Rerouting Flows When Links Fail*†

Jannik Matuschke[1], S. Thomas McCormick[2], and Gianpaolo Oriolo[3]

1 **TUM School of Management and Department of Mathematics, Technische Universität München, Munich, Germany**
jannik.matuschke@tum.de
2 **Sauder School of Business, University of British Columbia, Vancouver, Canada**
tom.mccormick@sauder.ubc.ca
3 **Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università di Roma "Tor Vergata", Roma, Italy**
oriolo@disp.uniroma2.it

Abstract

We introduce and investigate *reroutable flows*, a robust version of network flows in which link failures can be mitigated by rerouting the affected flow. Given a capacitated network, a path flow is reroutable if after failure of an arbitrary arc, we can reroute the interrupted flow from the tail of that arc to the sink, without modifying the flow that is not affected by the failure. Similar types of restoration, which are often termed "local", were previously investigated in the context of network design, such as min-cost capacity planning. In this paper, our interest is in computing maximum flows under this robustness assumption. An important new feature of our model, distinguishing it from existing max robust flow models, is that no flow can get lost in the network.

We also study a tightening of reroutable flows, called *strictly reroutable flows*, making more restrictive assumptions on the capacities available for rerouting. For both variants, we devise a reroutable-flow equivalent of an s-t-cut and show that the corresponding max flow/min cut gap is bounded by 2. It turns out that a strictly reroutable flow of maximum value can be found using a compact LP formulation, whereas the problem of finding a maximum reroutable flow is *NP*-hard, even when all capacities are in $\{1, 2\}$. However, the tightening can be used to get a 2-approximation for reroutable flows. This ratio is tight in general networks, but we show that in the case of unit capacities, every reroutable flow can be transformed into a strictly reroutable flow of same value. While it is *NP*-hard to compute a maximal integral flow even for unit capacities, we devise a surprisingly simple combinatorial algorithm that finds a half-integral strictly reroutable flow of value 1, or certifies that no such solutions exits. Finally, we also give a hardness result for the case of multiple arc failures.

1998 ACM Subject Classification G.2.2 [Graph Theory] Graph Algorithms

Keywords and phrases network flows, network interdiction, robust optimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.89

1 Introduction

Network infrastructures for transportation, communication, or energy transmission are an important backbone of our society. However, they are also prone to failure or intentional

* A full version of this article is available at https://arxiv.org/abs/1704.07067.
† This work was supported by the Alexander von Humboldt Foundation with funds of the German Federal Ministry of Education and Research (BMBF) and by an NSERC Discovery Grant.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 89; pp. 89:1–89:13


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

sabotage, and in such cases it is desirable to quickly recover the service provided through the network. A crucial frequent requirement of actual network restoration techniques is that restoration is handled locally [13]. As a motivating example, consider a communication network in which data packets are routed along paths. When a link in the network fails, it is desirable to only reroute the traffic that is actually affected by the failure, i.e., those paths that traverse the failing link, without changing or rerouting any part of the flow that is not affected by the failure. Note that arbitrary rearrangement of the flow after a failure is in general more powerful, but it is both undesirable to interrupt customer service and hard to do so reliably and safely [9, 16].

To cope with such a situation, we introduce the concept of reroutable network flows: A flow on s-t-paths is *reroutable* if after failure of any arc $\bar{a} = (\bar{v}, \bar{w})$ in the network, we can reroute all flow that was traversing $\bar{a}$ from $\bar{v}$ to the sink t, while not changing any flow that was not affected by the interruption. Similar concepts were previously discussed in a few other papers [6, 7, 17, 18], but with an emphasis on network design issues, e.g., minimizing the cost of the installed capacity. In contrast, our interest is in computing maximum flows (but we point out that a potential application are feasibility/separation subroutines for capacity reservation). Note that in this setting, we cannot simply send a standard maximum flow, as we need to leave space for rerouting. Before we discuss our findings and better relate them to existing literature, let us formalize the definition of our model.

