows from the given root s. We prove that for every fixed integer $k \geq 2$ it is

- an NP-complete problem to decide whether a network $\mathcal{N} = (V, A, u)$ where $u_{ij} = k$ for every arc ij has two arc-disjoint branching flows rooted at s.
- a polynomial problem to decide whether a network $\mathcal{N} = (V, A, u)$ on n vertices and $u_{ij} = n - k$ for every arc ij has two arc-disjoint branching flows rooted at s.

The algorithm for the later result generalizes the polynomial algorithm, due to Lovász, for deciding whether a given input digraph has two arc-disjoint out-branchings rooted at a given vertex. Finally we prove that under the so-called Exponential Time Hypothesis (ETH), for every $\epsilon > 0$ and for every $k(n)$ with $(\log(n))^{1+\epsilon} \leq k(n) \leq \frac{n}{2}$ (and for every large i we have $k(n) = i$ for some n) there is no polynomial algorithm for deciding whether a given digraph contains two arc-disjoint branching flows from the same root so that no arc carries flow larger than $n - k(n)$.

Keywords: disjoint branchings, branching flow, polynomial algorithm, NP-complete.

1 Introduction

Notation follows [3]. We denote the vertex set and arc set of a digraph D by $V(D)$ and $A(D)$, respectively and write $D = (V, A)$ where $V = V(D)$ and $A = A(D)$. Unless otherwise specified, the numbers n and m will always be used to denote the number of vertices, repectively arcs in the digraph in question. The digraphs may have parallel arcs but no loops. Paths and cycles are always directed unless otherwise specified. We will use the notation $[k]$ for the set of integers $\{1, 2, \ldots, k\}$.

An (s, t)**-path** in a digraph D is a directed path from the vertex s to the vertex t. The **underlying graph** of a digraph D, denoted $UG(D)$, is obtained from D by suppressing the orientation of each arc. A digraph D is **connected** if $UG(D)$ is a connected graph. When xy is an arc of D we say that x **dominates** y. For a digraph $D = (V, A)$ the **out-degree**, $d^+_D(x)$ (resp. the **in-degree**, $d^-_D(x)$) of

*Department of Mathematics and Computer Science, University of Southern Denmark, Odense DK-4230, Denmark (email:jbj@imada.sdu.dk). Parts of this work was done while the first author attended the program "Graphs, hypergraphs and computing" at Institute Mittag-Leffler, spring 2014. The research of Bang-Jensen was also supported by the Danish research council under grant number 1323-00178B

†Project Coati, I3S (CNRS, UNSA) and INRIA, Sophia Antipolis, France (email: frederic.havet@inria.fr)

‡Engineering Systems and Design, Singapore University of Technology and Design, 20 Dover Drive, 138682 Singapore, Singapore and Department of Mathematics, University of Johannesburg, Auckland Park, 2006, South Africa (email andersyeo@gmail.com).The research of Yeo was supported by the Danish research council under grant number 1323-00178B

a vertex $x \in V$ is the number of arcs of the kind xy (resp. yx) in A, where we count parallel arcs. When $X \subseteq V$ we shall also write $d^+_X(v)$ to denote the number of arcs vx with $x \in X$.

An **out-tree** rooted at s, also called an **s-out-tree**, is a tree containing the vertex s in $UG(D)$ such that every vertex v different from s has exactly one arc entering in the tree. Equivalently, s has a unique directed path to every other vertex of the tree using only arcs of the tree. An **s-out-branching** is a spanning s-out-tree. We use the notation T^+_s, B^+_s to denote an s-out-tree, respectively an s-out-branching.

Branchings in digraphs are important from a practical point of view and appear in many applications and hence it is relevant to consider quality measures for branchings. This has been done in a number of papers, see e.g. [5, 8, 9, 13]. An s-out-branching B^+_s is k-**safe** if no matter which out-neighbour v of s in B^+_s we consider, the s-out-tree T^+_s that we will obtain after deleting all the vertices of the out-tree T^+_v rooted at v in B^+_s contains at least k vertices (s and at least $k-1$ other vertices). In applications (where an s-out-branching is used to route information or similar from the root s) it is desirable to use an out-branching which is k-safe for a high value of k, because this means that no matter which arc we cut, s can still reach k other vertices using only arcs from the remaining s-out-tree.

In terms of protection against arc-faults, branchings are not a very good way of sending information from one source to all other vertices: If we insist that the vertex sets of the routes that we use to send the information from s to all other vertices may only intersect in a prefix of each route (this is equivalent to saying that the union of the routes is an s-out-branching), then the set of routes may be very vulnerable to arc-deletions. As an example consider the digraph H consisting of vertices $s = v_1, v_2, \ldots, v_{2p+1}$ and arcs $\{sv_2, sv_3, v_2v_3\} \cup \{v_3v_i | i \geq 4\}$. This digraph contains no 3-safe s-out-branching. On the other hand, if instead we use each of the arcs sv_2, sv_3 on p of the paths from s to $V - s$, then deletion of one of the arcs sv_2, sv_3 disconnects s from only half of the other vertices. We may then ask what is the best way to route the information from s to all other vertices, while preserving a high degree of protection against arc-faults. This leads to the study of branching flows. Before we can formally define these, we need to recall a bit of flow theory.

A **network** $\mathcal{N} = (V, A, u)$ is a digraph $D = (V, A)$ equipped with a non-negative capacity function $u : A \rightarrow \mathbf{R}_0$ on its arcs. A **flow** in $\mathcal{N}$ is any non-negative function $x : A \rightarrow \mathbf{R}_0$ which satisfies that $x_{ij} \leq u_{ij}$ for every $ij \in A$, where x_{ij}, u_{ij} denote, respectively, the flow value on ij and the capacity of ij. The **balance-vector** of a flow x is the function b_x on V which to each vertex $i \in V$ associates the value $b_x(i) = \sum_{ij \in A} x_{ij} - \sum_{pi \in A} x_{pi}$.

If $\mathcal{N} = (V, A, u, b)$, that is, there is also a balance-vector specified for $\mathcal{N}$, then a flow x is **feasible** in $\mathcal{N}$ if it satisfies $b_x(v) = b(v)$ for all $v \in V$. Two flows x, y in a network $\mathcal{N}$ are **arc-disjoint** if $x_{ij} \cdot y_{ij} = 0$ for every arc ij of $\mathcal{N}$.

An important result in flow theory is the following which states that it is possible to decide in polynomial time whether or not there exists a feasible flow for a given network $\mathcal{N} = (V, A, u, b)$ (see e.g. [3, Section 4.8]).

Theorem 1.1 *For a given network $\mathcal{N} = (V, A, u, b)$ with arc-capacities given by u and vertex-balances prescribed by b one can check, using one max-flow calculation, whether there exists a feasible flow x in $\mathcal{N}$.*

A **path flow** along the path P (resp. **cycle flow** along the cycle C) in a network $\mathcal{N}$ is a flow x which has $x_{ij} = k$ for every arc on P (resp. C) for some positive value k and $x_{ij} = 0$ for all arcs not on P (resp. C). An **s-branching flow** in a network $\mathcal{N}$ is a flow x in $\mathcal{N}$ with balance vector $b_x(v) = -1$ for $v \neq s$ and $b_x(s) = n - 1$, where n denotes the number of vertices in $\mathcal{N}$.

