Operations Research Letters 48 (2019) 18–23

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

The complexity of computing a robust flow

Yann Disser [a], Jannik Matuschke [b,*]

[a] Department of Mathematics, Graduate School CE, TU Darmstadt, Germany
[b] Research Center for Operations Management, KU Leuven, Belgium

ARTICLE INFO

Article history:
Received 6 May 2019
Received in revised form 3 September 2019
Accepted 27 October 2019
Available online 2 November 2019

Keywords:
Complexity theory
Network flows
Network interdiction
Robust optimization

ABSTRACT

The Maximum Robust Flow problem asks for a flow on the paths of a network maximizing the guaranteed amount of flow surviving the removal of any k arcs. We point out a flaw in a previous publication that claimed NP-hardness for this problem when $k = 2$. For the case that k is part of the input, we present a new hardness proof. We also discuss the complexity of the integral version of the problem.

1. Introduction

Network flows are an important tool for modeling vital network services, such as transportation, communication, or energy transmission [2]. In many of these applications, the flow is subjected to uncertainties such as failures of links in the network infrastructure. This motivates the study of *robust optimization* versions of network flows, which offer room to anticipate and counteract such failures [7]. A fundamental optimization problem within this framework is to find a flow that maximizes the amount of surviving flow after it is affected by a worst-case failure of k links in the network for some given number k.

The complexity of this problem, which is also known as Maximum Robust Flow, appeared to have been settled a decade ago: Aneja et al. [3] showed that the problem can be solved efficiently when $k = 1$, while an article by Du and Chandrasekaran [9] established that the problem is NP-hard for any constant value of k larger than 1.

In this paper, we point out an error in the latter result. In its stead, we give a new hardness proof which, however, requires the number k to be a non-constant part of the input. We further show that computing optimal integral solutions is already NP-hard for $k = 2$ (whereas for $k = 1$, an efficient algorithm is known). On the positive side, we give a polynomial time algorithm for computing an optimal integral solution for arbitrary k when capacities are small. Before we discuss these results in detail, we give a formal definition of the problem and discuss related literature.

1.1. Problem definition

We are given a directed graph $G = (V, E)$ with source s, sink t, capacities $u \in \mathbb{Z}_+^E$, and an integer k, specifying the number of possible link failures. Let $\mathcal{P}$ denote the set of s-t-paths in G and let $\mathcal{S} := \{S \subseteq E \ : \ |S| = k\}$. An *$s$-$t$-flow* is a vector $x \in \mathbb{R}_+^{\mathcal{P}}$ respecting the capacity constraints $\sum_{P:e\in P} x(P) \leq u(e)$ for all $e \in E$. The goal is to find an s-t-flow x that maximizes the *robust flow value*

$$\mathrm{val_r}(x) := \sum_{P\in\mathcal{P}} x(P) - \max_{S\in\mathcal{S}} \sum_{P\in\mathcal{P}:P\cap S\neq\emptyset} x(P),$$

i.e., the amount of remaining flow after failure of any set of k arcs.

1.2. Related work

Aneja et al. [3] were the first to investigate Maximum Robust Flow. They showed that if $k = 1$, the problem can be solved in polynomial time by solving a parametric linear program. In fact, their LP yields a flow x that simultaneously maximizes $\mathrm{val_r}(x)$ and the nominal flow value $\mathrm{val}(x) := \sum_{P\in\mathcal{P}} x(P)$. They also show that a maximum *integral* robust flow can be found in polynomial time for $k = 1$, even though its value might be strictly lower than that of the optimal fractional solution. Following up on this work, Du and Chandrasekaran [9] investigated the problem for values of k larger than 1. They presented a hardness proof for Maximum Robust Flow with $k = 2$. Unfortunately, however, this proof is incorrect. We explain this error in detail in Section 2.3.

Because of the presumed hardness of the problem, later work focused on approximation algorithms. Bertsimas et al. [6] use a variation of the parametric LP to obtain an approximation algorithm for Maximum Robust Flow whose factor depends on the fraction of flow lost through the failure. More recently, Bertsimas

* Corresponding author.
E-mail address: jannik.matuschke@kuleuven.be (J. Matuschke).

https://doi.org/10.1016/j.orl.2019.10.012
0167-6377/© 2019 Elsevier B.V. All rights reserved.

et al. [5] gave an alternative analysis of the same algorithm, establishing an approximation factor of $1+(k/2)^2/(k+1)$. Another related concept is *k-route flows* introduced by Aggarwal and Orlin [1]. A k-route flow is a conic combination of elementary flows, each sending flow uniformly along k disjoint paths. This structure ensures that the failure of any arc can only destroy a $1/k$ fraction of the total flow. Baffier et al. [4] observed that computing a maximum $(k+1)$-route flow yields a $(k+1)$-approximation for Maximum Robust Flow. They also performed computational experiments, which indicated that a heuristic improvement of the aforementioned algorithm computes optimal or near-optimal solutions on a large variety of instances.

Several alternative robustness models for flows have been proposed in different application contexts. Taking a less conservative approach, Bertsimas et al. [6] and Matuschke et al. [13] proposed different models of flows that can be rerouted after failures occur. Matuschke et al. [14] investigated variants of robust flows in which an adversary can target individual flow paths and the network can be fortified against such attacks. Gottschalk et al. [11] devised a robust variant of flows over time in which transit times are uncertain.