Network flows. Let $D = (V, A)$ be a digraph with source $s \in V$, a sink $t \in V$ and arc capacities $u \in \mathbb{R}^A_+$. Let $\mathcal{P} \subseteq 2^A$ be the set of simple[1] s-t-paths in D. For arcs $a, \bar{a} \in A$, define

$$\mathcal{P}_a := \{P \in \mathcal{P} : a \in P\} \quad \text{and} \quad \mathcal{P}_{\bar{a} \to a} := \{P \in \mathcal{P} : a, \bar{a} \in P,\, \bar{a} \prec_P a\},$$

where $\bar{a} \prec_P a$ means that P traverses $\bar{a}$ before a. An *s-t-flow* is a vector $x \in \mathbb{R}^{\mathcal{P}}_+$ that assigns a flow value $x(P) \geq 0$ to each $P \in \mathcal{P}$ such that the arc flow values $x(a) := \sum_{P \in \mathcal{P}_a} x(P)$ fulfill the capacity constraint $x(a) \leq u(a)$ for all $a \in A$. The value of a flow x is $\mathrm{val}(x) := \sum_{P \in \mathcal{P}} x(P)$.

Reroutable flows. Let x be an s-t-flow. If an arc $\bar{a} = (\bar{v}, \bar{w}) \in A$ fails, all flow on paths containing the failing arc gets interrupted when it reaches $\bar{v}$. For any $a \in A \setminus \{\bar{a}\}$, we define the *available capacity* of a after failure of $\bar{a}$ by

$$\bar{u}_{x,\bar{a}}(a) := u(a) - \sum_{P \in \mathcal{P}_a \setminus \mathcal{P}_{\bar{a} \to a}} x(P).$$

A *rerouting* of x for the failing arc $\bar{a}$ is a $\bar{v}$-t-flow $x_{\bar{a}}$ of value $x(\bar{a})$ in $(V, A \setminus \{\bar{a}\})$ with capacities $\bar{u}_{x,\bar{a}}$. The flow x is *reroutable* if for every failing arc $\bar{a} \in A$ there is a rerouting $x_{\bar{a}}$ of x.

Strictly reroutable flows. A rerouting $x_{\bar{a}}$ of a flow x for a failing arc $\bar{a}$ is *strict* if $x_{\bar{a}}(a) \leq \bar{u}_x(a) := u(a) - x(a)$ for every $a \in A \setminus \{\bar{a}\}$. We say that x is *strictly reroutable* if for every failing arc $\bar{a} \in A$ there is a strict rerouting of x.

Strictly reroutable flows are both a helpful tool for computing reroutable flows and interesting in their own right, in situations where more conservative assumptions have to be

[1] All our results also work for the case that $\mathcal{P}$ contains non-simple paths, but we restrict to simple paths for ease of notation.

made on the capacities available for rerouting. A natural question is what is the maximum flow value that can be sent by a (strictly) reroutable flow in a given network. We denote the corresponding optimization problem as Max RF and Max SRF, respectively.

1.1 Our results

Complexity of the problems (Sections 2.1 and 2.2). We observe that Max SRF can be solved in polynomial time by formulating it as a linear program. In contrast, Max RF is *NP*-hard, even when $u(a) \in \{1, 2\}$ for all $a \in A$. On the positive side, by showing that the maximum value of a reroutable flow is at most twice as large as the maximum value of a strictly reroutable flow, we obtain a 2-approximation for Max RF for arbitrary capacities. The problem can further be solved exactly in unit capacity networks (see below).

Max flow/min cut gap (Section 2.3). Max flow/min cut results play a central role in network flow theory. We devise a combinatorial upper bound for the maximum reroutable flow value, called *R-cut*, and prove that the corresponding flow/cut gap for both reroutable and strictly reroutable flows is bounded by 2. In fact, our proof is constructive and provides a combinatorial 2-approximation algorithm for the minimum capacity R-cut problem.

Unit capacity networks (Section 3). We consider the case of unit capacities. It turns out that in this case, Max RF and Max SRF are equivalent. Our proof is based on a careful uncrossing argument that allows to transform any reroutable flow into a strictly reroutable flow.

Computing (half-)integral solutions (Section 4). A common property of many flow problems is the existence of an integral optimal solution when capacities are integral. In the case of reroutable flows, this property does not hold. In fact, if we require flow to be integral, the problem becomes *NP*-hard, even for sending a single unit of flow in a unit capacity network. However, for this special case, we devise a simple combinatorial algorithm that computes a half-integral solution or certifies that no flow of value 1 exists. Via our max flow/min cut analysis we also show how to compute 2-approximate half-integral solutions.