The following folklore result (see e.g. [1, Section 3.5] or [3, Section 4.3.1]) is very useful when working with flows.

Theorem 1.2 (Flow Decomposition Theorem) *Every flow x in a network $\mathcal{N}$ on n vertices and m arcs is the arc-sum of at most $n+m$ path and cycle flows. Furthermore, the path flows can be taken along paths $P_1, \ldots, P_q$ such that P_i starts in a vertex s_i with $b_x(s_i) > 0$ and ends in a vertex t_i with*

$b_x(t_i) < 0$ *for* $i \in [q]$. *In particular, if* $b_x \equiv 0$ *there are no path flows in the decomposition and* x *is the arc-sum of at most* m *cycle flows. Given the flow* x*, a decomposition as above can be found in time* $O(nm)$.

Note that when we consider branching flows below, we are only interested in the acyclic part of such a flow, that is, the collection of paths from the root to all other vertices that we obtain by flow decomposition (we leave out flow along cycles since that does not contribute to the balance of the flow). By the Flow Decomposition Theorem, every branching flow x from s contains one or more out-branchings from s as a subdigraph (x sends one unit of flow from s to all other vertices).

As in the case of branchings, we may also measure the quality of an s-branching flow in terms of how vulnerable it is towards arc-deletions. If we have no restrictions on the flow values, a branching flow x may have flow equal to $r \leq n-1$ on some arc out of s, corresponding to that arc belonging to r of the path flows whose arc sum forms the branching flow x. This means that if we keep only the arcs used by x and one arc fails (is deleted) then s may be unable to reach as many as r vertices after deleting that arc. In particular, if $r = n-1$, one arc failure can disconnect s from all other vertices in the chosen solution. Thus, from a practical point of view (say, in an application where branching flows are used to route information), it could be useful to restrict the maximum flow value in an arc to as small as possible. If follows from Theorem 1.1 that for every k, there is a polynomial algorithm to check whether a network $\mathcal{N} = (V, A, u \equiv k)$ has an s-branching flow from a given vertex s. Furthermore, also the min-cost version of this can be solved in polynomial time using any polynomial algorithm for minimum cost flows. These observations should be compared to the following result.

Theorem 1.3 *[4] The problem of deciding whether a digraph on* n *vertices has an* $(n-k)$*-safe out-branching is polynomial time solvable if* $k \leq 2$ *and NP-complete for all* $k \geq 3$ *(*k *is not part of the input).*

Edmonds characterized digraphs with k arc-disjoint s-out branchings from a prescribed root s.

Theorem 1.4 (Edmonds' Branching Theorem) *[6] A digraph* $D = (V, A)$ *has* k *arc-disjoint* s*-out-branchings if and only if* D *has* k *arc-disjoint* (s, v)*-paths for every* $v \in V - s$.

From Edmonds' Branching Theorem and the algorithmic proof of this due to Lovász [10] (see also [3, Section 9.3]), we obtain the following characterization of networks in which all capacities are equal to $n-1$ that have k arc-disjoint branching flows:

Theorem 1.5 *A network* $\mathcal{N} = (V, A, u \equiv |V| - 1)$ *has* k *arc-disjoint* s*-branching flows* $x^1, x^2, \ldots, x^k$ *if and only if there are* k *arc-disjoint* (s, v)*-paths in* $\mathcal{N}$ *for every* $v \in V \setminus \{s\}$*. Furthermore, there is a polynomial algorithm for constructing such flows* $x^1, x^2, \ldots, x^k$ *when they exist.* $\diamond$

The purpose of this paper is to study the complexity of the problem of deciding the existence of arc-disjoint branching flows in networks whose arc-capacities are bounded. We first show that when all capacities are bounded by a constant k, the problem becomes NP-complete as soon as $k \geq 2$. The problem is trivial for $k = 1$ (see below).

In the second part of the paper we consider the case in which we still have uniform capacities, but now they are all $n-k$ for some constant k. The case $k = 1$ is the same as asking for a pair of arc-disjoint s-out-branchings and hence polynomial by our remarks above. For $k = 2$ the problem is still fairly simple: we just need a pair of arc-disjoint s-out-branchings both of which are 2-safe and this can be checked in polynomial time. For larger values of k the problem becomes more complicated, but we use structural observations about branching flows in networks with high capacities ($n-k$ for all arcs) to reduce the problem to that of checking the existence of a pair of arc-disjoint s-out-branchings for at least one digraph in a family of digraphs of polynomial size in n and k. Our method extends Lovász's algorithmic proof of Edmonds' Branching Theorem. The key idea is that, if the capacities are $n-k$ and there is a branching flow from s in $\mathcal{N}$ then there exists one in which only vertices that are close to s will receive flow on more than one arc.

Finally we prove that under the so-called Exponential Time Hypothesis (ETH), for every $\epsilon > 0$ and every $k(n)$ with $(\log(n))^{1+\epsilon} \leq k(n) \leq \frac{n}{2}$ (and for every large i we have $k(n) = i$ for some n) there is no polynomial algorithm for deciding whether a given digraph contains two arc-disjoint branching flows from the same root so that no arc carries flow larger than $n - k(n)$.

2 Arc-disjoint branching flows in bounded capacity networks

Clearly a unit capacity network has two arc-disjoint branching flows from a given root s if and only if s has at least two arcs to every other vertex. So the first interesting case is arc-disjoint branching flows in networks with maximum capacity 2. Here the complexity of the uniform capacity case was open but if we allow some capacities to be only one it was shown in [2] that the problem is already NP-complete (note that the proof in [2] does not extend to a proof for the uniform capacity case).

Theorem 2.1 *[2] It is NP-complete to decide whether a network $\mathcal{N} = (V, A, u)$, where $u_{ij} \in \{1, 2\}$ for all $ij \in A$, has two arc-disjoint s-branching flows.*

Our first goal is to use Theorem 2.1 to prove that it is NP-complete to decide the existence of arc-disjoint branching flows from the same root in networks where all the capacities equal some fixed number $k \geq 2$. This easily implies that it is also NP-complete to decide the existence of arc-disjoint branching flows from the same root in networks where all the capacities are at most k.

Theorem 2.2 *For every fixed integer $k \geq 2$, it is NP-complete to decide whether a network $\mathcal{N} = (V, A, u \equiv k)$ with special vertex s has two arc-disjoint s-branching flows.*

Proof: We will reduce from the case when the capacity of each arc is either 1 or 2, which is NP-hard by Theorem 2.1. So let $\mathcal{N}' = (V', A', u')$ be a network, where $u'_{ij} \in \{1, 2\}$ for all $ij \in A'$. We will create a new network $\mathcal{N} = (V, A, u \equiv k)$ as follows.

For each arc $ij \in A'$ we do the following. Add k new vertices, $y^1_{ij}, y^2_{ij}, \ldots, y^k_{ij}$ and add two parallel arcs from s to y^1_{ij} and from y^r_{ij} to y^{r+1}_{ij} for all $r = 1, 2, \ldots, k-1$. Finally remove the arc ij and add the arcs $iy^{u'_{ij}}_{ij}$ and $y^k_{ij}j$. Let $\mathcal{N}$ denote the network obtained after performing the above operation for all $ij \in A'$ and setting all capacities to k. Clearly $\mathcal{N}$ can be constructed in polynomial time given $\mathcal{N}' = (V', A', u')$. We will show that $\mathcal{N}'$ has two arc-disjoint s-branching flows if and only if $\mathcal{N}$ has two arc-disjoint s-branching, which would complete the proof by Theorem 2.1.