Robust flows can be seen as a dual version of *network flow interdiction*, where the task is to find a subset of k arcs whose removal minimizes the maximum flow value in the remaining network. Wood [15] proved that this problem is strongly *NP*-hard. The reduction presented in Section 3 also exploits the fact that interdiction is *NP*-hard, but the construction is considerably more involved in order to couple network flow and interdiction decisions in the correct way. For an overview of results on network flow interdiction, see the recent article by Chestnut and Zenklusen [8] on the approximability of the problem.

1.3. Results and structure of this paper

In Section 2, we give the background necessary to understand the reduction from [9] and the reason why it does not imply hardness for Maximum Robust Flow with $k = 2$.

In Section 3, we then give a new reduction that establishes *NP*-hardness for Maximum Robust Flow when k is an arbitrarily large number given in the input. Our reduction works even when the number of paths in the graph is polynomial in the size of the network and only two different capacity values occur (capacity 1 and a capacity that is large but polynomial in the size of the network). We also point out that the problem becomes easy for the case that all capacities are equal.

In Section 4, we show that it is *NP*-hard to compute an optimal *integral* solution for $k = 2$. Note that this is in contrast to the case $k = 1$, where the optimal integral solution can be computed efficiently [3]. While *NP*-hardness for the integral case even holds when capacities are bounded by 3, we show that the problem can be solved efficiently when capacities are bounded by 2, even for arbitrary values of k.

2. Background

The hardness result from [9] is based on an LP formulation of Maximum Robust Flow and the equivalence of optimization and separation, which we shortly recapitulate in this section.

2.1. LP formulation

For our further discussion of Maximum Robust Flow, the following linear programming formulation of the problem will be useful:

$$
\begin{array}{llrll}
\text{[P]} & \max & \sum_{P \in \mathcal{P}} x(P) - \lambda & & \\
& \text{s.t.} & \sum_{P : e \in P} x(P) & \le u(e) & \forall\, e \in E \\
& & \sum_{P : P \cap S \neq \emptyset} x(P) - \lambda & \le 0 & \forall\, S \in \mathcal{S} \\
& & x(P) & \ge 0 & \forall\, P \in \mathcal{P}
\end{array}
$$

Note that $\lambda = \max_{S \in \mathcal{S}} \sum_{P \in \mathcal{P} : P \cap S \neq \emptyset} x(P)$ in any optimal solution to [P], i.e., λ represents the amount of flow lost in a worst-case failure scenario for flow x. We also consider the dual of [P]:

$$
\begin{array}{llrll}
\text{[D]} & \min & \sum_{e \in E} u(e) y(e) & & \\
& \text{s.t.} & \sum_{e \in P} y(e) + \sum_{S : P \cap S \neq \emptyset} z(S) & \ge 1 & \forall\, P \in \mathcal{P} \\
& & \sum_{S \in \mathcal{S}} z(S) & = 1 & \\
& & y(e) & \ge 0 & \forall\, e \in E \\
& & z(S) & \ge 0 & \forall\, S \in \mathcal{S}
\end{array}
$$

Note that the number of s-t-paths in G and hence the number of variables of [P] can be exponential in $|E|$. On the other hand, the number of variables of [D] is $|E| + \binom{|E|}{k}$, which is polynomial in $|E|$ for constant values of k. In such a situation, a standard approach is to solve the dual via its *separation problem*, which is described in the next section.

2.2. Equivalence of optimization and separation

Let $Q \subseteq \mathbb{R}^n$ be a rational polyhedron. By a classic result of Grötschel et al. [12], optimizing arbitrary linear objectives over Q is polynomially equivalent to finding out whether a given point is in Q and finding a hyperplane separating the point from Q if not. We give a formal statement of this result below.

Separation(Q)

Input: a vector $y \in \mathbb{R}^n$

Task: Assert that $y \in Q$, or find a separating hyperplane, i.e., a vector $d \in \mathbb{R}^n$ such that $d^T x < d^T y$ for all $x \in Q$.

Optimization(Q)

Input: a vector $c \in \mathbb{R}^n$

Task: Either assert that $Q = \emptyset$, or find $x, d \in \mathbb{R}^n$ such that $c^T d > 0$ and $x + \alpha d \in Q$ for all $\alpha \ge 0$, or find $x \in Q$ maximizing $c^T x$.

Theorem 1 (*Grötschel et al. [12, Theorem 6.4.9]*). *The optimization problem for Q can be solved in oracle-polynomial time given an oracle for the separation problem for Q, and vice versa.*

2.3. Dual separation for maximum robust flow

Let Q be the set of feasible solutions to the dual program [D], i.e.,

$$
Q := \Big\{ (y, z) \in \mathbb{R}^{E \times \mathcal{S}} \ : \ \sum_{S \in \mathcal{S}} z(S) = 1, \\
\sum_{e \in P} y(e) + \sum_{S : P \cap S \neq \emptyset} z(S) \ge 1 \ \forall\, P \in \mathcal{P} \Big\}.
$$

In the separation problem for Q, we are given $(y, z) \in \mathbb{R}^{E \times \mathcal{S}}$ and have to decide whether $(y, z) \in Q$. Since checking whether $\sum_{S \in \mathcal{S}} z(S) = 1$ can be done in polynomial time for constant values of k, the separation problem is polynomial-time equivalent to finding a path P such that $\sum_{e\in P} y(e) + \sum_{S:P\cap S\neq\emptyset} z(S) < 1$ or deciding that no such path is exists.

