Theoretical Computer Science 526 (2014) 28–40

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

(Arc-)disjoint flows in networks

Jørgen Bang-Jensen [a,*,1,2], Stéphane Bessy [b]

[a] Department of Mathematics and Computer Science, University of Southern Denmark, Odense DK-5230, Denmark
[b] AlGCo, LIRMM, Université Montpellier 2, France

ARTICLE INFO

Article history:
Received 23 April 2013
Received in revised form 18 December 2013
Accepted 11 January 2014
Communicated by P. Widmayer

Keywords:
Arc-disjoint flows
Linkages
Spanning connected Eulerian digraph
Even cycle factor
Odd cycle factor
Acyclic digraph

ABSTRACT

We consider the problem of deciding whether a given network with integer capacities has two feasible flows x and y with prescribed balance vectors such that the arcs that carry flow in x are arc-disjoint from the arcs that carry flow in y. This generalizes a number of well-studied problems such as the existence of arc-disjoint out-branchings B_s^+, B_t^+ where the roots s, t may be the same vertex, existence of arc-disjoint spanning subdigraphs D_1, D_2 with prescribed degree sequences in a digraph (e.g. arc-disjoint cycle factors), the weak-2-linkage problem, the number partitioning problem, etc. Hence the problem is NP-complete in general. We show that the problem remains hard even for very restricted cases such as two arc-disjoint (s,t)-flows each of value 2 in a network with capacities 1 and 2 on the arcs. On the positive side, we prove that the above problem is polynomially solvable if the network is acyclic and the arc capacities as well as the desired flow values are bounded. Our algorithm for this case generalizes the algorithm (by Perl and Shiloach [14] for $k=2$ and Fortune, Hopcroft and Wyllie [11] for $k \geqslant 3$) for the k-linkage problem in acyclic digraphs. Besides, the problem is polynomial in general digraphs if all capacities are 1 and the two flows have the same balance for all vertices in $\mathcal{N}$, but remains NP-complete if the network contains at least one arc with capacity 2 (and the others have capacity 1). Finally, we also show that the following properties are NP-complete to decide on digraphs: the existence of a spanning connected Eulerian subdigraph, the existence of a cycle factor in which all cycles have even length and finally the existence of a cycle factor in which all cycles have odd length.

1. Introduction

Notation not given below is consistent with [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 $A=A(D)$. 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 $\{1,2,\dots,k\}$.

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 by $UG(D)$, is obtained from D by suppressing the orientation of each arc and deleting multiple edges. 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 a vertex $x \in V$ is the number of vertices y in V such that xy (resp. yx) is an arc of A. When $X \subseteq V$ we shall also write $d_X^+(v)$ to denote the number vertices in X that are dominated by v. A digraph $D=(V,A)$ is **Eulerian** if $d^+(v)=d^-(v)$ for all $v \in V$.

* Corresponding author.
E-mail addresses: jbj@imada.sdu.dk (J. Bang-Jensen), bessy@lirmm.fr (S. Bessy).

1 Research supported by the Danish research council under grant number DFF 602-01488.
2 Part of this work was done while the first author visited LIRMM, Université Montpellier 2, France and Mascotte-INRIA, Sophia Antipolis, France. The hospitality and financial support of both sites are gratefully acknowledged.

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2014.01.011

An **out-branching** rooted at s in a digraph D is a spanning tree in $UG(D)$ such that every vertex $v \neq s$ has exactly one arc entering. Equivalently, s has a directed path to every other vertex using only arcs of the tree. We use the notation B_s^+ to denote an out-branching rooted at s.

By a **spanning** subdigraph of a digraph $D = (V, A)$ (also called a **factor**) we mean a subdigraph $H = (V, A')$ with the same vertex set as D such that every vertex is incident to at least one arc from A', that is, $UG(H)$ has no isolated vertices. In particular, a **cycle factor** of D is a disjoint union of cycles that cover all vertices of D.

A **network** $\mathcal{N} = (V, A, u)$ is a digraph $D = (V, A)$ equipped with a capacity function $u : A \to \mathbf{R}_0$ on its arcs. A **flow** in $\mathcal{N}$ is any non-negative function $x : A \to \mathbf{R}_0$ which satisfies that $x_{ij} \leqslant u_{ij}$ 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, denoted by b_x, is the function on V which with 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 $v \in V$. One of the main theorems of flow theory 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]).

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). 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 1.1 *(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 paths 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 $O(nm)$.*

An (s, t)-**flow** is a flow x whose balance-vector is zero for all $v \notin \{s, t\}$ and $0 \leqslant b_x(s) = -b_x(t)$. The number $b_x(s)$ is called the **value** of x. By the flow decomposition theorem, for every (s, t)-flow x, there exists an (s, t)-flow x' (possibly $x' = x$) such that $b_{x'}(s) = b_x(s)$ and x' is the arc sum of at most $n+m$ path flows along (s, t) paths. A **branching flow** from s 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 $\mathcal{N}$.

Two flows x, y in a network $\mathcal{N}$ are **disjoint**, respectively **arc-disjoint**, if $x_{ij} \cdot y_{i'j'} = 0$ whenever $\{i, j\} \cap \{i', j'\} \neq \emptyset$, respectively whenever $ij = i'j'$.

The concept of flows in networks constitutes a very useful modelling tool and a large number of important problems can be formulated as (minimum cost) flow problems and hence solved in polynomial time. For a vast collection of results on flows see [1] (see also [3] for some other applications of flows to digraph problems). There are, however, a number of natural optimization problems that cannot be solved in polynomial time using the standard flow machinery, even though the problems have a 'flow flavour' in that they deal with paths and cycles in digraphs. One such example is the weak-k-linkage problem, where we are given vertices $s_1, s_2, \ldots, s_k, t_1, t_2, \ldots, t_k$ and we wish to decide the existence of k arc-disjoint paths $P_1, \ldots, P_k$ such that P_i is an (s_i, t_i)-path for $i \in [k]$. A classical result by Fortune, Hopcroft and Wyllie [11] asserts that the weak-k-linkage problem is NP-complete for all $k \geqslant 2$. Another example not solved by the flow theory is given by the problem of finding three (s, t)-paths in a digraph $D = (V, A)$ so that the first two may share arcs from a prescribed subset A' of A, but the third cannot share any arc with the other two.

In this paper, to obtain a more general framework including both the classical flow problems and also the problems mentioned above, we consider the question of deciding whether a given network with integer capacities has two feasible flows x and y with prescribed balance vectors such that the arcs that carry flow in x are (arc-)disjoint from the arcs that carry flow in y. This generalizes a number of well-studied problems such as the existence of arc-disjoint out-branchings B_s^+, B_t^+ where the roots s, t may be the same vertex, existence of arc-disjoint spanning subdigraphs D_1, D_2 with prescribed degree sequences in a digraph (e.g. arc-disjoint cycle factors), the weak-2-linkage problem, the number partitioning problem, etc.

In all these generalizations, the values of the capacity function play an important role. For instance, to model the existence of arc-disjoint out-branchings, we need to use a branching flow on a network with a constant capacity function (identically equal to $n - 1$, where n is the number of vertices of the considered network). As a direct corollary of Edmonds' branching theorem [9], we obtain in Section 3 a polynomial algorithm to decide if a network with capacity function identically equal to $n - 1$ admits k arc-disjoint branching flows from a given vertex s. However, if we restrain the capacity function to take values in $\{1, 2\}$, we show that the problem becomes NP-complete. In Section 4, we further investigate the role of the capacity function in the status of the studied problems. In particular, if the capacity function is identically equal to 1, then we can decide in polynomial time if a network contains two arc-disjoint flows with the same balance vector. We even generalize this result to k arc-disjoint flows, always with the same balance vector. Once again, we show that a slight modification in the capacity function (e.g. fixing one arc with capacity 2 and giving all the others capacity 1) leads to an NP-complete problem.

Another positive result in this context is given in Section 5. The arc-disjoint flow problem is polynomially solvable if the network is acyclic and the arc capacities as well as the desired flow values are bounded. Our algorithm for this case generalizes the algorithm (by Perl and Shiloach [14] for $k = 2$ and Fortune, Hopcroft and Wyllie [11] for $k \geqslant 3$) for the k-linkage problem in acyclic digraphs.

Finally, in order to provide tools for polynomial reductions in our NP-completeness proofs, we study some questions concerning spanning Eulerian subdigraphs of a given digraph, which are also worthy of interest by themselves. For instance, in Section 2, we prove that deciding the existence of a spanning connected Eulerian subdigraph is an NP-complete problem. In Section 6, we also address the problems of the existence of a cycle factor in which all cycles have even length, respectively odd, in a given digraph, and we show that these problems are NP-complete.