First assume that there are two arc-disjoint branching flows, x^1 and x^2, from s in $\mathcal{N}'$. For each $ij \in A'$ modify the flows as follows. Let $a^{r,1}_{ij}$ and $a^{r,2}_{ij}$ be the two parallel arcs from y^r_{ij} to y^{r+1}_{ij} for $r = 1, 2, \ldots, k-1$ and let $a^{0,1}_{ij}$ and $a^{0,2}_{ij}$ be the two parallel arcs from s to y^1_{ij}. Initially, for all $r = 0, 1, \ldots, k-1$, add the arc $a^{r,1}_{ij}$ to x^1 with flow $k - r$ and add the arc $a^{r,2}_{ij}$ to x^2 also with flow $k - r$. Also let the flow on $iy^{u'_{ij}}_{ij}$ and $y^k_{ij}j$ be zero in both branching flows. Now for each $q = 1, 2$, add an additional flow of x^q_{ij} to all the following arcs (which form a path).

$$iy^{u'_{ij}}_{ij}, a^{u'_{ij},q}_{ij}, a^{u'_{ij}+1,q}_{ij}, a^{u'_{ij}+2,q}_{ij}, \ldots, a^{k-1,q}_{ij}, y^k_{ij}j$$

As x^1 and x^2 are arc-disjoint flows we note that $x^1_{ij} = 0$ or $x^2_{ij} = 0$ (or both). Therefore we have only added flow to the arcs $iy^{u'_{ij}}_{ij}$ and $y^k_{ij}j$ in at most one of x^1 and x^2. This implies that even after the above operation the flows are arc-disjoint. Furthermore we note that no arc has received a flow of more than k, as if we add a flow of x^q_{ij} to an arc $a^{r,q}_{ij}$ then $r \geq u'_{ij}$, which implies that the flow in $a^{r,q}_{ij}$ is at most $k - r + x^q_{ij} \leq k - u'_{ij} + u'_{ij} \leq k$. Furthermore the resulting flows are branching flows as, in each of them, the flow entering any vertex (except s) is one more than the flow leaving the vertex. Therefore after performing the above operation for all $ij \in A'$ we have obtained the desired branching flows in $\mathcal{N}$.

Conversely assume that there are two arc-disjoint branching flows, x^1 and x^2, from s in $\mathcal{N}$. Without loss of generality assume that there is no directed cycle in x^1 or in x^2 (as we can easily remove such cycles by the Flow Decomposition Theorem). Consider some $ij \in A'$. We will first show that we may

assume that the flow from s to y^1_{ij} is k in both x^1 and in x^2 (each flow using a separate arc from s to y^1_{ij}). Assume that this is not the case and without loss of generality the flow from s to y^1_{ij} is less than k in x^1. As all parameters in the network are integer-valued we may assume that all flows are integer-valued. We must have at least one unit of flow on $iy_{ij}^{u'_{ij}}$ in x^1, as otherwise we cannot have a balance of -1 on all vertices $y^1_{ij}, y^2_{ij}, \ldots, y^k_{ij}$. Therefore there exists a path P from s to i in x^1 with a flow of at least 1 on all arcs of P. If P uses the arc from s to y^1_{ij} or the arc $y^k_{ij}j$, then adding the arc $iy_{ij}^{u'_{ij}}$ to P gives us a directed cycle, a contradiction. Now remove one unit of flow along all arcs of P and $iy_{ij}^{u'_{ij}}$ and add one unit of flow to the arc sy^1_{ij} and if $u'_{ij} = 2$ then also add one unit of flow to the arc $y^1_{ij}y^2_{ij}$. The resulting flow is still a branching flow and no capacities are violated, but we have increased the flow from s to y^1_{ij} by 1. Continue this process until the flow from s to y^1_{ij} is k in both x^1 and in x^2.

As the amount of flow entering the set $\{y^1_{ij}, y^2_{ij}, \ldots, y^k_{ij}\}$ has to be k larger than the amount of flow leaving the set in both x^1 and x^2, we note that in x^1 the flow on $iy_{ij}^{u'_{ij}}$ and on $y^k_{ij}j$ is the same. Analogously in x^2 the flow is the same and the flow is zero for either x^1 or x^2 (or both). Assume without loss of generality that x^2 has no flow on $iy_{ij}^{u'_{ij}}$ (and therefore also no flow on $y^k_{ij}j$). As the capacity on the arc $y_{ij}^{u'_{ij}} y_{ij}^{u'_{ij}+1}$ is k and there are $k-u'_{ij}$ vertices in $\{y_{ij}^{u'_{ij}+1}, y_{ij}^{u'_{ij}+2}, \ldots, y^k_{ij}\}$ we note that the arc $y^k_{ij}j$ cannot have flow more than $k-(k-u'_{ij}) = u'_{ij}$. So removing all vertices $y^1_{ij}, y^2_{ij}, \ldots, y^k_{ij}$ and all arcs touching these vertices, but adding back the arc ij with flow the same as we had in $iy_{ij}^{u'_{ij}}$ (and also in $y^k_{ij}j$) gives us two arc-disjoint branching flows where the flow on ij is bounded by u'_{ij}. Therefore doing the above process for all $ij \in A'$ gives us the desired branching flows in $\mathcal{N}'$. $\diamond$

3 Arc-disjoint k-safe branchings

Recall that n denotes the number of vertices of the digraph in question. In the next section we will consider branching flows in networks where the capacities are $n-k$ for some constant k. To study these, it turns out that it is relevant to study the problem of finding arc-disjoint k-safe s-out-branchings in digraphs with n vertices for some given constant k. When $k=2$ the connection is straightforward as every s-branching flow in $\mathcal{N} = (V, A, u)$ with $u \equiv n-2$ corresponds to a 2-safe branching (at least two children at the root). Hence there are arc-disjoint s-branching flows x_1, x_2 from s in $\mathcal{N}$ if and only if the network contains two arc-disjoint branchings, each of which uses at least two arcs from the root and such branchings are 2-safe. For higher values of k we can make use of Theorem 3.1 which also plays an important role in the next section.

The following consequence of the so-called strong version of Edmonds' Branching Theorem (see e.g. [12, Theorem 53.1]) will be useful in our search for arc-disjoint k-safe branchings. We include the proof for completeness.