Du and Chandrasekaran [9] showed that the problem SEPARATION(Q) is *NP*-hard, even when $k = 2$. They concluded that by the equivalence of optimization and separation, solving [D] and hence solving [P] is NP-hard. However, this implication is not correct. It is true that the hardness of SEPARATION(Q) implies that also OPTIMIZATION(Q) is *NP*-hard. However, [D] is only a special case of OPTIMIZATION(Q): The objective function of [D] is not an arbitrary vector in $\mathbb{R}^{E\times\mathcal{S}}$, but it is restricted to those objective functions where all coefficients corresponding to the z-variables are 0. Hence, a polynomial time algorithm for MAXIMUM ROBUST FLOW does not necessarily imply a polynomial time algorithm for OPTIMIZATION(Q) and therefore the hardness of the latter does not transfer to the former.

It is worthwhile to note that the instances of MAXIMUM ROBUST FLOW with $k = 2$ constructed in the reduction from [9] contain an s-t-cut of cardinality 2. For such instances, every s-t-flow has a robust value 0. Hence MAXIMUM ROBUST FLOW can be solved in polynomial time for these instances. This shows that the invalid implication is not just a mere technicality that could easily be fixed. If MAXIMUM ROBUST FLOW for $k = 2$ is indeed *NP*-hard, a reduction that shows this would need to construct considerably more involved instances of the problem.

3. Robust flows with large number of failing arcs

Theorem 2. MAXIMUM ROBUST FLOW *is strongly NP-hard, even when restricted to instances where the number of paths is polynomial in the size of the graph.*

Proof. We show this by a reduction from CLIQUE: Given a graph $G' = (V', E')$ and $k' \in \mathbb{Z}_+$, is there a clique of size k' in G'? We will construct an instance of MAXIMUM ROBUST FLOW consisting of a graph $G = (V, E)$, source s, sink t, capacities $u : E \to \mathbb{Q}_+ \cup \{\infty\}$, and $k \in \mathbb{Z}_+$ from the CLIQUE instance (at the end of the proof, we show how to obtain an equivalent instance with finite and integral capacities). Let

$$\ell := |V'| + 2|E'|, \qquad k := k'\ell + (|V'| - k') + 2|E'|,$$
$$\varepsilon := \frac{1}{\ell}, \qquad M := (1+\varepsilon)k.$$

For every vertex $v \in V'$ we introduce a node a_v and two additional groups of ℓ nodes each, $A_v = \{a_{v,1}, \ldots, a_{v,\ell}\}$ and $B_v = \{b_{v,1}, \ldots, b_{v,\ell}\}$. We connect a_v to every node in B_v by an arc of capacity M, and we also connect each node $a_{v,i}$ to $b_{v,i}$ by an arc of capacity 1. For every edge $e = \{u, v\} \in E'$ we introduce two nodes a'_e, a''_e and arcs $(a'_e, b_{u,i}), (a''_e, b_{u,i}), (a'_e, b_{v,i}), (a''_e, b_{v,i})$ for $i \in \{1, \ldots, \ell\}$, each of capacity M. We denote

$$A := \bigcup_{v\in V'} (\{a_v\} \cup A_v) \cup \bigcup_{e\in E'} \{a'_e, a''_e\} \text{ and } B := \bigcup_{v\in V'} B_v.$$

We also introduce a source s and a sink t and arcs (s, a) for every $a \in A$ and (b, t) for every $b \in B$, all of infinite capacity. We then add k parallel s-t-arcs $e_1, \ldots, e_k$. Defining $h := 2 \cdot \binom{k'}{2} - 2$, we set the capacity of $e_1, \ldots, e_h$ to $1 + \varepsilon$ and the capacity of $e_{h+1}, \ldots, e_k$ to 1. We finally add two additional nodes v', v'', together with two s-v'-arcs e'_1, e'_2, two v''-t-arcs e''_1, e''_2, and arcs $(s, v''), (v', t), (v', v'')$. We set the capacities $u(e'_1) = u(e''_1) = 1$, $u(e'_2) = u(e''_2) = u(v', v'') = \varepsilon$ and $u(s, v'') = u(v', t) = 1 + \varepsilon$. We let E_H denote the arcs in the subgraph H induced by the node set $\{s, v', v'', t\}$. The complete construction is depicted in Fig. 1.

We now prove the following lemma, which implies Theorem 2. For convenience we will use the notation $x(e) := \sum_{P:e\in P} x(P)$ for the total flow through an arc e.