Multiple arc failures (Section 5.2). We consider the natural generalization of our problems to multiple simultaneous arc-failures. We show that in this case both variants of the problem are *NP*-hard, even when only two arcs can fail and all arcs have unit capacity. All hardness results in this paper are based on reduction from an intermediary problem, called Forbidden Pairs s-t-Path. They are therefore grouped together in Section 5.

1.2 Related work

As we already pointed out above, "local" rerouting schemes, i.e., schemes that only change flow affected by the failure, have been investigated in network design. A routing scheme in which flow has to be sent along arc-disjoint paths was investigated in [6], see also [18]. The problem of finding a local rerouting from the tail to the head of a failed arc was investigated in [7] and [17]. However, in all these papers the focus was on min-cost capacity planning.

Concepts that deal with the maximization of flow subject to robustness constraints commonly fall under the moniker of *robust flows*. Aggarwal and Orlin [2] studied *k-route flows*. Such a flow is a conic combination of elementary flows, each of which consists of a uniform flow along k disjoint paths. Because of this structure, the failure of any arc can only

destroy a $1/k$ fraction of the flow. A maximum k-route flow can be computed in polynomial time by means of a parametric max flow problem. Another classic model is the *maximum robust flow problem*: Here, the goal is to find a path flow that maximizes the surviving flow after a worst-case failure of k arcs. Aneja et al. [3] showed that for $k = 1$ both an optimal fractional and an optimal integral solution can be found in polynomial time. If k is not bounded by a constant the problem is *NP*-hard [10], but the complexity for any constant value $k \geq 2$ is open. Bertsimas et al. [4] provide an $\Omega(1/k)$-approximation algorithm for the maximum robust flow. Robust flows are closely related to *network flow interdiction*, which takes a dual perspective: The goal is to find a subset of arcs whose removal minimizes the maximum flow value in the remaining network; see the recent article by Chestnut and Zenklusen [8] for an up-to-date overview of this topic.

To the best of our knowledge, the only other flow maximization model that allows for adjustment after the failure are *adaptive flows*, first introduced by Bertsimas et al. [5]: In the first step, an arc flow is specified. After failure of k arcs, a new flow is sent, with the flow value on every arc being bounded by the original flow value. Note that adaptive flows differ from reroutable flows in two important aspects: Adaptive flows allow flow to be 'lost' (the flow value after the failure is lower than the original flow value), whereas in reroutable flows all flow has to reach the sink. Furthermore, adaptive flows can reconfigure the flow in the entire network, whereas in reroutable flows, only the flow affected by the failure can be rerouted.

Another model closely related to reroutable flows is the *online replacement path* problem (ORP) introduced by Adjiashvili et al. [1]. The ORP is a generalization of the shortest path problem: Given a digraph with costs on the arcs, we have to specify an s-t-path. Along the path, we may encounter a failing arc $\bar{a} = \{\bar{v}, \bar{w}\}$, and we have to find a replacement path from $\bar{v}$ to t avoiding $\bar{a}$. The goal is to minimize the total traveled distance, assuming $\bar{a}$ is chosen by an adversary. Adjiashvili et al. [1] show that the ORP can be solved in polynomial time, even when a constant number of arcs fail.

2 LP formulation, approximation, and max flow/min cut

In this section, we discuss the complexity of the two problems and provide bounds on the gap between Max RF and Max SRF. We also introduce an analogue to minimum cuts for reroutable flows and bound the corresponding duality gap. At the end of the section, we show that all our bounds are tight.