2. Eulerian subdigraphs and Eulerian factors

We start with a complexity result which is of independent interest (the corresponding result for undirected graphs was shown in [15]) and which will be used in the following section.

It is a classical application of flows to decide in polynomial time if digraph has a spanning (i.e. every vertex has non-zero degree) Eulerian subdigraph. For sake of completeness we briefly indicate the proof.

Theorem 2.1 *(Classical). There exists a polynomial time algorithm to decide if a digraph has spanning Eulerian subdigraph.*

Proof. Starting from a digraph $D=(V,A)$, we construct the network $\mathcal{N}=(V',A',u)$ as follows. The set V' contains vertices s and t and for each vertex v of V, we add to V' two vertices v_1 and v_2. For each vertex in v we create the arcs sv_1, v_2v_1 and v_2t in A' and for each arc uv of D we add to A' the arc u_1v_2. Finally, every arc gets capacity 1 in $\mathcal{N}$ except the arcs of type v_2v_1 which have infinite capacity (or say capacity n, where $n=|V|$). Now, it is easy to check that D has a spanning Eulerian subdigraph if, and only if, $\mathcal{N}$ has an (s,t)-flow of value n. ∎

However, if we ask for a connected spanning Eulerian subdigraph, the problem becomes NP-complete.

Theorem 2.2. *It is NP-complete to decide whether a digraph $D=(V,A)$ contains a spanning Eulerian subdigraph which is connected.*[3]

Proof. We will describe a polynomial reduction from 3-SAT to the problem of deciding whether a given digraph contains a spanning Eulerian subdigraph. For natural numbers integer p,q, let $W[u,v,p,q]$ be the digraph (the variable gadget) with vertex set $\{u,v,y_1,y_2,\dots,y_p,z_1,z_2,\dots,z_q\}$ and arc set equal to the union of the arcs of the two (u,v)-paths $uy_1y_2\dots y_pv$, $uz_1z_2\dots z_qv$. Calls these paths the y-path, respectively the z-path of $W[u,v,p,q]$.

Let $\mathcal{F}$ be an instance of 3-SAT with variables $x_1,x_2,\dots,x_n$ and clauses $C_1,C_2,\dots,C_m$. We may assume that each variable x occurs at least once either in the negated form or non-negated in $\mathcal{F}$. The ordering of the clauses $C_1,C_2,\dots,C_m$ induces an ordering of the occurrences of a variable x and its negation $\bar{x}$ in these. With each variable x_i, where x_i occurs p_i times and $\bar{x}_i$ occurs q_i times in the clauses of $\mathcal{F}$, we associate a copy $W[u_i,v_i,p_i,q_i]$ of $W[u,v,p,q]$. Identify the end vertices of these digraphs by setting $v_i=u_{i+1}$ for $i=1,2,\dots,n-1$. Let $s=u_1$ and $t=v_n$. Let D' be the digraph obtained in this way.[4]

For each $i\in[m]$ we associate the clause C_i with three of the vertices $V_i=\{a_{i,1},a_{i,2},a_{i,3}\}$ from the graph D' above as follows: assume C_i contains variables x_j,x_k,x_l (negated or not). If x_j is not negated in C_i and this is the rth copy of x_j (in the order of the clauses that use x_j), then we identify $a_{i,1}$ with the rth internal vertex of the y-path of $W[u_j,v_j,p_j,q_j]$ and if C_i contains $\bar{x}_j$ and this is the kth occurrence of $\bar{x}_j$, then we identify $a_{i,1}$ with the kth internal vertex of the z-path of $W[u_j,v_j,p_j,q_j]$. We make similar identifications for $a_{i,2}$, $a_{i,3}$. Thus D' contains all the vertices $\{a_{j,i}\mid j\in[m],i\in[3]\}$.

Claim 1. *D' contains an (s,t)-path P which contains at least one vertex from $V_j=\{a_{j,1},a_{j,2},a_{j,3}\}$ for each $j\in[m]$ if and only if $\mathcal{F}$ is satisfiable.*

Proof of Claim 1. Suppose P is an (s,t)-path which contains at least one vertex from V_j for each $j\in[m]$. By construction, for each variable x_i, P traverses either the y-subpath or the z-subpath of the corresponding gadget. Now define a truth assignment by setting x_i true precisely when the first traversal occurs for i. This is a satisfying truth assignment for $\mathcal{F}$ since for any clause C_j at least one literal is contained in P and hence becomes true by the assignment (the literals traversed become true and those not traversed become false). Conversely, given a truth assignment for $\mathcal{F}$ we can form P by routing it through all the true literals in the chain of variable gadgets. ∎

Now let D_{F} be obtained from D' by adding 3 new vertices $\{a'_{i,1},a'_{i,2},a'_{i,3}\}$ and the arcs of the 6-cycle $a_{i,1}a'_{i,1}a_{i,2}a'_{i,2}a_{i,3}a'_{i,3}a_{i,1}$ for each $i\in[m]$ and finally adding the arc ts, see Fig. 1. Let H be a spanning Eulerian subdigraph of D_{F}. Since $a'_{i,j}$ has in- and out-degree one for every $i\in[m]$ and $j\in[3]$, H has to contain all the 6-cycles $a_{i,1}a'_{i,1}a_{i,2}a'_{i,2}a_{i,3}a'_{i,3}a_{i,1}$ for $i\in[m]$. Moreover, since s has in-degree one in D_{F}, H must contain the arc ts and hence also a directed (s,t)-path. As the union of the m 6-cycles $a_{i,1}a'_{i,1}a_{i,2}a'_{i,2}a_{i,3}a'_{i,3}a_{i,1}$ forms an Eulerian subdigraph of D_{F}, H has to

[3] Following Catlin's [8] notion for undirected graphs, we call such a digraph **supereulerian**.

[4] This part of the construction has been used several times before in NP-completeness proofs, see e.g. [4–6].

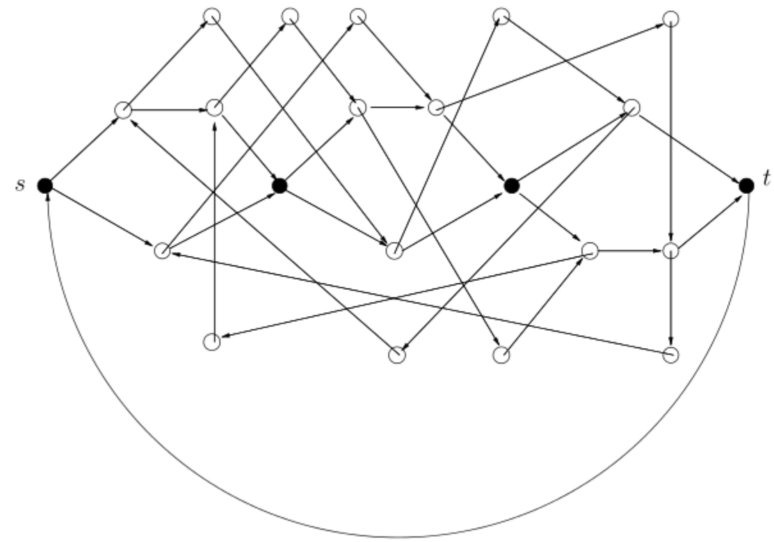

Fig. 1. An illustration of the digraph D_{F} when $\mathsf{F} = (x_1 + \bar{x}_2 + x_3)(x_1 + x_2 + \bar{x}_3)(\bar{x}_1 + x_2 + \bar{x}_3)$. The black vertices are the vertices u_i, v_i of the variable gadgets.

contain a directed (s,t)-path P which is a subdigraph of D'. Furthermore, if H is connected, P has to contain at least one vertex from every V_i in order to connect all the 6-cycles. Thus, by Claim 1, $\mathcal{F}$ is satisfiable. Conversely, by Claim 1, if $\mathcal{F}$ is satisfiable we obtain the desired spanning Eulerian subdigraph H by taking an (s,t)-path P which contains at least one vertex from V_i, $i \in [m]$ and the adding the m 6-cycles as above. ∎

Using similar arguments, we can also prove the following.