Lemma 3. *Let (x^*, λ^*) be an optimal solution to* MAXIMUM ROBUST FLOW. *Then there is a clique of size k' in G' if and only if $x^*(v', v'') > 0$.*

Proof. In order to prove Lemma 3 we first observe that, without loss of generality, we can assume that all arcs in $E \cap (A \times B)$ and the arcs $e_1, \ldots, e_k$ are saturated by x^*: If any of these arcs is not saturated, we can increase the flow along the unique path containing that arc and increase λ^* by the same value, not decreasing the value of the solution and not changing the flow on (v', v'').

Consider the set $F := \{e_1, \ldots, e_k, (s, v''), (v', t)\}$ and define

$$f_{x^*}(r) := \max\left\{\sum_{e\in F'} x^*(e) \,:\, F' \subseteq F,\ |F'| \leq r\right\}$$

for $r \in \mathbb{N}$. To prove Lemma 3, we establish the following result that relates the value of λ^* to the maximum number of edges induced by a k'-vertex subgraph of G'.

Lemma 4. *Let $h^* := \max\{|E'[U]| \,:\, U \subseteq V',\ |U| \leq k'\}$. Then,* $\lambda^* = (|V'| + 4|E'|)\ell M + k'\ell + f_{x^*}(2h^*)$.

Proof. Let $S \in \mathcal{S}$ be such that $\sum_{P\in\mathcal{P}:S\cap P\neq\emptyset} x^*(P) = \lambda^*$. We first observe that we can assume $S \cap (A \times B) = \emptyset$ without loss of generality: If S contains an arc $(a, b) \in A \times B$, we can replace it by either of the arcs (s, a) or (b, t), each of which intersects the unique s-t-path containing (a, b).

Now define

$$U := \{v \in V' \,:\, (b, t) \in S \ \forall\, b \in B_v\}.$$

Note that $|U| \leq \lfloor k/\ell \rfloor \leq k'$ by choice of k and ℓ. Furthermore, note that $x^*(P) = M$ for exactly $(|V'| + 4|E'|)\ell$ paths $P \in \mathcal{P}$ by our earlier assumption that arcs in $E \cap (A \times B)$ are fully saturated. Also, by choice of M and since every other path carries at most $1 + \varepsilon$ units of flow, the only possibility to destroy at least $(|V'| + 4|E'|)\ell M$ units of flow is for S to intersect all these paths, and by maximality of λ^*, this must indeed be the case. Therefore, we can assume that for every $v \in V'$, either $v \in U$ or $\{(s, a_v)\} \cup \{(s, a'_e), (s, a''_e) \,:\, e \in \delta(v)\} \subseteq S$. This implies that U already determines a subset S_U of

$$k_U := \ell|U| + |V'| - |U| + 2(|E'| - |E'[U]|)$$

arcs in S, destroying a flow of $(|V'| + 4|E'|)\ell M + |U|\ell$ units. The remaining $k - k_U$ arcs in S can destroy an additional flow of at most $f_{x^*}(k - k_U)$, as no arc in $E \setminus F$ carries more than 1 unit of flow after destruction of the flow paths of value M and there are at least k arcs in F with flow value at least 1. Furthermore observe that $f_{x^*}(r' + r'') \leq f_{x^*}(r') + (1+\varepsilon)r''$ as none of the arcs in F carries more than $1 + \varepsilon$ units of flow. We deduce that

$$\begin{aligned}
\lambda^* &\leq (|V'| + 4|E'|)\ell M + |U|\ell + f_{x^*}(k - k_U) \\
&= (|V'| + 4|E'|)\ell M + |U|\ell \\
&\quad + f_{x^*}((k' - |U|)(\ell - 1) + 2|E'[U]|) \\
&\leq (|V'| + 4|E'|)\ell M + |U|\ell \\
&\quad + f_{x^*}(2|E'[U]|) + (1+\varepsilon)(k' - |U|)(\ell - 1) \\
&= (|V'| + 4|E'|)\ell M + k'\ell \\
&\quad + (k' - |U|)(\underbrace{\varepsilon(\ell - 1) - 1}_{\leq 0}) + f_{x^*}(2\underbrace{|E'[U]|}_{\leq h^*}) \\
&\leq (|V'| + 4|E'|)\ell M + k'\ell + f_{x^*}(2h^*).
\end{aligned}$$