2.1 Complexity of Max RF and Max SRF

We now consider an LP formulation for Max SRF. For $\bar{a} \in A$, let $\mathcal{R}(\bar{a})$ be the set of all tail$(\bar{a})$-t-paths in $(V, A \setminus \{\bar{a}\})$, which are exactly the paths that a rerouting for failing arc $\bar{a}$ can use.

$$
\begin{aligned}
[\mathrm{LP}_{\mathrm{strict}}] \qquad \max \quad & \sum_{P \in \mathcal{P}} x(P) & & \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_a} x(P) + \sum_{R \in \mathcal{R}(\bar{a}) \,:\, a \in R} x_{\bar{a}}(R) \;\leq\; u(a) & & \forall\, a, \bar{a} \in A \\
& \sum_{P \in \mathcal{P}_{\bar{a}}} x(P) - \sum_{R \in \mathcal{R}(\bar{a})} x_{\bar{a}}(R) \;=\; 0 & & \forall\, \bar{a} \in A \\
& x, x_{\bar{a}} \geq 0 & & \forall\, \bar{a} \in A
\end{aligned}
$$

The first set of constraints bound the capacities for each rerouting; note in particular that for $\bar{a} = a$, the second term becomes 0, ensuring $x(a) \leq u(a)$ for all $a \in A$. The second set of constraints ensures that the rerouting flow $x_{\bar{a}}$ has value $x(\bar{a})$. Although [$\text{LP}_{\text{strict}}$] has an exponential number of variables, it can be solved in polynomial time via dual separation.

▶ **Theorem 1.** Max SRF *can be solved in polynomial time.*

For the special case of unit capacity networks, we show in Section 3 that an optimal solution to [$\text{LP}_{\text{strict}}$] is also optimal for Max RF.

▶ **Theorem 2.** *For $u \equiv 1$,* Max RF *can be solved in polynomial time.*

An LP for Max RF can be obtained by replacing $\sum_{P \in \mathcal{P}_a} x(P)$ by $\sum_{P \in \mathcal{P}_{\bar{a} \to a}} x(P)$ in the capacity constraints of [$\text{LP}_{\text{strict}}$]. Unfortunately, this modification prevents the dual separation approach from working. In fact, it turns out that Max RF is hard as soon as two different capacities occur. The proof of this result is discussed in Section 5.1.

▶ **Theorem 3.** Max RF *is NP-hard, even when $u(a) \in \{1, 2\}$ for all $a \in A$.*

2.2 Reroutable flows vs. strictly reroutable flows

As Max SRF is a tightening of Max RF, the optimal value of the former is at most that of the latter. We show that the gap between the two values cannot be larger than 2. As we can compute maximum strictly reroutable flows, we obatain a 2-approximation for Max RF.

▶ **Lemma 4.** *Let x be an s-t-flow. If x is strictly reroutable, then x is reroutable. If x is reroutable, then $\frac{1}{2}x$ is strictly reroutable.*

▶ **Corollary 5.** *There is a 2-approximation algorithm for* Max RF.

2.3 Max flow/min cut gap for reroutable flows

An *s-t-cut* is a set of arcs that intersects every s-t-path. Its *capacity* is the sum of capacities of its arcs. A fundamental result in network flow theory is that the value of a maximum s-t-flow is equal to the capacity of a minimum s-t-cut. This result has been successfully generalized to many variants of network flows, such as abstract flows [14] or flows over time [11]. However, in other cases, such as multicommodity flows, the equality does not hold and instead, researchers investigate the worst case ratio between maximum flow and minimum cut; see, e.g., [15].

We present a counterpart to an s-t-cut for reroutable flows. It turns out that max flow and min cut are not necessarily equal and we give a tight bound on the corresponding max flow/min cut gap. An *R-cut* is a set of arcs $R \subseteq A$ together with a collection of cuts $(C_a)_{a \in R}$, where each C_a is a tail(a)-t-cut containing a. We denote $(R, (C_a)_{a \in R})$ by (R, C) for short. The capacity of the R-cut (R, C) is

$$\text{cap}(R, C) := \phi(R, C) + \sum_{a \in R} u(C_a \setminus \{a\}),$$

where $\phi(R, C)$ is the capacity of a minimum s-t-cut in $(V, A \setminus \cup_{a \in R} C_a)$.

The intuition behind this definition is the following: For every $a \in R$, all flow that crossed the cut C_a must cross the $C_a \setminus \{a\}$ if a fails. If a flow path does not cross any cut in C_a, then it crosses the minimum s-t-cut in $(V, A \setminus \cup_{a \in R} C_a)$. Therefore the capacity of an R-cut is an upper bound on the value of any reroutable flow.