Theorem 2.3. *It is NP-complete to decide whether a digraph $D = (V, A)$ contains a spanning Eulerian subdigraph D' such that every connected component of D' has an even number of vertices.*

Proof. In the proof above, we replace each 6-cycle corresponding to a clause by a directed 9-cycle $a_{i,1}b_{i,1}a'_{i,1}a_{i,2}b_{i,2}a'_{i,2}a_{i,3}b_{i,3}a'_{i,3}a_{i,1}$, where all the $3m$ vertices $b_{i,j}$ are distinct. If the total number of vertices of D_{F} is odd (i.e. $n+m$ is even), we subdivide once the arc st to obtain an even number of vertices. Now it is easy to check that the new digraph has a spanning Eulerian factor all of whose components are even if and only if it has a connected spanning Eulerian digraph. ∎

Finally we state the following observation which will be used in the next section.

Lemma 2.4. *Every connected Eulerian digraph $H = (V, A)$ with $|V| = 2k$ even has a vertex partition $V = S \cup T$, with $|S| = |T| = k$ and a collection of $|V|$ arc-disjoint paths $P_1, \dots, P_k$, $Q_1, \dots, Q_k$ such that $P_1, \dots, P_k$ start in distinct vertices of S and end in distinct vertices of T and $Q_1, \dots, Q_k$ start in distinct vertices of T and end in distinct vertices of S.*

Proof. The following linear time algorithm constructs the desired partition and the paths: Find a closed Eulerian walk W of H in linear time. Let $T = \emptyset$ and $S = \emptyset$ and let $i = 1$, $j = 0$. Start at an arbitrary vertex v; add v to S and let P_1 be the path formed by the arc from v to its successor w in W. Increase i to 2. Let $j = 1$, add w to T and let $Q_1 = W[w, w']$ and add w' to S, where w' is the first vertex of W that has not been seen so far. Increase j to 2. In the general step: after having added a new vertex z to S (resp. T), we continue along W to the first new vertex z' and let the next path P_i (resp. Q_j) be any (z, z')-path contained in $W[z, z']$ and increase i (resp. j) by one. This process will eventually stop when we reach v for the last time (having traversed all of W) and now we let Q_k be any path contained in $W[z, v]$ where z is the last vertex added to T. ∎

3. Arc-disjoint branching flows

In this section, we consider flows along branchings. Clearly a digraph D has an out-branching from s if and only if it has an (s, v)-path for all $v \in V$. Recall that a branching flow from s 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 $\mathcal{N}$. We have the following straight equivalence.

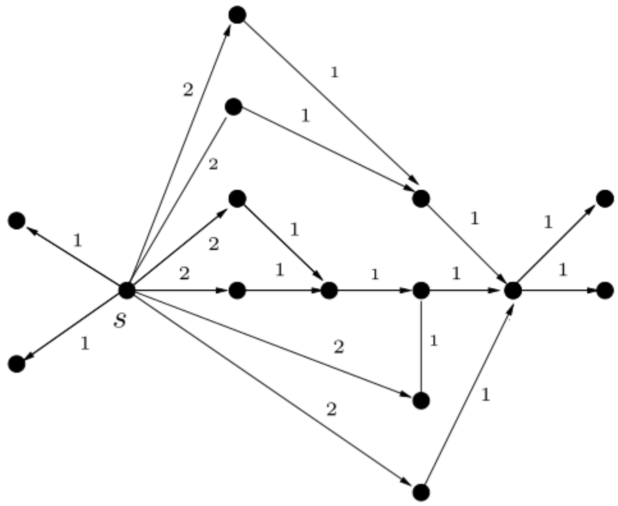

Fig. 2. A branching flow from s in a network with capacities 1 and 2.

Lemma 3.1. *A digraph $D = (V, A)$ has an out-branching B_s^+ rooted at s if and only if the network*[5] $\mathcal{N} = (V, A, u \equiv |V| - 1)$ *has a branching flow from s.*

Proof. By the flow decomposition theorem and the remark above, a digraph $D = (V, A)$ has an out-branching B_s^+ if and only if the network we obtain by letting all capacities equal to $n - 1$ has a branching flow from s. ∎

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).

Edmonds [9] characterized digraphs with k arc-disjoint branchings from a prescribed root.

Theorem 3.2 *(Edmonds' branching theorem). A digraph $D = (V, A)$ has k arc-disjoint branchings all rooted at s if and only if D has k arc-disjoint (s, v)-paths for every $v \in V - s$.*

By Edmonds' branching theorem and the algorithmic proof of the theorem due to Lovász [13] (see also [3, Section 9.3]), we have the following characterization of networks with all capacities equal to $n - 1$ that have k arc-disjoint branching flows:

Theorem 3.3. *A network $\mathcal{N} = (V, A, u \equiv |V| - 1)$ has k arc-disjoint branching flows $x^1, x^2, \dots, x^k$, all from s, if and only if there are k arc-disjoint (s, v)-paths in $\mathcal{N}$ for every $v \in V - s$. Furthermore, there is a polynomial algorithm for constructing such flows $x^1, x^2, \dots, x^k$ when they exist.* ∎

A branching flow x may have flow equal to $r \leqslant n - 1$ on some arc, corresponding to that arc belonging to r of the paths whose union 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 in the chosen solution. In particular, if $r = n - 1$, one arc failure can disconnect s from all other vertices in the chosen branching. Thus, from a practical point of view (say, in an application where branchings are used to route information), it could be interesting to consider branching flows where the maximum flow value in an arc is as small as possible. 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. Fig. 2 shows a typical structure of the acyclic part of a branching flow when $u_{ij} \leqslant 2$ for every arc ij in $\mathcal{N}$. Notice that the subdigraph corresponding to the arcs that carry flow contains a number of out-branchings from s, all of which satisfy that s has out-degree at least $\frac{n-1}{2}$.

However, despite the simple structure of branching flows, we obtain the following result concerning arc-disjoint branching flows in networks with maximum capacity 2,

Theorem 3.4. *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 branching flows from s.*

---

[5] That is, the capacity of each arc ij is $|V| - 1$.

Proof. We will give a polynomial reduction from the problem of deciding whether a digraph D contains an Eulerian factor with all components even to the problem above. Then the result will follow from Theorem 2.3. Given a digraph $D = (V, A)$ with an even number of vertices, we form the network $\mathcal{N}$ by adding one new vertex s and all possible arcs from s to V. These arcs get capacity 2 and all other arcs (those in A) get capacity 1. Suppose $\mathcal{N}$ has two arc-disjoint branching flows x, y from s. As these flows send a total of $2|V|$ units out of s, all arcs out of s are filled by either x or y (but not both). Since x, y are branching flows, each vertex receiving flow 2 from s in either flow must send one unit to some other vertex. This implies that x and y induce a partition $V = X \cup Y$ of V, into sets of the same size (implying that $|V| = 2k$ is even) where X (resp. Y) is the set of vertices receiving flow 2 from s in x (resp. y). By the remark above on vertices that receive 2 units of flow and the flow decomposition theorem, x can be decomposed into k arc-disjoint paths $P_1, \ldots, P_k$ all of which start in distinct vertices of X and end in distinct vertices of Y. Similarly y can be decomposed into k arc-disjoint paths $Q_1, \ldots, Q_k$ all of which start in distinct vertices of Y and end in distinct vertices of X. Since x and y are arc-disjoint, the digraph H formed by the union of $P_1, \ldots, P_k$ and $Q_1, \ldots, Q_k$ is Eulerian. Furthermore, every connected component C of H is even, since (X, Y) is a partition of V and $|V(C) \cap X| = |V(C) \cap Y|$ because every vertex in X (resp. Y) is the terminal vertex of some Q_i (resp. P_j).

Conversely suppose D contains an Eulerian factor H' all of whose components $H'_1, \ldots, H_p$ are even. By Lemma 2.4 we can find a partition of each connected component H_r of H into two sets X_r, Y_r of the same size k_r and paths $P_{r,1}, \ldots, P_{r,k_r}$, all of which start in distinct vertices of X_r and end in distinct vertices of Y_r as well as paths $Q_{r,1}, \ldots, Q_{r,k_r}$ all of which start in distinct vertices of Y_r and end in distinct vertices of X_r. Now we can obtain the desired flows x, y by letting x (resp. y) saturate the arcs from s to $X_1 \cup \cdots \cup X_p$ (resp. $Y_1 \cup \cdots \cup Y_p$) and send flow one on all of the paths $P_{1,1}, \ldots, P_{1,k_1}, \ldots, P_{p,1}, \ldots, P_{p,k_p}$ (resp. $Q_{1,1}, \ldots, Q_{1,k_1}, \ldots, Q_{p,1}, \ldots, Q_{p,k_p}$). ∎

4. Flows in unit and almost unit capacity networks

Even in unit capacity networks deciding the existence of arc-disjoint flows x and y is difficult because the weak-2-linkage problem is a special case.