By Edmonds' Branching Theorem, there are two arc-disjoint s-out-branchings if and only if we have

$$d^-(X) \geq 2 \quad \text{for all } X \subseteq V - s \tag{1}$$

Theorem 3.1 *Let D be a digraph with two arc-disjoint s-out-trees $T^+_{s,1}, T^+_{s,2}$. Then D contains two s-out-branchings $B^+_{s,1}, B^+_{s,2}$ such that $A(T^+_{s,i})$ is contained in $A(B^+_{s,i})$ for $i = 1, 2$ if and only if (1) holds and $D - A(T^+_{s,i})$ has an out-branching (in which case it has one which contains all the arcs of $T^+_{s,3-i}$) for $i = 1, 2$. Furthermore, such branchings can be found in polynomial time when they exist.*

Proof: The two conditions above are clearly necessary and can be checked in polynomial time, so assume that they are both satisfied. We will show how to modify the algorithmic proof of Edmonds' Branching Theorem by Lovász, for the case of two branchings (see e.g. [3, Section 9.3] or [12, Section 53.1]) so that we achieve two arc-disjoint branchings $B^+_{s,1}, B^+_{s,2}$ with $A(T^+_{s,i}) \subseteq A(B^+_{s,i})$ for $i = 1, 2$. We give the full proof as it is not very long and also makes it easy to check that there is a polynomial algorithm for constructing the desired branchings.

First we delete all arcs of $T^+_{s,2}$, call the resulting digraph D' and start the algorithm from the out-tree $T^+_s = T^+_{s,1}$ in D'. We say that a set $X \subseteq V - V(T^+_{s,2})$ is **dangerous** if it has in-degree 1 in $D' - A(T^+_s)$, where T^+_s is the current out-tree (containing $T^+_{s,1}$) which has been constructed so far by the algorithm. By submodularity of the in-degree function, if X, Y are both dangerous and they intersect, then so are $X \cup Y$ and $X \cap Y$. Also, since D satisfies (1), every dangerous set must intersect $V(T^+_s)$.

We have two possible scenarios when trying to extend T^+_s. Note that, by our assumption, as long as $V - V(T^+_s) \neq \emptyset$ there is at least one arc from $V(T^+_s)$ to $V - V(T^+_s)$:

(a) Every dangerous set (possibly there are none) is contained in $V(T^+_s)$.

(b) Some dangerous set intersects both $V(T^+_s)$ and $V - V(T^+_s)$.

If (a) holds, then we can take any arc $uv \in A(D')$ with $u \in V(T^+_s)$ and add uv to T^+_s and continue. If (b) holds, then (as in the original proof by Lovász) let X be an inclusionwise minimal dangerous set intersecting both $V(T^+_s)$ and $V - V(T^+_s)$. It is easy to check that there is some arc $uv \in A(D')$ with $u \in V(T^+_s) \cap X$ and $v \in X - V(T^+_s)$: Suppose not, then every arc which enters $X - V(T^+_s)$ also enters X and we would have

$$d^-_D(X - V(T^+_s)) = d^-_{D'}(X - V(T^+_s)) = d^-_{D' - A(T^+_s)}(X - V(T^+_s)) \leq d^-_{D' - A(T^+_s)}(X) = 1,$$

contradicting (1). So there is an arc $uv \in A(D')$ with $u \in V(T^+_s) \cap X$ and $v \in X - V(T^+_s)$. This arc cannot enter a dangerous set since (by the remark above on intersecting dangerous sets) that would contradict the minimality of X. Hence we may extend T^+_s by adding uv and continue the algorithm until T^+_s is spanning. At that point, since every set not intersecting $V(T^+_{s,2})$ has in-degree at least one in $D - A(T^+_s)$, we can extend $T^+_{s,2}$ to a branching arc-disjoint from the first one and the proof is complete.

◇

Corollary 3.2 *A digraph D contains a pair of arc-disjoint s-out-branchings such that s has out-degree at least 2 in both if and only if $d^+(s) \geq 4$ and D contains a pair of arc-disjoint branchings.*

Proof: By Theorem 3.1 it suffices to show that we can find two arc-disjoint out-trees $T^+_{s,1}, T^+_{s,2}$, each of which have out-degree 2 at s, such that there are arc-disjoint branchings extending each of these.

Call a set $X \subseteq V \setminus \{s\}$ **tight** if $d^-(X) = 2$. If $X, Y \subseteq V \setminus \{s\}$ are distinct tight sets both containing two out-neighbours of s, then they cannot share exactly one out-neighbour of s: Suppose p, q, r are all out-neighbours of s such that $X \cap \{p, q, r\} = \{p, q\}$ and $Y \cap \{p, q, r\} = \{q, r\}$. Then it follows from (1) the submodularity of the in-degree function that we have

$$2 + 2 = d^-(X) + d^-(Y) \geq d^-(X \cup Y) + d^-(X \cap Y) \geq 3 + 2, \tag{2}$$

a contradiction.

It follows from the observation above and the fact that $d^+(s) \geq 4$ that we can find four out-neighbours p, q, r, t of s such that there is no tight set containing both p, q and no tight set containing both r, t. This is clear if there are no tight sets containing two neighbours of s and otherwise let X be a tight set containing two neighbours a, b of s. By the claim there is no tight set containing a but not b and conversely so the claim follows. Now let $T^+_{s,1}$ be the out-tree formed by the two arcs sp, sq and $T^+_{s,2}$ be the out-tree formed by the arcs sr, st and apply the algorithm of Theorem 3.1. ◇

See [4] for a more results related to Corollary 3.2.

Theorem 3.1 can be generalized to extensions of p arc-disjoint s-out-trees. In [**?**] Szegö attributes the following result to Frank.

Theorem 3.3 *Let $D = (V, A)$ be a digraph and let $T^+_{s,1}, \ldots, T^+_{s,p}$ be arc-disjoint s-out-trees in D. There exists arc-disjoint s-out-branchings $B^+_{s,1}, \ldots, B^+_{s,p}$ in D such that $A(T^+_{s,i}) \subseteq A(B^+_{s,i})$ for $i \in [p]$ if and only if*

$$d^-_{D'}(X) \geq q(X) \text{ for all } X \subseteq V, \tag{3}$$

where D' is the digraph obtained from D by deleting all arcs of $T^+_{s,1}, \ldots, T^+_{s,p}$ and $q(X)$ denotes the number of s-out-trees among $T^+_{s,1}, \ldots, T^+_{s,p}$ which do not intersect X.

The proof of a generalization of this result can be found in [**?**]. This proof, which is a direct generalization of the proof we used for Theorem 3.1, can be turned into a polynomial algorithm which either constructs the desired s-out-branchings or finds a set X violating (3) above.

Theorem 3.4 *For every pair of fixed positive integer k, p, there exists a polynomial algorithm for checking whether a given input digraph D has p arc-disjoint k-safe s-out-branchings. Furthermore, such branchings can be found in polynomial time when they exist.*

Proof: Observe that an s-out-branching is k-safe if and only if it contains an s-out-tree with at most $2k-1$ vertices which is itself k-safe. Let $\mathcal{B}$ be the set of all k-safe s-out-trees from s on at most $2k-1$ vertices. As k is a constant, this set can be constructed in polynomial time $O(|V(D)|^{2k-1})$. Now, by Theorem 3.3, it suffices to check, using the algorithmic version of the theorem for (at most) all possible collections of p arc-disjoint out-trees $T^+_{s,1} \ldots, T^+_{s,p} \in \mathcal{B}$ whether the current collection can be extended to p arc-disjoint s-out-branchings and return the branchings constructed if some iteration is successful. If no set of p arc-disjoint out-trees $T^+_{s,1} \ldots, T^+_{s,p} \in \mathcal{B}$ can be extended, then D has no set of p arc-disjoint k-safe out-branchings from s. ◇

4 Arc-disjoint branching flows, the case of high capacities

We now study the complexity of finding arc-disjoint branching flows in networks where every arc has capacity $n-k$ for some constant k. If x is an s-branching flow in $D = (V, A)$, then let $D_x = (V, A_x)$ be the subdigraph of D with vertex set V and arc set $A_x = \{a \in A \mid x(a) > 0\}$. Thus D_x only contains the arcs with non-zero flow in x.