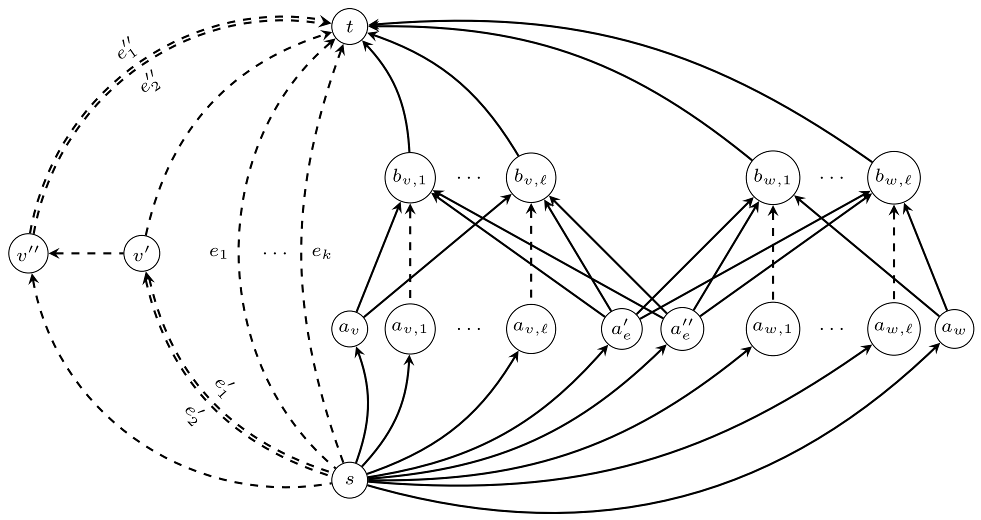

Fig. 1. Construction of the reduction from CLIQUE for two vertices v, w and an edge $e = \{v, w\}$. Solid arcs have a "large" capacity (i.e., $u(e) \in \{M, \infty\}$), dashed arcs have a "small" capacity (i.e., $u(e) \in \{\varepsilon, 1, 1+\varepsilon\}$).

Now let $U^* \subseteq V'$ be such that $|U^*| = k'$ and $|E'[U^*]| = h^*$ and let $F^* \subseteq F$ be such that $|F^*| = 2h^*$ and $\sum_{e\in F^*} x^*(e) = f_{x^*}(2h^*)$. Consider the set

$$S^* := \bigcup_{v\in U^*} B_v \cup \{a_v : v \in V' \setminus U^*\} \cup \{a_e', a_e'' : e \notin E'[U^*]\} \cup F^*$$

and observe that $|S^*| \le k$. Note that $\sum_{P:P\cap F^*\neq\emptyset} x^*(P) = \sum_{e\in F^*} x^*(e)$ because no s-t-path contains more than one arc from $F^* \subseteq F$. Therefore

$$\lambda^* \ge \sum_{P:P\cap S^*\neq\emptyset} x^*(P) = (|V'| + 4|E'|)\ell M + k'\ell + f_{x^*}(2h^*).$$

This proves Lemma 4. □

We use Lemma 4 to prove Lemma 3 as follows. Observe that (x^*, λ^*) maximizes the quantity $\sum_{P\in\mathcal{P}} x^*(P) - \lambda^*$. Define $C_1 := (|V'| + 4|E'|)\ell M + \ell|V'| + \sum_{i=1}^{k} u(e_i)$. As we already fixed the flow value on all paths outside of the subgraph H, we know that

$$\sum_{P\in\mathcal{P}} x^*(P) = C_1 + \sum_{P\in\mathcal{P}:P\subseteq E_H} x^*(P)$$
$$= C_1 + x^*(v', t) + x^*(v', v'') + x^*(s, v''),$$

where the last three summands together determine the total nominal flow through H. Defining

$$C_2 := (|V'| + 4|E'|)\ell M + k'\ell,$$

Lemma 4 states that $\lambda^* = C_2 + f_{x^*}(2h^*)$. Thus

$$\sum_{P\in\mathcal{P}} x^*(P) - \lambda^* = C_1 - C_2 + x^*(v', t) + x^*(v', v'') + x^*(s, v'') - f_{x^*}(2h^*).$$

Since, by definition, the values C_1 and C_2 do not depend on x^*, optimality of x^* implies that the flow maximizes the quantity

$$x^*(v', t) + x^*(v', v'') + x^*(s, v'') - f_{x^*}(2h^*).$$

We use this to finally show that $x^*(v', v'') > 0$ if and only if G' contains a clique of size k'.

First, assume that there is no clique of size k' in G', i.e., $h^* \le \binom{k'}{2} - 1$. In this case, $2h^* \le h$ and therefore $f_{x^*}(2h^*) = 2h^*(1+\varepsilon)$, independent of the flow values in the subgraph H, as no arc in E_H can carry more than $1+\varepsilon$ units of flow and there are already h arcs with flow value $1+\varepsilon$ in $F \setminus E_H$. Therefore, x^* maximizes $x^*(v', t) + x^*(v', v'') + x^*(s, v'')$, which implies it is the unique maximum flow in H, which fulfills $\sum_{P:(v',v'')\in P} x^*(P) = 0$.

Now assume G' has a clique of size k' and thus $h^* = \binom{k'}{2}$. In this case $2h^* = h + 2$ and hence

$$f_{x^*}(2h^*) = 2h \cdot (1+\varepsilon) + \max\{1, x^*(v', t)\} + \max\{1, x^*(s, v'')\},$$

as (v', t) and (s, v'') are the only two arcs in F outside $\{e_1, \ldots, e_h\}$ that can carry more than 1 unit of flow. Thus the flow x^* maximizes

$$x^*(v', t) + x^*(v', v'') + x^*(s, v'') - \big(\max\{1, x^*(v', t)\} + \max\{1, x^*(s, v'')\}\big).$$

This term is maximized for $x^*(v', t) = x^*(s, v'') = 1$ and $x^*(v', v'') = \varepsilon$.