▶ **Lemma 6.** $\mathrm{val}(x) \leq \mathrm{cap}(R, C)$ *for any reroutable flow* x *and any* R*-cut* (R, C).

It can be shown that R-cuts correspond to integral solutions to the dual of $[\mathrm{LP}_{\mathrm{strict}}]$. We now give a constructive proof bounding the duality gap between maximum strictly reroutable flow and minimum R-cut (or, equivalently, the integrality gap of the dual LP). In Section 2.4 we give an example showing that the bound is tight.

▶ **Theorem 7.** *Let* x *be a strictly reroutable flow of maximum value and let* (R, C) *be an* R*-cut of minimum capacity. Then* $\mathrm{val}(x) \geq \frac{1}{2}\,\mathrm{cap}(R, C)$.

Proof. For $a \in A$, let C_a be minimum $\mathrm{tail}(a)$-t-cut in D containing a and define $u'(a) := \min\{u(a), u(C_a \setminus \{a\})\}$. Let C' be a minimum s-t-cut in D with respect to the capacities u' and let x' be a corresponding maximum flow. Now define $R := \{a \in C' : u'(a) < u(a)\}$. Observe that R and $(C_a)_{a \in R}$ define an R-cut and that $\phi(R, C) \leq u(C' \setminus R)$. We obtain

$$\mathrm{cap}(R, C) \leq \sum_{a \in C' \setminus R} u(a) + \sum_{a \in R} u(C_a \setminus \{a\}) = \sum_{a \in C'} u'(a) = \mathrm{val}(x').$$

Now let $x := x'/2$. It is sufficient to show that x is a strictly reroutable flow. By contradiction assume that there is $\bar{a} \in A$ for which there is no strict rerouting of x. By the max flow/min cut theorem, there must be a $\mathrm{tail}(\bar{a})$-t-cut $\bar{C}$ in $(V, A \setminus \{\bar{a}\})$ with $\sum_{a \in \bar{C}} \bar{u}_x(a) < x(\bar{a})$. Note that $x(a) \leq u'(a)/2 \leq u(a)/2$ for every $a \in A$ by construction of x. Thus

$$\frac{1}{2} \sum_{a \in \bar{C}} u(a) \leq \sum_{a \in \bar{C}} (u(a) - x(a)) < x(\bar{a}) \leq \frac{1}{2} u'(\bar{a}) \leq \frac{1}{2} u(C_{\bar{a}} \setminus \{\bar{a}\}).$$

However, this implies that $\bar{C} \cup \{\bar{a}\}$ is a smaller $\mathrm{tail}(\bar{a})$-t-cut than $C_{\bar{a}}$, a contradiction. ◀

Computing a minimum capacity R-cut. Let us denote the problem of finding an R-cut of minimum capacity by Min R-Cut. The proof of Theorem 7 describes how to compute a 2-approximate solution to this problem.

▶ **Corollary 8.** *There is a* 2*-approximation algorithm for* Min R-Cut.

2.4 Summary of the bounds and tightness

Putting the bounds from Lemma 4 and Theorem 7 together, we obtain the following corollary.

▶ **Corollary 9.** *Let* (R, C) *be a minimum capacity* R*-cut and let* x_{RF} *and* x_{SRF} *be maximal reroutable and strictly reroutable flows, respectively. Then*

$$\mathrm{val}(x_{\mathrm{RF}}) \leq \mathrm{cap}(R, C) \leq 2\,\mathrm{val}(x_{\mathrm{SRF}}) \leq 2\,\mathrm{val}(x_{\mathrm{RF}}).$$

The example given in Figure 1 shows that each of the bounds proven in this section is tight. It also shows that optimal solutions to both Max RF and Max SRF can be fractional, even when capacities are integral. In the depicted network, and in further examples and reductions throughout the paper, we use the following gadget.

Backup links. A *backup link* from v to w is a v-w-path (a', a'') of length 2 in which the intermediate node is incident only to the two arcs of the path and $u(a') := u(a'') := \max_{a \in A} u(a)$. Note that $x(a') = x(a'') = 0$ for any reroutable flow, because when a'' fails, there is no $\mathrm{tail}(a'')$-t-path for rerouting the flow on that arc. A *bidirected* backup link between v and w consists of two distinct backup links, one from v to w and one from w to v.