Theorem 4.1. *It is NP-complete to decide whether a unit capacity network contains arc-disjoint flows x and y with prescribed balance vector for each.*

Proof. We reduce the weak 2-linkage problem to this problem. This follows from the easy observation that a digraph $D = (V, A)$ contains arc-disjoint (s_1, t_1)- and (s_2, t_2)-paths if and only if the unit capacity network $\mathcal{N} = (V, A, u \equiv 1)$ obtained by adding capacity 1 to every arc of D contains arc-disjoint flows x and y where x is an (s_1, t_1)-flow of value 1 and y is an (s_2, t_2)-flow of value 1. Thus the claim follows from the NP-completeness of the weak-2-linkage problem [11]. ∎

Next we consider the case when the two flows must have the same balance at every vertex and we show that this problem is tractable in unit capacity networks, whereas it becomes NP-complete if we allow arcs with capacity 1 and 2.

We need the following lemma, which is generalized by Lemma 4.4, but of the proof given here is more natural in some sense.

Lemma 4.2. *The edge set of every Eulerian bipartite graph $G = (V, E)$ can be split into two sets E_1, E_2 such that $d_{E_i}(v) = d(v)/2$ for all $v \in V$. Furthermore, this partition can be computed in polynomial time.*

Proof. Since G is Eulerian and bipartite we can decompose E into cycles of even length. Now taking every second edge on each of those cycles in E_1 and the others in E_2, we obtain the desired partition. As the decomposition of E into cycles can be computed in polynomial time (greedily for instance), we obtain the claimed partition in polynomial time also. ∎

Theorem 4.3. *Let $\mathcal{N} = (V, A, u \equiv 1, b)$ be a unit capacity network with a prescribed balance vector b such that*[6] *$b \not\equiv 0$. There exist arc-disjoint flows x and y in $\mathcal{N}$ both with balance vector b (that is, $b_x \equiv b_y \equiv b$) if and only if $\mathcal{N}$ has a feasible flow z with balance vector $b_z \equiv 2b$. Hence one can decide the existence of x and y in polynomial time.*

Proof. The 'only if' implication is clear so assume $\mathcal{N}$ has a feasible flow z with balance vector $2b$. Let $P_1, \ldots, P_{2k}, C_1, \ldots, C_r$ be a decomposition of z into an even number, $2k$ of path flows and $r \geqslant 0$ cycle flows each of value 1. Let $V_1 = \{v \in V:\ b(v) > 0\}$ and $V_2 = \{v \in V:\ b(v) < 0\}$. Each P_i starts in a vertex of V_1 and terminates in a vertex of V_2. Define the bipartite graph $B = (V_1, V_2, E)$ where each path P_i corresponds to an edge in B between its end vertices. Since $2b(v)$ is even for every $v \in V$, B is Eulerian and now we can apply Lemma 4.2 to partition $P_1, \ldots, P_{2k}$ into two sets of k paths such that the union of each of these sets gives a flow with balance b in $\mathcal{N}$. As $P_1, \ldots, P_{2k}$ are arc-disjoint the theorem follows. ∎

[6] If $b \equiv 0$ then just take x and y to be zero flows.

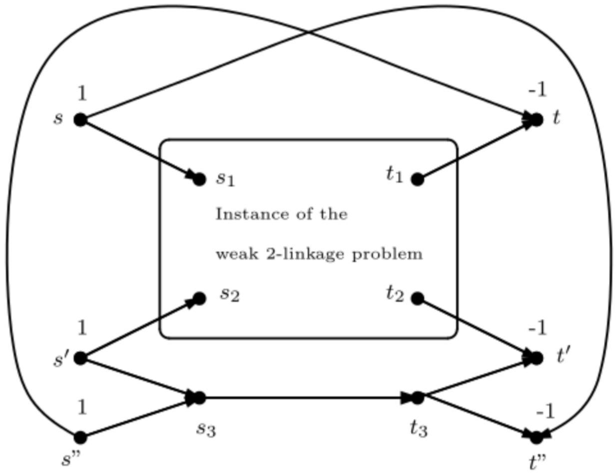

Fig. 3. The reduction from the weak 2-linkage problem in the proof of Theorem 4.6 (the non-zero balance values are indicated and all the arcs have capacity 1, except the arc s_3t_3).

It is possible to generalize this result to the problem of finding k arc-disjoint flows in a network with unit capacities. First, we generalize Lemma 4.2.

Lemma 4.4. *Let k be an integer and $G=(V,E)$ be a bipartite graph in which the degree of every vertex is a multiple of k for every vertex x. Then E can be split into sets $E_1,\dots,E_k$ such that $d_{E_i}(v)=d(v)/k$ for all $v\in V$ and all $i\in\{1,\dots,k\}$. This partition can be computed in polynomial time.*

Proof. We construct an auxiliary bipartite graph D' obtained from D by splitting every vertex into vertices of degree k. More precisely, for every vertex v of G, with $d(v)=kp$ for some integer p, we create p copies of v: $v_1,\dots,v_p$. Now, for every edge $vw\in E(G)$, we add an edge between a copy v_i of v and a copy w_j of w with the constraint that every vertex in G' must have degree at most k. At the end of the construction, every vertex of G' has degree exactly k. Now, G' is a k regular bipartite graph and it is possible to partition $E(G')$ into k sets $E'_1,\dots,E'_k$, each of them forming a matching on G' (see [7, Theorem 17.2]). Finally, for each set i, we define E_i as the set of edges of $E(G)$ corresponding to edges of E'_i. If, for a vertex v of G, we have created p copies in G', then, v will have degree p in each set E_i. ∎

Now, using Lemma 4.4, the generalization of Theorem 4.3 is straightforward, and we obtain the following.

Theorem 4.5. *Let k be an integer and $\mathcal{N}=(V,A,u\equiv 1,b)$ be a unit capacity network with a prescribed balance vector b such that $b\not\equiv 0$. There exist k arc-disjoint flows in $\mathcal{N}$, all with balance vector b if and only if $\mathcal{N}$ has a feasible flow z with balance vector $b_z\equiv kb$. Hence one can decide the existence of these flows in polynomial time.* ∎

We now return to the case of two arc-disjoint flows with the same balance vector, and study what happens if we slightly change the condition of unit capacities. Surprisingly, as soon as we allow just one arc to have capacity 2, the problem becomes NP-complete.

Theorem 4.6. *It is NP-complete to decide whether a network $\mathcal{N}=(V,A,u,b)$ with arc capacities 1 and 2 and at least one arc with capacity 2 has two arc-disjoint flows with balance vector b.*

Proof. We show how to reduce the weak 2-linkage problem to the problem above in polynomial time. Given an instance $[D=(V,A),s_1,s_2,t_1,t_2]$ of the weak 2-linkage problem (that is, we wish to decide whether D has arc-disjoint (s_1,t_1) and (s_2,t_2)-paths) we construct the network $\mathcal{N}$ as follows: first add new vertices s, s', s'', s_3 and t, t', t'', t_3 and the arcs st'', ss_1, $s's_2$, $s's_3$, $s''s_3$, $s''t$, t_1t, t_2t', t_3t', t_3t'' and s_3t_3 (see Fig. 3).

In $\mathcal{N}$, every arc has capacity 1, except s_3t_3 which has capacity 2. We fix also the balance vector $b(s)=b(s')=b(s'')=1$, $b(t)=b(t')=b(t'')=-1$ and every other vertex x of $\mathcal{N}$ satisfies $b(x)=0$.

The construction is clearly polynomial in the size of D and we will see that D has arc-disjoint (s_1,t_1) and (s_2,t_2)-paths if and only if $\mathcal{N}$ has two arc-disjoint flows with balance vector b. First, assume that D contains two arc-disjoint (s_1,t_1) and (s_2,t_2)-paths P_1 and P_2. Then we define the flow x saturating by 1 unit of flow the arcs of P_1 and the arcs ss_1 and t_1t. To complete the flow x, we fix $x(s's_3)=x(s''s_3)=x(t_3t')=x(t_3t'')=1$ and $x(s_3t_3)=2$. The other arcs receive value 0 for x. The flow y saturates by 1 unit of flow the arcs of P_2 and the arcs $s's_2$ and t_2t'. We fix also $y(s''t)=y(st'')=1$ and $y(e)=0$ for every other arcs e. Then, we check that the two flows x and y are arc-disjoint and have balance vector b.