The above two observations conclude the proof of Lemma 3. □

Note that the size of the graph $G = (V, E)$ constructed in the reduction is polynomial in the size of G'. Furthermore observe that the number of paths in G is polynomial in $|E|$ and that all capacities are polynomial in the size of G (note that the capacity ∞ can be replaced by $|E|M$, and multiplying all capacities with ℓ yields integral capacities). This concludes the proof of Theorem 2. □

3.1. Reduction to two capacity values

We now observe that there is a pseudopolynomial transformation that converts general instances of MAXIMUM ROBUST FLOW to instances with where the capacity of each arc is one of two different values: 1 or $u_{\max}$, where $u_{\max}$ is the maximum capacity value occurring in the original instance.

Lemma 5. *There is an algorithm that given an instance of* MAXIMUM ROBUST FLOW $I = ((V, E), s, t, u, k)$ *computes in time* $\mathcal{O}(|E|u_{\max})$ *an instance of* MAXIMUM ROBUST FLOW $I' = ((V', E', s', t', u', k)$ *with* $u'(e) \in \{1, u_{\max}\}$ *for all* $e \in E'$ *such that the maximum robust flow value of* I *and* I' *is identical, where* $u_{\max} := \max_{e\in E} u(e)$*. Moreover, given an (integral) flow* x' *in* I' *one can compute in time polynomial in* $|E|$ *and* $|\mathrm{supp}(x')|$ *an (integral) flow* x *in* I *with* $\mathrm{val}_r(x) = \mathrm{val}_r(x')$.

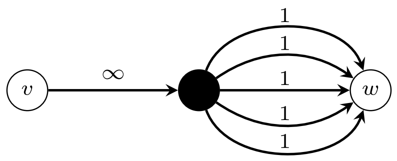

Fig. 2. Construction for Lemma 5. Arc (v, w) with capacity 5 is replaced by a sequence of an arc with capacity ∞ and 5 arcs with capacity 1.

Proof. To obtain an equivalent instance in which only the capacities 1 and $u_{\max}$ occur, observe that we can replace any arc of capacity u by the concatenation of an arc of capacity $u_{\max}$ and u parallel arcs of capacity 1. Failure of the original arc corresponds to failure of the infinite capacity arc in the modified instance. See Fig. 2 for an illustration. □

In particular, this implies that our hardness result still holds in the more restricted setting of capacities 1 and ∞ (where ∞ can also be replaced by a number that is polynomially bounded in the network size).

Corollary 6. Maximum Robust Flow *is NP-hard, even when restricted to instances where* $u(e) \in \{1, \infty\}$ *for all* $e \in E$ *and where the number of paths is polynomial in the size of the graph.*

3.2. Unit capacities

On the other hand, it is not hard to see that the problem becomes easy in the unit capacity case.

Theorem 7. *If* $u \equiv 1$ *then any maximum flow also is a maximum robust flow.*

Proof. Let C be a minimum s-t-cut in G. If $|C| \leq k$, then every flow has robust value 0. Thus assume $|C| > k$. Let x be any s-t-flow. Clearly, $\text{val}_r(x) \leq |C| - k$, as after removal of any k arcs from C, the remaining flow must traverse the $|C| - k$ remaining arcs in the cut. Now assume x is a maximum flow, i.e., $\text{val}(x) = |C|$. Since every arc carries at most 1 unit of flow, the removal of any k arcs from G can only decrease the flow value by k, thus x is an optimal solution to Maximum Robust Flow. □

4. Integral robust flows

In this section, we show that finding a maximum *integral* robust flow is *NP*-hard already for instances with $k = 2$. This is in contrast to the case $k = 1$, for which it is possible to efficiently compute the best integral solution [3]. In fact, our reduction implies that it is hard to distinguish instances with optimal value 2 or 3, resulting in hardness of approximation for the integral problem. Interestingly, the fractional version of the problem admits a 4/3-approximation algorithm for $k = 2$ [5], indicating that the integral problem is indeed harder.

Theorem 8. *Unless* $P = NP$, *there is no* $(3/2-\varepsilon)$*-approximation algorithm for* Integral Maximum Robust Flow, *even when restricted to instances where* $k = 2$ *and* $u(e) \leq 3$ *for all* $e \in E$.

Proof. We reduce from Arc-disjoint Paths, which is well-known to be *NP*-hard [10]. As input of Arc-disjoint Paths, we are given a directed graph $G' = (V', E')$ and two pairs of nodes (s_1, t_1) and (s_2, t_2). The task is to decide whether there is an s_1-t_1-path P'_1 and an s_2-t_2-path P'_2 in G' with $P'_1 \cap P'_2 = \emptyset$.

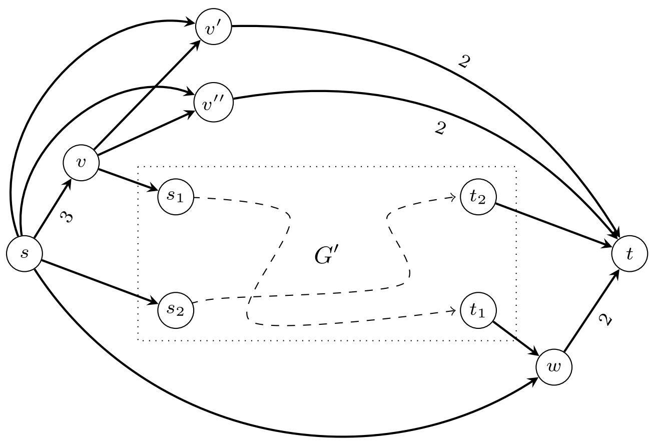

Fig. 3. The construction for showing *NP*-hardness of Integral Maximum Robust Flow with $k = 2$. The dotted box contains an instance of Arc-disjoint Paths. Labels at the arcs show the capacities. All unlabeled arcs have unit capacity.