4.1 Growing and trimming a branching flow

Definition 4.1 Let $D = (V, A)$ be a digraph of order n, let $s \in V(D)$ be given, and let k be an integer. An **(s, k)-branching flow**, is a branching flow x from s in D, such that $x(e) \leq n - k$ for all arcs[1] $e \in A$. Note that x is also an (s, k)-branching flow in D_x.

(s, k)-branching flows have the the nice property that an (s, k)-branching flow of an induced subdigraph of D (containing s) can be extended to an (s, k)-branching flow of D provided that every vertex of D is reachable from s. This extra condition is not too demanding because it is a trivial necessary condition for a digraph to have an (s, k)-branching flow.

Lemma 4.2 *Let D be a digraph, let s be a vertex of D such that s can reach all vertices in D, and X a set of vertices containing s. If $D[X]$ contains an (s, k)-branching flow, then D contains an (s, k)-branching flow.*

Proof: Since one can reach all vertices of D, there is an ordering $v_1, \ldots, v_p$ of the vertices of $V(D) \setminus X$ such that for all $1 \leq i \leq p$, there is an (s, v_i)-path P_i in $D_i := D[X \cup \{v_1, \ldots, v_i\}]$.

Assume that $D_0 = D[X]$ contains an (s, k)-branching flow x^0. Let us prove by induction on i that D_i contains an (s, k)-branching flow x^i for all $0 \leq i \leq p$.

[1] It is easy to check that every s-branching flow from s which decomposes into path flows (no cycle in the decomposition) is an $(s, 1)$-branching flow.

Suppose now that $i \geq 1$. By the induction hypothesis, D_{i-1} contains an (s,k)-branching flow x^{i-1} in D_{i-1}. Let x^i be the flow obtained from x^{i-1} by adding one unit of flow on all arcs in P_i. Clearly, x^i is an (s,k)-branching flow in D_i.

Hence x^p is an (s,k)-branching flow in $D_p = D$. ◇

Definition 4.3 An **(s,k)-core** in D is a set X of vertices containing s, of cardinality $2k$, and such that $D[X]$ contains an (s,k)-branching flow.

Consider a digraph D and a vertex s from which all vertices of D are reachable. By Lemma 4.2, if D has an (s,k)-core, then it contains an (s,k)-branching flow. We shall now prove that this easy sufficient condition is also necessary (provided that D has order at least $2k$).

Theorem 4.4 *Let k be a positive integer, let D be a digraph D of order at least $2k$, and let s be a vertex from which all vertices of D are reachable. D contains an (s,k)-branching flow if and only if it has an (s,k)-core.*

In order to prove this theorem, we need two lemmas.

Lemma 4.5 *Let $D = (V,A)$ be a digraph on n vertices and let k be a positive integer such that $n > 2k$. Assume that there exists an (s,k)-branching flow, x, in D and let $A' \subseteq A_x$ be arbitrary. Let N be the network consisting of the digraph D_x and with capacity $u(e) = n-k$ on all $e \in A_x \setminus A'$ and $u(e) = n-k-1$ for all $e \in A'$.*

If there is no feasible s-branching flow in N, then there exists a set $X \subseteq V$, such that $s \in X$, $|X| = k$, there is only one arc, e, out of X in D_x and $e \in A'$.

Proof: Let N be the network defined as in the statement of the lemma and let N_k be the same as N, except all capacities are $n-k$. In both N and N_k add a vertex t and an arc from all vertices in $V(D) \setminus \{s\}$ to t of capacity 1 and let the resulting networks be called N^t and N_k^t, respectively. By the assumption of the lemma, there exists an (s,t)-flow in N_k^t of value $n-1$, but there does not exist an (s,t)-flow in N^t of value $n-1$. Let $(X, V \cup \{t\} \setminus X)$ be a minimum (s,t)-cut in N^t, which, by the Max-Flow Min-Cut Theorem (see e.g. [3, Theorem 4.5.3]), has capacity less than $n-1$. Recall that, by definition of an (s,t)-cut, $s \in X$. Thus by the existence of x, the cut $(X, V \cup \{t\} \setminus X)$ has capacity at least $n-1$ in N_k. Therefore some arc of A' must leave X, as otherwise the cut would have the same capacity in N^t and N_k^t. We must have $d^+_{D_x}(X) \leq 1$ as otherwise the capacity of the cut in N^t would be at least $2(n-k-1) > n-2$, a contradiction. Therefore there is exactly one arc, e, out of X in D_x and we have $e \in A'$. Finally if $|X| < k$ then the capacity of $(X, V \cup \{t\} \setminus X)$ in N_k^t would be at most $(|X|-1)+n-k < n-1$, a contradiction, and if $|X| > k$ then the capacity of $(X, V \cup \{t\} \setminus X)$ in N^t would be $(|X|-1)+n-k-1 > n-2$, a contradiction. Therefore $|X| = k$, completing the proof. ◇

Lemma 4.6 *Let k be a positive integer. If there exists an (s,k)-branching flow x in a digraph D of order greater than $2k$, then there exists an (s,k)-branching flow in $D-v$ for some vertex $v \in V(D) \setminus \{s\}$.*

Proof: Let x and D be defined as in the statement of the lemma. Any set, X, with $s \in X$ and $|X| = k$ and $d^+_D(X) = 1$ is called a **critical** set in D.

We first show that at most one critical set exists in D. For the sake of contradiction assume that X and Y are distinct critical sets in D. As $X \neq Y$ we note that $|X \cap Y| < k$. As x has flow $n-1-(|X \cap Y|-1) > n-k$ leaving $X \cap Y$, we note that there are at least two arcs, e_1 and e_2, leaving $X \cap Y$ in D. Each e_i leaves X or Y (or both). As only one arc leaves X and only one arc leaves Y we can without loss of generality assume that e_1 leaves X but not Y and e_2 leaves Y but not X. This implies that no arc leaves $X \cup Y$ as otherwise $d^+(X)$ or $d^+(Y)$ would be greater than one. As $n > 2k$ and $|X \cup Y| \leq 2k-1$ we get a contradiction to s being able to reach all vertices in D. Therefore at most one critical set exists in D.

If Y is a critical set in D, then let e be the arc out of Y in D and let $A' = A(D) \setminus \{e\}$ and if no critical set exists in D then let $A' = A(D)$. By Lemma 4.5, we note that there exists an (s,k)-branchng flow y in D with $y(e') \leq n-k-1$ for all $e' \in A'$.

Now, without loss of generality, we may assume that $D = D_y$ (that is, every arc in D has non-zero flow). We may also assume that D is acyclic, as if there exists a cycle in D then we can reduce the flow along this cycle until one of the arcs gets flow zero and therefore disappears from D_y.