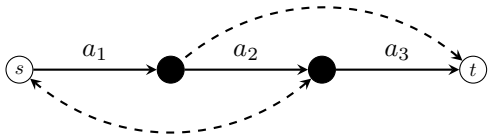

Figure 1 Example showing that the bounds given in Lemma 4 and Theorem 7 are tight. Dashed arcs correspond to (bidirected) backup links, which can only be used for rerouting. When all arcs have unit capacities, the maximum (strictly) reroutable flow has a value of 1/2. When changing the capacity of a_1 to 2, the maximum reroutable flow value increases to 1, whereas the maximum strictly reroutable flow value remains 1/2. The minimum R-cut capacity is 1 in both cases.

▶ Remark. Note that the worst-case for the bounds in Corollary 9 cannot be attained simultaneously, i.e., in any given instance either the max flow/min cut gap or the gap between reroutable and strictly reroutable flow has to be significantly smaller than 2—in fact, at least one of them has to be within $\sqrt{2}$.

3 Unit capacity networks

Throughout this section, we assume $u \equiv 1$. We will show that in this case, any reroutable flow can be transformed into a strictly reroutable flow of the same value. We start by giving an alternative characterization for strictly reroutable flows in unit capacity networks.

Cuts separating t. For $S \subseteq V$, let $\delta^+(S) := \{a \in A : \text{tail}(a) \in S, \text{head}(a) \in V \setminus S\}$ denote the cut induced by S. We define $\mathcal{S} := \{S \subset V \setminus \{t\} : S \neq \emptyset\}$ and let $\mathcal{C} := \{\delta^+(S) : S \in \mathcal{S}\}$ be the set of *t-separating cuts*. W.l.o.g. we assume $\delta^+(S) \neq \emptyset$ for all $S \in \mathcal{S}$, as no vertex in a set S with $\delta^+(S) = \emptyset$ can be on an s-t-path.

▶ **Lemma 10.** *Let x be an s-t-flow for capacities $u \equiv 1$. Then x is strictly reroutable if and only if $\sum_{a \in C}(1 - x(a)) \geq 1$ for all $C \in \mathcal{C}$.*

In the following, we identify those cuts that might violate the condition given in Lemma 10 for a (non-strictly) reroutable flow. We then show that this class of cuts forms a semi-lattice. This allows us to apply an uncrossing of the flow paths that iteratively eliminates the problematic cuts while maintaining reroutability.

Bad cuts. Let x be an s-t-flow and let $C \in \mathcal{C}$ be a t-separating cut. An arc $\bar{a} \in C$ is *(x, C)-bad* if there is an arc $a \in C$ and a path $P \in \mathcal{P}_{\bar{a} \to a}$ with $x(P) > 0$. A cut C is *x-bad* if all arcs $\bar{a} \in C$ are (x, C)-bad.

▶ **Lemma 11.** *Let x be a reroutable flow for capacities $u \equiv 1$. Let $C \in \mathcal{C}$ be a t-separating cut. If $\sum_{a \in C}(1 - x(a)) < 1$ then C is x-bad.*

Proof. By contradiction assume C is not x-bad. Then there must be an arc $\bar{a} \in C$ that is not (x, C)-bad. This implies that $\sum_{P \in \mathcal{P}_{\bar{a} \to a}} x(P) = 0$ for every $a \in C \setminus \{\bar{a}\}$. In particular, $\bar{u}_{x,\bar{a}}(a) = \bar{u}_x(a) = 1 - x(a)$ for all $a \in C \setminus \{\bar{a}\}$. Since all flow in the rerouting of x for failure of $\bar{a}$ needs to cross $C \setminus \{\bar{a}\}$, we obtain $\sum_{a \in C \setminus \{\bar{a}\}} \bar{u}_{x,\bar{a}}(a) \geq x(\bar{a})$. Adding $1 - x(\bar{a})$ to both sides of this inequality yields a contradiction. ◀

▶ **Lemma 12.** *Let x be a flow and let $S, S' \in \mathcal{S}$ be such that $\delta^+(S)$ and $\delta^+(S')$ are both x-bad. Then $\delta^+(S \cup S')$ is an x-bad t-separating cut as well.*