From the input graph $G' = (V', E')$, we construct an instance of Maximum Robust Flow by adding 6 new nodes and 13 new arcs, obtaining a new directed graph $G = (V, E)$ with

$$\begin{aligned} V &= V' \cup \{s,\ t,\ v,\ v',\ v'',\ w\} \\ E &= E' \cup \{(s, v),\ (s, v'),\ (s, v''),\ (v, s_1),\ (v, v'),\ (v, v''), \\ &\qquad (v', t),\ (v'', t),\ (s, w),\ (t_1, w),\ (w, t),\ (s, s_2),\ (t_2, t)\}. \end{aligned}$$

We set $u(s, v) = 3$ and $u(v', t) = u(v'', t) = u(w, t) = 2$. All other arcs have capacity 1. The whole construction is depicted in Fig. 3.

We show that there is an integral flow x with $\text{val}_r(x) \geq 3$ if an only if there is an s_1-t_1-path P'_1 and an s_2-t_2-path P'_2 in G' with $P'_1 \cap P'_2 = \emptyset$. It is thus *NP*-hard to distinguish instances of Integral Maximum Robust Flow with optimal value at least 3 from those with optimal value at most 2.

First assume there is an integral flow x with robust value $\text{val}_r(x) = 3$. Consider the arc set $S' = \{(s, v), (w, t)\}$. As $\sum_{P:P\cap S'=\emptyset} x(P) \geq 3$, there must be three s-t-paths carrying 1 unit of flow each and not intersecting with S'. These paths must thus start with the arcs (s, v'), (s, v''), and (s, s_2), respectively. In particular, the latter path must end with (t_2, t), because (t_2, t) together with (w, t) forms an s_2-t-cut, and $(w, t) \in S'$ is not contained in the path. Let P_2 be this unique flow-carrying path starting with (s, s_2) and ending with (t_2, t). Note that all arcs of P_2 have unit capacity and thus no other flow-carrying path can intersect P_2. Now consider the arc set $S'' = \{(v', t), (v'', t)\}$. Because the arc (v, s_1) is part of an s-t-cut with capacity 3 in the network $(V, E \setminus S'')$, there must be a flow path P_1 containing (v, s_1). As (t_2, t) is already saturated by the flow on P_2, the path P_1 must use (t_1, w). In particular, P_1 contains an s_1-t_1-path P'_1 and P_2 contains an s_2-t_2-path P'_2, and $P_1 \cap P_2 = \emptyset$.

Conversely, assume there is an s_1-t_1-path P'_1 and an s_2-t_2-path P'_2 in G' with $P'_1 \cap P'_2 = \emptyset$. Let

$$P_1 := \{(s, v),\ (v, s_1)\} \cup P'_1 \cup \{(t_1, w),\ (w, t)\} \text{ and}$$
$$P_2 := \{(s, s_1)\} \cup P'_2 \cup \{(t_2, t)\}.$$

Send 1 unit of flow along each of the paths P_1, P_2 and the five remaining paths s-v'-t, s-v''-t, s-v-v'-t, s-v-v''-t, and s-w-t, obtaining a flow x. Now assume by contradiction that $\text{val}_r(x) < 3$. Because $\text{val}(x) = 7$, there must be $S \in \mathcal{S}$ with $\sum_{P:P\cap S\neq\emptyset} x(P) > 4$. In particular, the arc (s, v) must be contained in S, as it is the only arc carrying more than 2 units of flow. The other arc in S must be one of the arcs with capacity 2, i.e., (v, v'), (v, v''), or (w, t). However, each of these three arcs is contained in one of the three flow paths using (s, v). Thus $\sum_{P:P\cap S\neq\emptyset} x(P) = 4$, a contradiction. □

For the reduction given above to work, it is sufficient to have arcs of capacity at most 3. We now argue that the problem can be solved efficiently for arbitrary values of k when capacities are bounded by 2.

Theorem 9. Integral Maximum Robust Flow *restricted to instances with $u(e) \leq 2$ for all $e \in E$ can be solved in polynomial time.*

Proof. Let x^* be an optimal solution to Integral Maximum Robust Flow. Let x_1 be a maximum flow in G with respect to unit capacities and let x_2 be a maximum flow in G with respect to capacities u. As capacities are integral, we can assume without loss of generality that x_1 and x_2 are integral. We will show that $\mathrm{val_r}(x^*) = \max\{0,\ \mathrm{val}(x_1) - k,\ \mathrm{val}(x_2) - 2k\}$.