Let P be a longest path in D starting at s, such that if some arc has flow $n-k$ in y, then this arc belongs to P. Let v be the last vertex on this path. We must have $d^+_D(v) = 0$ as otherwise there either exists a cycle in D or the path P wasn't longest possible. Let $D' = D - v$ and consider the flow y' in D' obtained from y by reducing the flow of each arc on P by one. By construction, no arc $e' \in A(D')$ has $y'(e') = n-k$, which implies that $y'(e'') \leq n-k-1 = |V(D')| - k$ for all $e'' \in A(D')$. It is now not difficult to see that y' is an (s,k)-branching flow in $D-v$, which completes the proof. $\diamond$

Proof of Theorem 4.4: We already proved that having an (s,k)-core is a sufficient condition for D to contain an (s,k)-branching flow, because all vertices are reachable from s.

Let us now prove that it is a necessary condition. Assume that D contains an (s,k)-branching flow. We repeatedly apply Lemma 4.6 and remove vertices from D, until we are left with D_{2k} of order $2k$ which contains an (s,k)-branching flow. Then $V(D_{2k})$ is an (s,k)-core. $\diamond$

4.2 A polynomial algorithm for the case of capacities n minus a constant

We are now ready to apply the results developed in the previous subsection to obtain the desired algorithm.

Theorem 4.7 *For every fixed natural number k there exists a polynomial algorithm for deciding whether a given input digraph D, with a prescribed vertex s contains two arc-disjoint (s,k)-branching flows.*

Proof: Let the digraph $D = (V,A)$ on n vertices and the special vertex s be given. We may assume that $n > 2k$ since otherwise the size of D is constant and we can solve the problem by a brute force method.

As the arc-set of every branching flow from s contains an s-out-branching, we first check whether D satisfies (1) and stop if this is not the case. Hence we assume below that D satisfies (1).

Now for every choice of two sets, $X, Y \subseteq V$, such that $|X| = |Y| = 2k$, and $s \in X \cap Y$, we will check whether they are (s,k)-cores in two arc-disjoint digraphs D^X and D^Y in which every vertex is reachable from s. We proceed as follows. For every subset of arcs $A_X \in D[X]$ and $A_Y \in D[Y]$, let $D^X_{core} = (X, A_X)$ and $D^Y_{core} = (Y, A_Y)$ and check if the following hold.

(i): $A_X \cap A_Y = \emptyset$.

(ii): There exists an (s,k)-branching flow in D^X_{core} and an (s,k)-branching flow in D^Y_{core}.

(iii): We can partition $A(D) \setminus (A_X \cup A_Y)$ into A' and A'', such that there exists an s-out-branching in $D^X = (V(D), A_X \cup A')$ and also in $D^Y = (V(D), A_Y \cup A'')$.

Clearly, X and Y are (s,k)-cores in two arc-disjoint digraphs D^X and D^Y in which every vertex is reachable from s if and only if (i)-(iii) are possible. Hence, by Theorem 4.4, our algorithm is correct.

Let us now analyse its complexity. Part (i) can be checked in time $O(n+m)$ and part (ii) can be checked in $O(k^3)$ time, using a max-flow algorithm. Now let us show how to check part (iii) given that (i) and (ii) passed. Fix arbitrary spanning s-out-trees $T^+_{s,X}, T^+_{s,Y}$ in D^X_{core}, D^Y_{core} respectively and let D^* be the spanning subdigraph that we obtain from D by deleting the arcs in $(A_X \setminus A(T^+_{s,X})) \cup (A_Y \setminus A(T^+_{s,Y}))$. We claim that (iii) holds if and only D^* contains a pair of arc-disjoint s-out-branchings $B^+_{s,X}, B^+_{s,Y}$ such that $A(T^+_{s,X}) \subset A(B^+_{s,X})$ and $A(T^+_{s,Y}) \subset A(B^+_{s,Y})$. This follows from the fact that every s-out-branching B^+_s in $D^X = (V(D), A_X \cup A')$ can be converted to an s-out-branching $B^+_{s,X}$

such that $A(T^+_{s,X}) \subset A(B^+_{s,X})$ by replacing the arcs of B^+_s that enters a vertex in X by $A(T^+_{s,X})$. The same applies to every s-out-branching B^+_s in $D^Y = (V(D), A_Y \cup A')$ and $A(T^+_{s,Y})$. Now it follows from Theorem 3.1 that all we need to do is to check that D^* contains a pair of arc-disjoint s-out-branchings and then check that $D^* - A_X$ contains an s-out-branching and $D^* - A_Y$ contains an s-out-branching. For a fixed choice of A_X, A_Y all of this can be done in time $O(n^3)$ (the time to check that (1) holds).

Note that there are less than n^{2k} choices for choosing X and less than n^{2k} choices for choosing Y. As $|A(X)|, |A(Y)| < 2k^2$ we note that there are less than $2^{(2k^2)}$ options for choosing A_X and analogously for choosing A_Y. Therefore there are less than $n^{4k} \times 2^{(4k^2)}$ options for choosing X, Y, A_X and A_Y, which is a polynomial when k is a constant. Therefore the algorithm runs in polynomial time. $\diamond$

It is not difficult to see that using the algorithmic version Theorem 3.3 we obtain the following.

Theorem 4.8 *For every pair of fixed natural numbers k, p there exists a polynomial algorithm for deciding whether a given input digraph D, with a prescribed vertex s contains p arc-disjoint (s,k)-branching flows.*

Modifying the first part of the proof of Theorem 3.4 we can even generalize as follows. We leave the easy details to the interested reader.

Theorem 4.9 *For every fixed natural number p and every collection of fixed natural numbers $k_1, \ldots, k_p$ there exists a polynomial algorithm for deciding whether a given input digraph D, with a prescribed vertex s contains p arc-disjoint branching flows $x_1, \ldots, x_p$ such that x_i is an (s,k_i)-branching flow.*

5 The case of capacities $n - f(n)$

We now show that, under a common complexity theoretic assumption, we cannot decrease the capacities by much more than a constant before the problem of the existence of arc-disjoint branching flows becomes non-polymomial.

Exponential Time Hypothesis (ETH): [7] *There is a positive real s such that 3-SAT with n variables and m clauses cannot be solved in time $2^{sn}(n+m)O(1)$.*

If the ETH is true, then there is no algorithm that solves 3-SAT in subexponential time. That is, there is no $2^{o(n)}$ algorithm that solves a 3-SAT instances with n variables. Using the so-called Sparsification Lemma we can obtain the following lemma.

Lemma 5.1 *[7] Given ETH, there is a positive real s' such that 3-SAT with n variables and m clauses cannot be solved in time $2^{s'm}(n+m)O(1)$.*

Lemma 5.2 *Assume the ETH holds and let $\epsilon > 0$ be arbitrary and let $k(n)$ be an integer function, such that $(log(n))^{1+\epsilon} \leq k(n) \leq n/2$ for all $n > 0$. Furthermore assume that there exists a constant C^* such that for all $c \geq C^*$ there exists an n such that $k(n) = c$.*

Then there is no polynomial (in n) algorithm for deciding if a digraph D of order n and with $s \in V(D)$ contains two arc-disjoint $(s, k(n))$-branching flows.