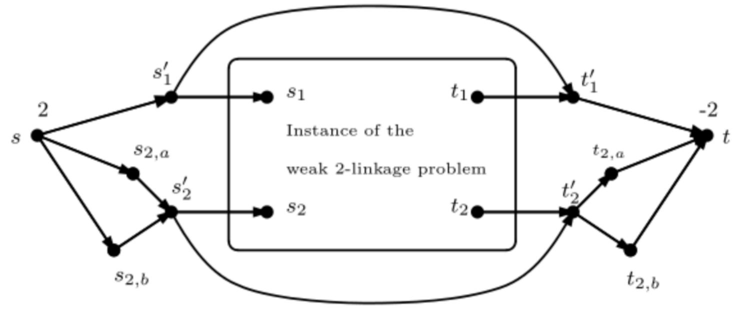

Fig. 4. The reduction from the weak 2-linkage problem in the proof of Theorem 4.7 (the non-zero balance values are indicated, in s and t, and all the arcs have capacity 1, except the arcs ss'_1 and t'_1t).

Conversely, assume that $\mathcal{N}$ admits two arc-disjoint flows x and y with balance vector b. As there are only two arcs of capacity 1 going out of s, x has to saturate one of the two and y the other, so we may assume that $x(ss_1) = 1$. Then, we have $y(st'') = 1$ and $x(t_3t'') = 1$. As there are only two arcs of capacity 1 entering in t', we have $x(t_3t') = 1$ or $y(t_3t') = 1$. In this later case, the arc s_3t_3 would carry 1 unit of both x and y, which is not possible as x and y are arc-disjoint. So, we have $y(t_2t') = 1$ and $x(t_3t') = 1$, and then, $x(s_3t_3) = 2$, $x(s''s_3) = x(s's_3) = 1$ and finally $y(s''t) = y(s's_2) = 1$ and $x(t_1t) = 1$. So, in the copy of D, we have 1 unit of flow x arriving at s_1 and leaving at t_1 and 1 unit of flow y arriving at s_2 and leaving at t_2. As these two flows are arc-disjoint, it means that we have arc-disjoint (s_1, t_1) and (s_2, t_2) paths in D. ∎

To conclude this section, we focus on the problem of computing arc-disjoint (s, t)-flows with the same initial (s) and terminal (t) vertices. If we look for flows of value 1, then we just have to compute the maximum number of arc-disjoint paths from s to t, and use one path for each flow. For flows of value 2, things become more complicated. If the network $\mathcal{N}$ only contains arcs with capacity 1, then, by Theorem 4.5, there exist k arc-disjoint (s, t)-flows of value 2 in $\mathcal{N}$ if, and only if, there exist $2k$ arc-disjoint paths from s to t (using two paths to carry one flow). If we relax a little bit the condition on the capacities and allow one arc e of capacity 2 in $\mathcal{N}$, then, we can still decide if there exist k arc-disjoint (s, t)-flows in $\mathcal{N}$ or not. Indeed, we replace e by two parallel arcs of capacity 1 and, as previously, compute the maximum number of arc-disjoint (s, t)-paths in the new network $\mathcal{N}'$. If this number is less than $2k$, the desired flows do not exist. If it is larger than $2k$ the flows exist even if we delete one copy of e. So we may assume that the maximum number of (s, t)-paths in $\mathcal{N}'$. If two of these paths use the two parallel arcs corresponding to e, then we use these two paths to carry the same flow. And we construct the other flows by taking arbitrarily two paths to carry one flow.

Finally, in this context, if we allow two arcs to have capacity 2, then the problem is no more tractable, as stated by the following theorem.

Theorem 4.7. *It is NP-complete to decide whether a network $\mathcal{N}$ with arc capacities* 1 *and* 2 *and at least two arcs with capacity* 2 *has two arc-disjoint (s, t)-flows of value* 2 *for prescribed vertices s, t of $\mathcal{N}$.*

Proof. We show how to reduce the weak 2-linkage problem to the problem above in polynomial time. Given an instance $[D = (V, A), s_1, s_2, t_1, t_2]$ of the weak 2-linkage problem (that is, we wish to decide whether D has arc-disjoint (s_1, t_1) and (s_2, t_2)-paths) we construct the network $\mathcal{N}$ as follows: first add new vertices $s, s'_1, s'_2, t, t'_1, t'_2, s_{2,a}, s_{2,b}, t_{2,a}, t_{2,b}$ and the set A' of arcs: $A' = \{ss'_1, s'_1s_1, t_1t'_1, t'_1t, s'_1t'_1, ss_{2,a}, ss_{2,b}, s_{2,a}s'_2, s_{2,b}s'_2, s'_2s_2, t_2t'_2, t'_2t_{2,a}, t'_2t_{2,b}, t_{2,a}t, t_{2,b}t, s'_2t'_2\}$. The arcs ss'_1 and t'_1t get capacity 2 and all the other arcs get capacity 1 (see Fig. 4). Clearly the construction is polynomial in the size of $[D = (V, A), s_1, s_2, t_1, t_2]$.

Assume that D has a pair of arc-disjoint $(s_1, t_1), (s_2, t_2)$-paths P_1, P_2. Then we can obtain two arc-disjoint (s, t)-flows x, y in $\mathcal{N}$ by letting x saturate all arcs in the paths $ss'_1s_1 \cup P_1 \cup t_1t'_1t$ and $ss'_1t'_1t$ (so, ss'_1 and t'_1t carry 2 units of flow), and y saturate all arcs of the arc-disjoint paths $ss_{2,a}s'_2s_2 \cup P_2 \cup t_2t'_2t_{2,a}t$ and $ss_{2,b}s'_2t'_2t_{2,b}t$. Thus, we obtain two arc-disjoint (s, t)-flows of value 2.

Conversely assume x and y are arc-disjoint (s, t)-flows of value 2 in $\mathcal{N}$. Clearly, by flow preservation, together x and y saturate all the arcs in A'. As, x and y are arc-disjoint, one of them, say x, saturates ss'_1. Then, x saturates also $s'_1t'_1$, s'_1s_1, t'_1t and $t_1t'_1$. It means that there exists an (s_1, t_1)-path in D which carries 1 unit of the flow x. Similarly, y has to saturate the remaining arcs of A' and also an (s_2, t_2)-path in D proving that $[D = (V, A), s_1, s_2, t_1, t_2]$ is a 'yes'-instance. ∎

Remark that in the above reduction, we fixed exactly two arcs with capacity 2, but if we want to have more, we can put capacity 2 on any subset of the arcs of D. Indeed, we asked for two arc-disjoint flows of value 1 in D and capacities greater than 1 on the arcs do not change the problem.

The proof above also shows that it is NP-complete to decide the existence of two arc-disjoint (s, t)-flows x, y where x has value 2 and y has value 1. In particular (the set B below corresponds to arcs of capacity 2, ss'_1 and t'_1t) the following holds.

Theorem 4.8. *It is NP-complete to decide whether a given digraph $D = (V, A)$ contains three (s, t)-paths P_1, P_2, P_3 so that P_3 is arc-disjoint from both P_1 and P_2 and P_1, P_2 may share arcs only from a specified set $B \subseteq A$ with $|B| \geqslant 2$.* ∎

5. (Arc-)disjoint (s, t)-flows in acyclic digraphs

We now turn our attention to acyclic digraphs. Motivated by the fact that the weak-k-linkage problem is polynomially solvable for fixed k in acyclic digraphs [11], we expect that we may find more polynomial instances for (arc-)disjoint flow problems when the networks in question are acyclic. We first observe that if we do not bound the values of the flows we still get NP-complete problems.

Theorem 5.1. *It is NP-complete to decide, for a given acyclic network $\mathcal{N} = (V, A, u)$ and a natural number k, whether $\mathcal{N}$ has two arc-disjoint (s, t)-flows both of value k.*

Proof. We reduce the classical NP-complete number partition problem [12, p. 223] to our problem. The number partition problem is as follows: given a set $S = \{a_1, a_2, \dots, a_p\}$ of integers such that $\sum_{i\in[p]} a_i = 2K$ for some integer K; Does there exist $J \subset \{1, 2, \dots, p\}$ such that $\sum_{j\in J} a_j = K$? Given an instance of this problem we form the network $\mathcal{N}$ by taking the union p paths sv_it, $i \in [p]$ where sv_i and v_it both have capacity a_i for $i \in [p]$. Clearly $\mathcal{N}$ has arc-disjoint (s, t)-flows x, y, each of value K if and only if there exists a subset J such that $\sum_{j\in J} a_j = K$, so the claim follows. ∎

Now, we focus on the case when k is fixed, and first, when $k = 2$. The following algorithm generalizes the algorithm for the 2-linkage problem by Perl and Shiloach [14].