To prove this claim, consider a minimum cardinality s-t-cut C in G. We greedily construct a set $S \subseteq C$ with $|S| \leq k$ as follows: Starting with $S = \emptyset$, iteratively add an arc $e \in C \setminus S$ to S that maximizes

$$\Delta(S, e) := \textstyle\sum_{P \in \mathcal{P}\,:\,e \in P,\ P \cap S = \emptyset} x^*(P)$$

until $|S| = k$ or $S = C$. In other words, we greedily add an arc from C to S that removes the most flow from x^*. Note that throughout this selection process, the value $\Delta(S, e) \in \{0, 1, 2\}$ is non-increasing for each $e \in C$. After construction of S, let $\Delta := 0$ if $S = C$ and $\Delta := \max_{e \in C \setminus S} \Delta(S, e)$ otherwise.

1. If $\Delta = 0$, then $\sum_{P \in \mathcal{P}: P \cap S \neq \emptyset} x^*(P) = \mathrm{val}(x^*)$ and therefore $\mathrm{val_r}(x^*) = 0$. In this case, any integral flow is an optimal solution.
2. If $\Delta = 1$, then $\sum_{P \in \mathcal{P} \in \mathcal{P}: e \in P, P \cap S = \emptyset} x^*(P) \leq 1$ for every arc $e \in C \setminus S$. Therefore $\mathrm{val_r}(x^*) \leq \sum_{P: P \cap S = \emptyset} x^*(P) \leq |C \setminus S| = \mathrm{val}(x_1) - k \leq \mathrm{val_r}(x_1)$, where the equality follows from $\mathrm{val}(x_1) = |C|$ and the last inequality follows from the fact that every arc carries at most 1 unit of flow in x_1. We conclude that x_1 is an optimal solution in this case.
3. If $\Delta = 2$, then $\sum_{P \in \mathcal{P}: P \cap S \neq \emptyset} x^*(P) = 2k$, because in every iteration an arc e with $\Delta(S, e) = 2$ was added to S. Thus $\mathrm{val_r}(x^*) \leq \mathrm{val}(x^*) - 2k \leq \mathrm{val}(x_2) - 2k \leq \mathrm{val_r}(x_2)$, where the last inequality follows from the fact that every arc carries at most 2 units of flow in x_2. We conclude that x_2 is an optimal solution in this case.

Computing x_1 and x_2 and determining whether or not $\mathrm{val}(x_1) - k \geq \mathrm{val}(x_2) - 2k$ can be done in polynomial time. □

5. Conclusion

Our hardness result for Maximum Robust Flow presented in Section 3 crucially requires the number of failing arcs k to be large. This immediately leads to the following question, which had seemingly been settled over a decade ago, but is now open once more:

Question. *What is the complexity of* Maximum Robust Flow *when restricted to instances where $k \geq 2$ is a constant?*

Acknowledgments

We thank Tom McCormick and Gianpaolo Oriolo for numerous helpful discussions that led to the discovery of the flaw in the previous publication. This work was supported by the 'Excellence Initiative' of the German Federal and State Governments, Germany, the Graduate School CE at TU Darmstadt, Germany, and by the Alexander von Humboldt Foundation, Germany with funds of the German Federal Ministry of Education and Research (BMBF), Germany. Part of this research was carried out while the second author worked at Technische Universität München.

References

[1] C.C. Aggarwal, J.B. Orlin, On multiroute maximum flows in networks, Networks 39 (2002) 43–52.
[2] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993.
[3] Y.P. Aneja, R. Chandrasekaran, K.P.K. Nair, Maximizing residual flow under an arc destruction, Networks 38 (2001) 194–198.
[4] J.F. Baffier, V. Suppakitpaisarn, H. Hiraishi, H. Imai, Parametric multiroute flow and its application to multilink-attack network, Discrete Optim. 22 (2016) 20–36.
[5] D. Bertsimas, E. Nasrabadi, J.B. Orlin, On the power of randomization in network interdiction, Oper. Res. Lett. 44 (2016) 114–120.
[6] D. Bertsimas, E. Nasrabadi, S. Stiller, Robust and adaptive network flows, Oper. Res. 61 (2013) 1218–1242.
[7] D. Bertsimas, M. Sim, Robust discrete optimization and network flows, Math. Program. 98 (2003) 49–71.
[8] S.R. Chestnut, R. Zenklusen, Hardness and approximation for network flow interdiction, Networks 69 (2017) 378–387.
[9] D. Du, R. Chandrasekaran, The maximum residual flow problem: NP-hardness with two-arc destruction, Networks 50 (2007) 181–182.
[10] S. Fortune, J. Hopcroft, J. Wyllie, The directed subgraph homeomorphism problem, Theoret. Comput. Sci. 10 (1980) 111–121.
[11] C. Gottschalk, A.M. Koster, F. Liers, B. Peis, D. Schmand, A. Wierz, Robust flows over time: models and complexity results, Math. Program. 171 (2017) 55–85.
[12] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer, 1988.
[13] J. Matuschke, S.T. McCormick, G. Oriolo, Rerouting flows when links fail, in: 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017), Dagstuhl Publishing, 2017a, pp. 89:1–89:13.
[14] J. Matuschke, S.T. McCormick, G. Oriolo, B. Peis, M. Skutella, Protection of flows under targeted attacks, Oper. Res. Lett. 45 (2017b) 53–59.
[15] R.K. Wood, Deterministic network interdiction, Math. Comput. Modelling 17 (1993) 1–18.