Proof: Let $\mathcal{F}$ be an instance of 3-SAT, with variables $v_1, v_2, \ldots, v_l$ and clauses $C_1, C_2, \ldots, C_m$. That is, $\mathcal{F} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$. We will construct a digraph D as follows (see Figure 1).

Let $X_1, X_2, X_3 \ldots, X_l$ and $W_1, W_2, \ldots, W_{l-1}$ be independent vertex sets of size two. Let Q be an independent set of size q (q will be determined later) and let the vertex set of D be the following.

$$V(D) = \{s, t, c_1, c_2, \ldots, c_m\} \cup X_1 \cup X_2 \cup \cdots \cup X_l \cup W_1 \cup W_2 \cup \cdots W_{l-1} \cup Q$$

Add all possible arcs from X_i to W_i and from W_i to X_{i+1} for all $i = 1, 2, 3, \ldots, l-1$. Add two parallel arcs from s to each vertex in X_1 and two parallel arcs from t to each vertex in Q. Add both arcs from X_l to t. Denote the vertices of X_i as $X_i = \{x_i, \bar{x}_i\}$ for $i = 1, 2, \ldots, l$ and if clause C_j contains

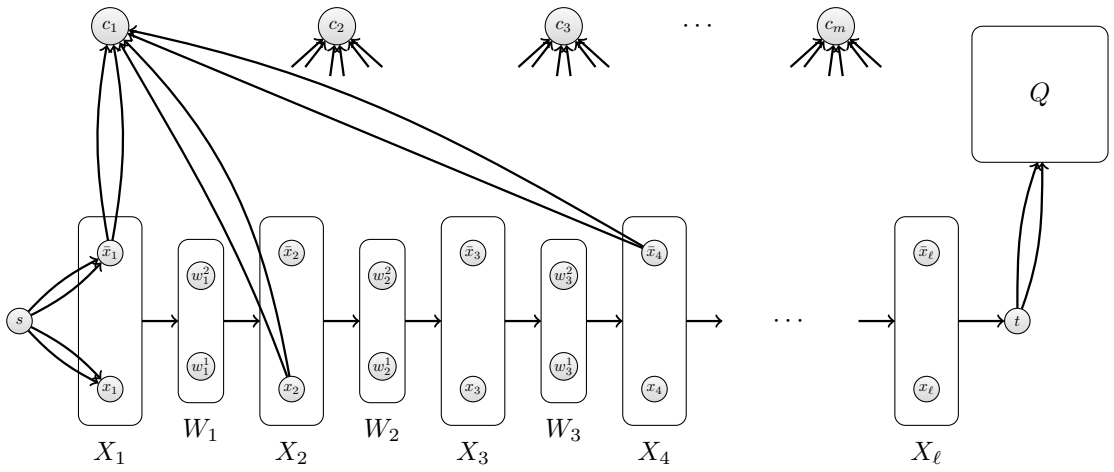

Figure 1: The digraph D, where $C_1 = (\bar{v}_1 \vee v_2 \vee \bar{v}_4)$

literal v_s then add two parallel arcs from x_s to c_j and if C_j contains literal $\bar{v}_s$ then add two parallel arcs from $\bar{x}_s$ to c_j (for all $j = 1, 2, \ldots, m$). Note that $|V(D)| = n = 4l + m + q$. Let q be determined such that $k(4l + m + q) = k(n) = 2l + m$. By the definition of $k(n)$ we note that the following holds (as we may assume that $2l + m > C^*$, where C^* is defined in the statement of the lemma).

$$(log(n))^{1+\epsilon} \leq k(n) = 2l + m \leq \frac{n}{2}$$

We will now consider the problem of deciding if D contains two arc-disjoint $(s, k(n))$-branching flows. First assume that it does and let B_1 and B_2 denote the two arc-disjoint $(s, k(n))$-branching flows. As all vertices in $V(D) \setminus \{s, c_1, c_2, \ldots, c_m\}$ have in-degree two and all vertices in $\{c_1, c_2, \ldots, c_m\}$ have out degree zero (and therefore in-degree exactly one in B_i for $i = 1, 2$) we note that the arcs carying flow in B_1 and B_2 must form arc-disjoint out-branchings in D. Below we will use the name B_i for both the flow B_i and the corresponding branching in D.

Let P_i be the (s, t)-path in B_i for $i = 1, 2$. Note that $|V(P_i)| = 2l + 1$ and that in B_i we include an arc from t to every vertex in Q. Therefore we must send at least $2l + q$ units of flow through the first arc in P_i in B_i. As $2l + q = n - k(n)$ we must send exactly this much flow through the arc, which implies that all vertices in $\{c_1, c_2, \ldots, c_m\}$ belong to a subtree, S_i, in B_i which does not start with the first arc of P_i. Note that S_i contains exactly one vertex in each X_j, for $j = 1, 2, \ldots, l$ (the other is on P). If S_i contains the vertex x_j then let $v_j = True$ and if S_i contains the vertex $\bar{x}_j$ then let $v_j = False$. It is not difficult to see that clauses C_r are satisfied as the arc into c_r indicates which literal is true. Therefore $\mathcal{F}$ is satisfied in this case.

Conversely, assume that $\mathcal{F}$ is satisfied by the truth assignment t. Let X' contain x_j if $v_j = True$ under t and let X' contain $\bar{x}_j$ otherwise. Let $X'' = (X_1 \cup X_2 \cup \cdots \cup X_l) \setminus X'$. Denote the vertices of W_i by $W_i = \{w_i^1, w_i^2\}$ for all $i = 1, 2, \ldots, l - 1$. Let P_i be the (s, t)-path containing all vertices in X'' and all vertices w_j^i for $j = 1, 2, 3, \ldots, l - 1$ and $i = 1, 2$. Let S_i be the out-tree containing all vertices in $X' \cup \{c_1, c_2, \ldots, c_m\}$ and all vertices w_j^{3-i} for $j = 1, 2, 3, \ldots, l - 1$ and $i = 1, 2$ (which is possible as $\mathcal{F}$ is satisfied by t). Let B_i be the out tree obtained by adding (a copy of) the arcs sx_1 and $s\bar{x}_1$ and (a copy of) each arc from t to Q to P_i and S_i. Note that B_1 and B_2 are two arc-disjoint out-branchings, which can easily be made into $(s, k(n))$-branching flows (by sending the appropriate amount of flow through the arcs of B_i). We have now shown that there exists two disjoint $(s, k(n))$-branching flows if and only if $\mathcal{F}$ is satisfiable.

For the sake of contradiction assume that there is a polynomial (in n) algorithm for deciding if a digraph D of order n and with $s \in V(D)$ contains two arc-disjoint $(s, k(n))$-branching flows. Let the algorithm have complexity $O(n^c)$, for some constant c.

As $(log(n))^{1+\epsilon} \leq k(n)$ we note that $log(n) \leq k(n)^{1/(1+\epsilon)}$, which gives us an algorithm with the following complexity.

$$O(n^c) = O\left(\left(2^{(k(n)^{1/(1+\epsilon)})}\right)^c\right) = O\left(\left(2^{c((2l+m)^{1/(1+\epsilon)})}\right)\right)$$

As $l \leq 3m$, the running time is at most $O(2^{c(7m)^{\epsilon'}})$, where $\epsilon' = 1/(1+\epsilon) < 1$. This is a contradiction to Lemma 5.1. ◇

Let $\epsilon > 0$ be arbitrary. By Lemma 5.2 we note that there is no polynomial (in n) algorithm (assuming the ETH) for deciding if a digraph D of order n and with $s \in V(D)$ contains two arc-disjoint $(s, \lfloor log(n)^{1+\epsilon} \rfloor)$-branching flows. Furthermore, if $c < 1/2$ then there is also no polynomial algorithm for deciding if a digraph D of order n and with $s \in V(D)$ contains two arc-disjoint $(s, \lfloor cn \rfloor)$-branching flows.