Theorem 5.2. *There exists a polynomial algorithm for deciding whether an acyclic network $\mathcal{N} = (V, A, u)$ with $u_{ij} \in \{1, 2\}$ for all $ij \in A$ has vertex disjoint flows x_1 and x_2 such that x_i is an (s_i, t_i)-flow of value 2 for $i = 1, 2$, where s_1, s_2, t_1, t_2 are distinct prescribed vertices of V.*

Proof. Given an instance $[\mathcal{N} = (V, A, u), s_1, s_2, t_1, t_2]$ of the flow problem above, we first modify $\mathcal{N}$ so that $d^-(s_i) = d^+(t_j) = 0$ for $i, j \in [2]$. As we are looking for vertex disjoint flows this will not change the problem. Now form a new digraph D_{N} whose vertex set is the set of all 4-tuples of vertices u, v, p, q of V such that $\{u, v\} \cap \{w, z\} = \emptyset$ (but $u = v$ or $w = z$ is possible). The pair u, v (and w, z) are called **cousins** and the positions $(1, 2)$, $(3, 4)$ in the vector corresponding to a vertex of D_{N} are called **cousin coordinates**.

We say that a vertex p of a 4-tuple X is **minimal** (in X) if p cannot be reached in $\mathcal{N}$ from any other vertex q distinct from p in X. Remark that every 4-tuple contains at least one minimal vertex as $\mathcal{N}$ is acyclic. The arcs of D_{N} are defined as follows:

Let $X = (p_1, p_2, q_1, q_2)$ be a vertex of D_{N}.

- If p_1 is minimal in X and $p_1p'_1$ is an arc of $\mathcal{N}$ such that $p'_1 \notin \{q_1, q_2\}$ then we add the arc $(p_1, p_2, q_1, q_2) \to (p'_1, p_2, q_1, q_2)$ to $A(D_{\mathrm{N}})$. If $p_1 = p_2$ and the capacity of $p_1p'_1$ is 2, then we also add the arc $(p_1, p_2, q_1, q_2) \to (p'_1, p'_1, q_1, q_2)$ to $A(D_{\mathrm{N}})$.
- If p_2 is minimal in X and $p_2p'_2$ is an arc of $\mathcal{N}$ such that $p'_2 \notin \{q_1, q_2\}$ then we add the arc $(p_1, p_2, q_1, q_2) \to (p_1, p'_2, q_1, q_2)$ to $A(D_{\mathrm{N}})$.
- If q_1 is minimal in X and $q_1q'_1$ is an arc of $\mathcal{N}$ such that $q'_1 \notin \{p_1, p_2\}$ then we add the arc $(p_1, p_2, q_1, q_2) \to (p_1, p_2, q'_1, q_2)$ to $A(D_{\mathrm{N}})$. If $q_1 = q_2$ and the capacity of $q_1q'_1$ is 2, then we also add the arc $(p_1, p_2, q_1, q_2) \to (p_1, p_1, q'_1, q'_1)$ to $A(D_{\mathrm{N}})$.
- If q_2 is minimal in X and $q_2q'_2$ is an arc of $\mathcal{N}$ such that $q'_2 \notin \{p_1, p_2\}$ then we add the arc $(p_1, p_2, q_1, q_2) \to (p_1, p_2, q_1, q'_2)$ to $A(D_{\mathrm{N}})$.

By the flow decomposition theorem, $\mathcal{N}$ has the desired flows x_1, x_2 if and only if $\mathcal{N}$ contains paths P_1, P_2, Q_1, Q_2 where P_i is an (s_1, t_1)-path $i = 1, 2$ and Q_j is an (s_2, t_2)-path $j = 1, 2$ such that P_i and Q_j are vertex disjoint for $i, j \in \{1, 2\}$.

We claim that $\mathcal{N}$ has these paths if and only if there is a directed path from (s_1, s_1, s_2, s_2) to (t_1, t_1, t_2, t_2) in D_{N}. Suppose first that P_1, P_2, Q_1, Q_2 are paths such P_i and Q_j are vertex disjoint $i, j \in \{1, 2\}$ and such that x_1 is the union of flows of value 1 on P_1, P_2 and x_2 is the union of flows of value 1 on Q_1, Q_2. Let $\mathcal{O}$ be an acyclic ordering[7] of $\mathcal{N}$. Clearly P_1, P_2, Q_1, Q_2 move consistently with $\mathcal{O}$. Hence we can find a path from (s_1, s_1, s_2, s_2) to (t_1, t_1, t_2, t_2) in D_{N} by processing the arcs of P_1, P_2, Q_1, Q_2 one by one, always modifying (by following the corresponding arc from one of P_1, P_2, Q_1, Q_2) a coordinate of the current 4-tuple whose current vertex is not one of t_1, t_2 and which has the lowest number in $\mathcal{O}$. Observe that such a vertex is minimal in the corresponding 4-tuple. See Fig. 5 for an example. The solution in the figure corresponds for instance to the path $(s_1, s_1, s_2, s_2)(c, s_1, s_2, s_2)(c, a, s_2, s_2)(c, c, s_2, s_2)(c, c, b, b)(c, c, e, e)(d, d, e, e)(t_1, d, e, e)$

[7] An **acyclic ordering** of a digraph $D = (V, A)$ is an enumeration $v_1, v_2, \dots, v_n$ of its vertices such that every arc in A is of the form v_iv_j where $i < j$.

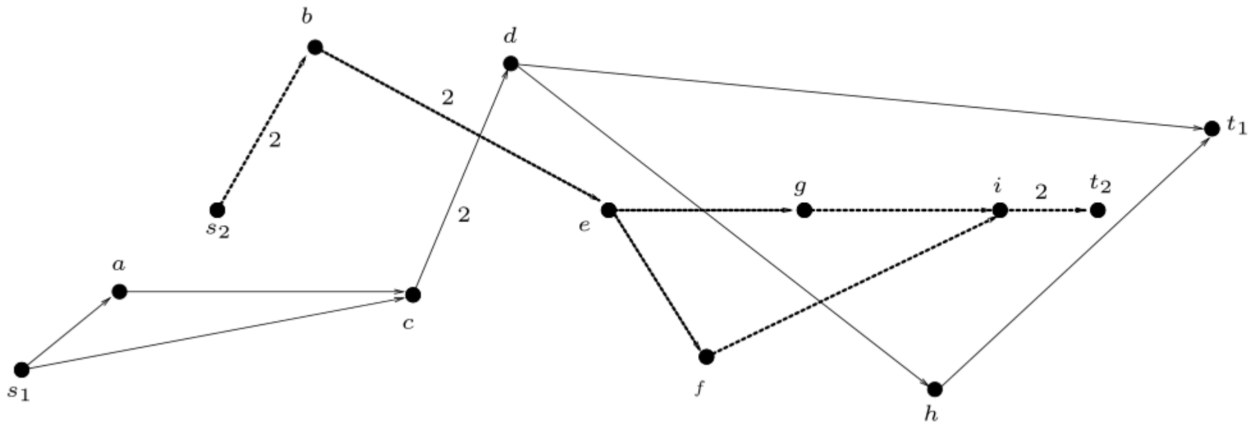

Fig. 5. A feasible solution to the flow problem where x (resp. y) follows the full (resp. dotted) arcs.

$(t_1,h,e,e)(t_1,h,e,f)(t_1,h,g,f)(t_1,h,g,i)(t_1,h,i,i)(t_1,t_1,i,i)(t_1,t_1,t_2,t_2)$ in D_{N}. Here we have followed the acyclic ordering of the vertices from left to right.

Suppose now that there is a directed path P from (s_1,s_1,s_2,s_2) to (t_1,t_1,t_2,t_2) in D_{N}. We claim that we can extract the desired paths P_1, P_2, Q_1, Q_2 as above from P. We do this simply by following the arcs of P and extending P_1, P_2, Q_1 or Q_2 in each step depending on which coordinate was changed (it is possible that two cousin coordinates changed at the same time in which case P_1, P_2 or Q_1, Q_2 share the corresponding arc of $\mathcal{N}$). Clearly this gives two (s_1,t_1)-paths P_1, P_2 and two (s_2,t_2)-paths Q_1, Q_2 so that an arc is used by both of P_1, P_2 (resp. Q_1,Q_2) only if it has capacity 2. It remains to show that P_i, Q_j are vertex disjoint. Suppose this is not the case and that some vertex v belongs to both P_i and Q_j. Without loss of generality, when we extract P_i and Q_j from P we add v to P_i first. This means that there is some legal 4-tuple containing v in the coordinate corresponding to P_i and some other vertex w which can reach v in $\mathcal{N}$ in the coordinate corresponding to Q_j. Now every vertex on $Q_j[w,v]$ can reach v in $\mathcal{N}$ so, according to the rules for arcs in D_{N}, P cannot change the coordinate corresponding to P_i until it has processed all the arcs corresponding to the arcs of $Q_j[w,v]$, but at that time we would reach a tuple containing the same vertex v in two non-cousin coordinates, contradicting the definition of $V(D_{\mathrm{N}})$[8]. ∎

Just as Fortune, Hopcroft and Wyllie could extend Perl and Shiloach's algorithm to an algorithm for k-linkage in acyclic digraphs any fixed integer k, it is not difficult to modify our proof above to show the following.

Corollary 5.3. *For every fixed integer k there exists a polynomial algorithm for deciding whether an acyclic network $\mathcal{N}=(V,A,u)$ with $u_{ij}\in\{1,2\}$ for all $ij\in A$ has vertex disjoint flows $x_1,x_2,\dots,x_k$ such that x_i is an (s_i,t_i)-flow of value 2 for $i\in[k]$, where $s_1,\dots,s_k,t_1\dots,t_k$ are distinct vertices of V.*

Similarly, we can mimic higher capacities as long as they are bounded above by some integer U. We do this by allowing up to $\min\{h,U\}$ cousin-coordinates (where h is the number of cousin coordinates in the corresponding set of cousins) to change at the same time provided that these vertices are equal in the current tuple. Similarly, in the proof above, we did not really use that we were looking for the same number of (s_i,t_i)-paths for $i=1,2$. Hence the following is the most general statement that still can be shown using analogous arguments to those above.

Corollary 5.4. *For every fixed collection of integers $k,\alpha_1,\alpha_2,\dots,\alpha_k,U$ there exists a polynomial algorithm for deciding whether an acyclic network $\mathcal{N}=(V,A,u)$ with $u_{ij}\in\{1,2,\dots U\}$ for all $ij\in A$ has vertex disjoint flows $x_1,x_2,\dots,x_k$ such that x_i is an (s_i,t_i)-flow of value α_i for $i\in[k]$, where $s_1,\dots,s_k,t_1\dots,t_k$ are distinct vertices of V.*

The following shows that the algorithm of Theorem 5.1 can be translated to an algorithm for arc-disjoint rather than vertex-disjoint flows. Similarly, each of the corollaries above have an arc-disjoint analogue which we leave to the interested reader.

Theorem 5.5. *There exists a polynomial algorithm for deciding whether an acyclic network $\mathcal{N}=(V,A,u)$ with $u_{ij}\in\{1,2\}$ for all $ij\in A$ has arc-disjoint flows x_1 and x_2 such that x_i is an (s_i,t_i)-flow of value 2 for $i=1,2$, where possibly $s_1=s_2$ or $t_1=t_2$. In particular, we can check in polynomial time whether an acyclic network $\mathcal{N}=(V,A,u)$ with $u_{ij}\in\{1,2\}$ for all $ij\in A$ has arc-disjoint (s,t)-flows x, y of value 2 each.*

[8] This part of the proof is identical to the classical argument by Perl and Shiloach and Fortune, Hopcroft and Wyllie.

Proof. Given $\mathcal{N} = (V, A, u)$ we construct the network $\mathcal{N}'$ as follows: Replace each vertex $v \in V$ by $d^-(v)$ vertices $I_v = \{v_1^-, \ldots, v_{d^-(v)}^-\}$ and $d^+(v)$ vertices $O_v = \{v_1^+, \ldots, v_{d^-(v)}^+\}$ and also fix an ordering of the in- and out-neighbours around each vertex. For each arc vw of $\mathcal{N}$, if w is the ith out-neighbour of v and v the jth in-neighbour of w, then add the arc $v_i^+ w_j^-$ to $\mathcal{N}'$ with capacity u_{vw}. Finally, for every vertex v, add all possible arcs from I_v to O_v and set the capacities of these arcs as follows: If p is the ith in-neighbour of v and q is the jth out-neighbour of v then the arc $v_i^- v_j^+$ gets capacity the minimum of the capacities of the arc from $v's$ ith in-neighbour to v and the capacity of the arc from v to its jth out-neighbour. Now it is easy to check that $\mathcal{N}'$ has vertex disjoint flows x'^1 and x'^2 such that x'^i is an (s_i, t_i)-flow of value 2 for $i = 1, 2$ if and only if $\mathcal{N}$ has arc-disjoint flows x_1 and x_2 such that x_i is an (s_i, t_i)-flow of value 2 for $i = 1, 2$. It is also easy to handle the case where $s_1 = s_2$ or $t_1 = t_2$ by adding a copy of such a vertex to $\mathcal{N}'$. ∎

The following results, which we state only for disjoint flows of value 2, hold for all the other variants discussed above also. The proofs of these use the standard methods for flows with cost function, see [3, Section 4.10]. We leave the easy proofs to the reader (arcs with 2 units of flow get twice the cost of the original arc).

Theorem 5.6. *There exists a polynomial algorithm for finding, in an acyclic network $\mathcal{N} = (V, A, u, c)$ with $u_{ij} \in \{1, 2\}$ for all $ij \in A$ and cost function $c : A \to \mathbf{R}$ on the arcs, a pair vertex disjoint flows x_1 and x_2 such that x_i is an (s_i, t_i)-flow of value 2 for $i = 1, 2$, where s_1, s_2, t_1, t_2 are distinct vertices of V and the total cost of these flows is minimum among all such solutions (the value will be ∞ if there is no such pair of flows).*

Theorem 5.7. *There exists a polynomial algorithm for finding in an acyclic network $\mathcal{N} = (V, A, u, c)$ with $u_{ij} \in \{1, 2\}$ for all $ij \in A$ and cost $c : A \to \mathbf{R}$ a pair vertex disjoint flows x_1 and x_2 such that x_i is an (s_i, t_i)-flow of value 2 for $i = 1, 2$, where s_1, s_2, t_1, t_2 are distinct vertices of V and the total cost of arcs used by these flows is minimum among all such flows (the value will be ∞ if there is no such pair of flows).*

6. Cycle factors with all cycles odd or all cycles even

We saw in Theorem 2.3 that deciding whether a digraph has a spanning Eulerian subdigraph in which all connected components are even is an NP-complete problem. A cycle factor is a special kind of spanning Eulerian subdigraph and hence it is natural to ask about the complexity of deciding whether a digraph has a cycle factor all of whose cycles are even. A cycle factor $\mathcal{C}$ of a digraph is **even** (resp. **odd**) if all the cycles of $\mathcal{C}$ have even (resp. odd) length. The *even cycle factor problem* (resp. the *odd cycle factor problem*) consists in deciding whether or not a given digraph contains an even (resp. odd) cycle factor.

Lemma 6.1. *It is NP-complete to decide whether or not a digraph has an even cycle factor (resp. an odd cycle factor). This also holds for digraphs without 2-cycles (oriented graphs).*

Proof. First we reduce the 2-linkage problem to the even cycle factor problem in polynomial time. Given an instance $[D = (V, A), s_1, s_2, t_1, t_2]$ of the 2-linkage problem (that is, we want to decide whether D contains vertex-disjoint (s_1, t_1)- and (s_2, t_2)-paths), we first modify it so that $d^-(s_i) = d^+(t_j) = 0$ for $i, j \in [2]$. This does not change the problem. Now we construct the digraph D' as follows: first replace each vertex v of D different from s_1, s_2, t_1 and t_2 by two vertices v^- and v^+ and each arc uv, with u and v different from s_1, s_2, t_1 and t_2, by an arc u^+v^-. We replace also each arc xv (resp. vx) with $x \in \{s_1, s_2, t_1, t_2\}$ by an arc xv^- (resp. v^+x). Now, for every vertex $v \in V \setminus \{s_1, s_2, t_1, t_2\}$, between v^- and v^+, we add a copy of the gadget H_v, defined in Fig. 6. Finally, we add the two vertices u_1 and u_2 and the arcs t_1u_1, t_2u_2, u_1s_2 and u_2s_1. Let us see that D has vertex-disjoint (s_1, t_1) and (s_2, t_2)-paths if and only if D' has an even cycle factor. First, suppose that D' has an even cycle factor $\mathcal{C}$ and let C be the cycle of $\mathcal{C}$ which contains s_1. To enter in s_1, the cycle C has to contain the path $t_2u_2s_1$. Observe that when C enters a gadget H_v through the vertex v^-, then C has to contain the path $v^-a_vc_vd_vb_vv^+$ and then to go out of H_v at v^+. So after visiting s_1 C covers some gadgets H_v and eventually goes to t_1 or t_2. If C goes directly to t_2, then we have totally described C, but it cannot be, because in this case C would have odd length (as each H_v contains an even number of vertices). So, C has to go through t_1, implying that it contains the subpath $t_1u_1s_2$, then covers some other gadgets H_v and finally ends in t_2. This implies that, back in D, the cycle C corresponds to vertex-disjoint (s_1, t_1) and (s_2, t_2)-paths. Conversely, if D contains two vertex-disjoint (s_1, t_1) and (s_2, t_2)-paths P_1 and P_2, then we form an even cycle C in D' by replacing each vertex $v \notin \{s_1, s_2, t_1, t_2\}$ on each path by the path $v^-a_vc_vd_vb_vv^+$ of the corresponding gadget H_v, and the paths $t_2u_2s_1$ and $t_1u_1s_2$ to close C. Finally, for all $v \notin V(P_1) \cup V(P_2)$ we add the cycle $v^-a_vb_vv^+c_vd_v$ to obtain an even cycle factor in D'. This concludes the proof of equivalence between the instance $[D = (V, A), s_1, s_2, t_1, t_2]$ of the 2-linkage problem and the instance D' of the even cycle factor problem.