6 Remarks and open problems

6.1 Analogous statements in undirected graphs

A natural question is to ask whether the analogues of our results holds for undirected graph. Some of them do. For example, the analogous statement to Corollary 3.2 is easy to show.

Proposition 6.1 *A graph G contains a pair of edge-disjoint s-rooted spanning trees such that s has degree at least 2 in both if and only if $d(s) \geq 4$ and D contains a pair of s-rooted spanning trees.*

Proof: Suppose T, T' are edge-disjoint s-rooted spanning trees and s has degree only 1 in T' and at least 3 in T. Let $K_1, K_2, \ldots K_p$ be the connected components of $T - s$ and suppose w.l.o.g that $x \in K_1$ where x is the unique neighbour of s in T'. If we add the edge sy from s to K_2 in T to the tree T' we obtain a cycle C. Let uv be any edge on C that enters K_2. Then $T - sy + uv$ and $T' - uv + sy$ are the desired spanning trees. ◇

In contrast, the analogue of Theorem 3.1 does not hold for edge-disjoint spanning trees in undirected graphs. To see this, consider the graph $G = (V, E)$ with $V = \{s, a, b, c, d\}$ and edges $E = \{sa, sb, sc, sd, ab, ad, bc, cb\}$ (so there are two parallel edges between b and c). G is the union of two spanning trees and $G - \{sa, sd\}$ and $G - \{sb, sc\}$ are both connected but there is no pair of spanning trees T_1, T_2 such that T_1 contains the edges sa, sd and T_2 contains the edges sb, sc.

Still we can check for such trees efficiently using matroids. This is not new but we include the proof here for completeness.

Theorem 6.2 *Let p be a fixed natural number. There exists a polynomial-time algorithm for the following problem. Given an undirected graph $G = (V, E)$ and p edge-disjoint forrests $U_1, \ldots, U_p$; decide whether G contains p edge-disjoint spanning trees $T_1 \ldots, T_p$ such that $E(U_i) \subseteq E(T_i)$ for $i \in [p]$.*

Proof: We show how to formulate this problem as a matroid partition problem. Given $G = (V, E)$ and p disjoint subsets $E_1, \ldots, E_p$ of E, each inducing a forrest in G, we first define the following p subsets of E:
$\mathcal{F}_i = \{E' \subseteq E \setminus E_i : E' \cup E_i$ induces a forest in $G\}$ for $i \in [p]$. It is well-known and easy to show that each $\mathcal{F}_i$ is the set of independent sets of a matroid and hence, by Edmonds' Matroid Union Theorem, the family $\mathcal{F} = \{X \subset E : \exists X_1, \ldots, X_p \subset E$ such that $X = X_1 \cup \ldots \cup X_p, X_i \cap X_j = \emptyset$ when $i \neq j$ and $X_i \in \mathcal{F}_i, i \in [p]\}$ is the set of independent sets of a matroid (see .e.g. [11]). Now it follows that G contains the desired edge-disjoint trees if and only if the size of a base in the matroid $M = (E, \mathcal{F})$ is $p(|V| - 1) - \sum_{i \in [p]} |E_i|$. Furthermore, applying the greedy algorithm for constructing a base of M we obtain either a solution or a proof that none exists in polynomial time. ◇

The trees we will consider below are rooted at a prescribed vertex s and we say that T is **s-rooted** if it is rooted in s. Let us call an s-rooted spanning tree T **k-safe** if the tree that remains after deleting the subtree of any of the neighbours of s has at least k vertices.

Theorem 6.3 *For every pair of fixed integers k, p, there exists a polynomial-time algorithm for deciding whether an undirected graph G with prescribed root s contains a collection of p edge-disjoint k-safe s-rooted spanning trees.*

Proof: Observe that an s-rooted spanning tree T is k-balanced if and only if it contains an s-rooted subtree with at most $2k-1$ vertices which is itself k-balanced (we need $2k-1$ only if s has degree 2 in T). Now let $\mathcal{T}$ be the set of all k-balanced s-rooted trees on at most $2k-1$ vertices in G. We can find the set $\mathcal{T}$ in time $O(|V|^{2k-1})$. For each collection of p edge-disjoint trees $U_1, \ldots, U_p$ of $\mathcal{T}$ each containing s we can test in polynomial time using the algorithm of Theorem 6.2 whether there are p edge-disjoint s-rooted spanning trees that extend $U_1, \ldots, U_p$ respectively. By the remark above this proves the theorem. ◇

6.2 Further research

Theorem 4.7 asserts that for every fixed integer k, deciding whether an input digraph with a prescribed vertex s contains two arc-disjoint (s,k)-branching flows can be done in polynomial time. A natural question is to ask about the parameterized complexity of the problem with respect to the parameter k. The above result shows that the problem is in XP. A natural question is to ask whether it is fixed-parameter tractable (FPT) or not.

Problem 6.4 *Does there exists an integer-valued function f a constant c and an algorithm that, given an integer k, an input digraph D and a vertex s in D, decides in time $f(k) \cdot n^c$ whether D contains two arc-disjoint (s,k)-branching flows ?*

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] J. Bang-Jensen and S. Bessy. (Arc-)disjoint flows in networks. *Theoretical Computer Science*, in Press, 2014.

[3] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications, 2nd Edition.* Springer-Verlag, London, 2009.

[4] J. Bang-Jensen and A. Yeo. Balanced branchings in digraphs. manuscript, 2014.

[5] J Daligault, G. Gutin, E.J. Kim, and A. Yeo. FPT Algorithms and Kernels for the Directed k-Leaf Problem. *Journal of Computer and System Sciences*, pages 144–152, 2010.

[6] J. Edmonds. Edge-disjoint branchings. In B. Rustin, editor, *Combinatorial Algorithms*, pages 91–96. Academic Press, 1973.

[7] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, pages 512–530, 2001.

[8] M. Las Vergnas. Sur les arborescences dans un graphe orienté. *Discrete Math.*, 15(1):27–39, 1976.

[9] D. Lokshtanov, V. Raman, S. Saurabh, and S. Sikdar. On the directed degree-preserving spanning tree problem. 5917:276–287, 2009.

[10] L. Lovász. On two min–max theorems in graph theory. *J. Combin. Theory Ser. B*, 21:96–103, 1976.

[11] A. Recski. *Matroid theory and its applications in electric network theory and in statics.* Springer-Verlag, Berlin, 1989.

[12] A. Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. B*, volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2003. Matroids, trees, stable sets, Chapters 39–69.

[13] S. Thomassé. Covering a strong digraph by $\alpha-1$ disjoint paths: a proof of Las Vergnas' conjecture. *J. Combin. Theory Ser. B*, 83(2):331–333, 2001.