Remark that if we want a smaller reduction, for each vertex v, we can use a 2-cycle on v^+, v^- instead of the gadget H_v, but it forces the cycle factor to contain digons.

We can also reduce the 2-linkage problem to the odd cycle factor problem in polynomial time. The reduction is quite similar. Given an instance D of the 2-linkage problem (for which we may assume that $d^-(s_i) = d^+(t_j) = 0$ for $i, j \in [2]$) we construct the digraph D'' which is the same as D' previously built, except that we uncross the paths $t_1u_1s_2$ and $t_2u_2s_1$.

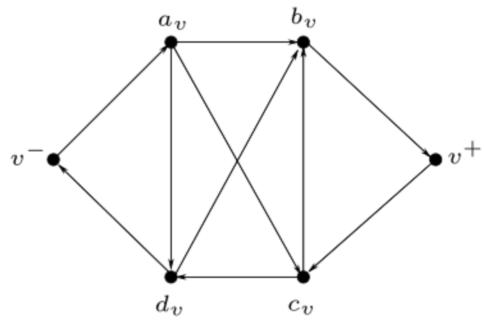

Fig. 6. The gadget H_v.

Namely, we remove the arcs u_1s_2 and u_2s_1 from D' and add the arcs u_1s_1 and u_2s_2 to form D''. Now, we argue that D has vertex-disjoint (s_1,t_1) and (s_2,t_2)-paths if and only if D'' has an odd cycle factor. If D'' has an odd cycle factor $\mathcal{C}$, let C be the cycle of $\mathcal{C}$ containing s_1. By the above arguments, C starts in s_1, traverses some gadgets H_v and goes to t_1 or t_2. If C goes to t_2, it has to contain the subpaths $t_2u_2s_2$ traverse other gadgets H_v and then end by the subpath $t_1u_1s_1$ but then C would have even length. So, C goes directly to t_1 and ends in the subpath $t_1u_1s_1$. Hence back in D, C corresponds to an (s_1,t_1)-path. Similarly, the cycle of $\mathcal{C}$ containing s_2 corresponds to an (s_2,t_2)-path in D, and D has vertex-disjoint (s_1,t_1) and (s_2,t_2)-paths. Conversely, if D contains two vertex-disjoint (s_1,t_1) and (s_2,t_2)-paths P_1 and P_2, we form two disjoint odd cycles in D'' as we did above and add, for each $v \notin V(P_1) \cup V(P_2)$ the cycles $v^-a_vd_v$ and $v^+c_vb_v$ to obtain an odd cycle factor in D''. □

7. Concluding remarks

There are many more questions to study which are related to the questions which we dealt with in the paper. Some of these are basic questions about flows in networks. The following problem is easy to solve for $k=1$ using a modification of Dijkstra's algorithm to find a maximum capacity (s,t)-path (this idea was already used in the classical paper by Edmonds and Karp [10]). Already for $k=2$ the problem becomes NP-complete.

Theorem 7.1. *[2] For every fixed natural number $k \geqslant 2$ it is NP-hard to find, for a given network $\mathcal{N}$ with source s and sink t, the maximum value of an (s,t)-flow which can be decomposed into at most k paths in $\mathcal{N}$.*

The following seems closely related. Again we can decide in polynomial time whether $p=1$.

Problem 7.2. What is the complexity of the following problem: Given network $\mathcal{N}$ with source s and sink t which has an (s,t)-flow of value k; find the minimum natural number p so that $\mathcal{N}$ has an (s,t)-flow of value k which can be decomposed (via flow decomposition) into p (s,t)-paths and some cycles?

We can also ask for the complexity of finding a decomposition of a prescribed flow into as few paths (and some cycles) as possible.

Problem 7.3. Is there a polynomial algorithm for finding, in a given network $\mathcal{N}$ and a given (s,t)-flow x of value k in $\mathcal{N}$, a decomposition of x into (s,t)-paths and cycles which uses the minimum possible number of (s,t)-paths?

Problem 7.4. Determine the minimum function $f(n)$ so that there is a polynomial algorithm for deciding the existence of two arc-disjoint branching flows in a network $\mathcal{N}=(V,A,u)$ where $|V|=n$ and $u_{ij} \in [f(n)]$ for all arcs $ij \in A$.

By the results in Section 3 we have $2 < f(n) \leqslant n-1$.

Our method in the proof of Theorem 5.2 can neither be extended to two disjoint flows of arbitrary high values nor to arbitrarily many disjoint flows of value 2 (because this would mean that the tuples could have size $O(|V|)$). In particular the following problem which fits in the framework[9] is open.

Problem 7.5. Is there a polynomial algorithm for deciding whether a digraph D has three arc-disjoint cycle factors $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$ such that $\mathcal{F}_1$, $\mathcal{F}_3$ and $\mathcal{F}_2$, $\mathcal{F}_3$ are arc-disjoint and $\mathcal{F}_1$, $\mathcal{F}_2$ share at most k arcs?

[9] The problem of deciding the existence of a cycle factor in a digraph on n vertices can be formulated as that of checking for an (s,t)-flow of value n in an acyclic network (see e.g. Section 4.11.3 [3]). The arcs where F_1 and F_2 may overlap correspond to arcs of capacity 2 that may be used by the circulation formed by the union of the two cycle factors.

For $k = 0$ this can be solved by checking, via a maxflow algorithm, whether D contains a spanning 3-regular digraph.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows, Prentice Hall, Englewood Cliffs, NJ, 1993.
[2] G. Baier, E. Köhler, M. Skutella, The *k*-splittable flow problem, Algorithmica 42 (2005) 231–238.
[3] J. Bang-Jensen, G. Gutin, Digraphs: Theory, Algorithms and Applications, 2nd edition, Springer-Verlag, London, 2009.
[4] J. Bang-Jensen, M. Kriesell, A. Madalloni, S. Simonsen, Vertex-disjoint directed and undirected cycles in general digraphs, J. Comb. Theory, Ser. B (2014), in press.
[5] J. Bang-Jensen, A. Yeo, Arc-disjoint spanning sub(di)graphs in digraphs, Theor. Comput. Sci. 438 (2012).
[6] A. Bernáth, Z. Király, Finding edge-disjoint subgraphs in graphs, Egres Quick Proof, 2010-04.
[7] J.A. Bondy, U.S.R. Murty, Graph Theory, Graduate Texts in Mathematics, vol. 244, Springer-Verlag, Berlin, 2008.
[8] P.A. Catlin, Supereulerian graphs, collapsible graphs and four-cycles, Congr. Numer. 58 (1987) 233–246.
[9] J. Edmonds, Edge-disjoint branchings, in: B. Rustin (Ed.), Combinatorial Algorithms, Academic Press, 1973, pp. 91–96.
[10] J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, J. Assoc. Comput. Mach. 19 (1972) 248–264.
[11] S. Fortune, S. Hopcroft, J. Wyllie, The directed subgraph homeomorphism problem, Theor. Comput. Sci. 10 (1980) 111–121.
[12] M.R. Garey, D.S. Johnson, Computers and Intractability, W.H. Freeman, San Francisco, 1979.
[13] L. Lovász, On two minimax theorems in graph theory, J. Comb. Theory, Ser. B 21 (1976) 96–103.
[14] Y. Perl, Y. Shiloach, Finding two disjoint paths between two pairs of vertices in a graph, J. Assoc. Comput. Mach. 21 (1978) 1–9.
[15] W.R. Pulleyblank, A note on graphs spanned by Eulerian graphs, J. Graph Theory 3 (1979) 309–310.