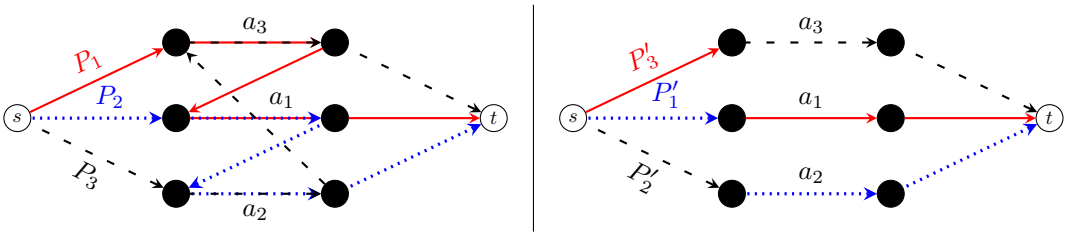

Figure 2 Uncrossing of paths on a bad cut.

Uncrossing paths. Let $P \in \mathcal{P}$. For two nodes $v, w \in V$ visited by P (in that order), we let $P[v, w]$ denote the subpath of path P starting at v and ending at w. Given another path $Q \in \mathcal{P}$ and an arc $a \in P \cap Q$, let $P \times_a Q$ be a simple s-t-path in the concatenation of $P[s, \text{head}(a)]$ and $Q[\text{head}(a), t]$.

▶ **Theorem 13.** *Let x be a reroutable flow for capacities $u \equiv 1$. Then there is a strictly reroutable flow x' with $\text{val}(x') = \text{val}(x)$ and $x'(a) \leq x(a)$ for all $a \in A$.*

Sketch of Proof. If x is not strictly reroutable, then by Lemmas 10 and 11 there must be an x-bad cut. By Lemma 12, there is a "rightmost" x-bad cut $C^* := \delta^+(S^*)$ where S^* is the union of all vertex sets defining x-bad cuts. Because C^* is bad, we obtain flow-carrying paths $P_1, \ldots, P_k$ and arcs $a_1, \ldots, a_k$ such that $a_i \in P_i \cap P_{i+1}$ is the last arc of P_i that crosses C^* for each $i \in [k]$ (with $P_{k+1} := P_1$). See Figure 2 for an illustration.

We uncross these paths by defining $P'_i := P_{i+1} \times_{a_i} P_i$ for $i \in [k]$. We obtain a new flow x' by decreasing the flow on all paths P_i by $\varepsilon := \min_i x(P_i)$ and increasing the flow on paths P'_i by ε for all $i \in [k]$. Observe that $\text{val}(x') = \text{val}(x)$ and $x'(a) \leq x(a)$ for all $a \in A$. We show that x' is also a reroutable flow. To this end, let $\bar{a} \in A$ and let $S \subseteq V \setminus \{t\}$ with $\text{tail}(\bar{a}) \in S$ and define $C := \delta^+(S)$. If $S \not\subseteq S^*$, then C is not x-bad by construction of S^*. In this case, it is easy to show that $\sum_{a \in C \setminus \{\bar{a}\}} \bar{u}_{x',\bar{a}}(a) \geq x'(\bar{a})$. If $S \subseteq S^*$, then a careful analysis shows that $\bar{u}_{x',\bar{a}}(a) \geq \bar{u}_{x,\bar{a}}(a)$ for all $a \in C$. Thus in both cases there is sufficient capacity to reroute flow when $\bar{a}$ fails. We repeat this procedure until we arrive at a strictly reroutable flow. ◀

▶ Remark. The proof of Theorem 13 preserves integrality. More specifically, if $x(P)$ is an integer multiple of α for every $P \in \mathcal{P}$, then x' can be chosen such that also $x'(P)$ is an integer multiple of α for every $P \in \mathcal{P}$.

▶ Remark. The characterization of strictly reroutable flows for unit capacities given in Lemma 10 can be extended to instances with arbitrary capacities. However, in the general case, a non-strictly reroutable flow might not have a bad cut.

4 Computing (half-)integral solutions

In some application contexts, flow cannot be split into arbitrarily small pieces. This is the setting we consider in this section. We say a flow x is *integral*, if $x(P) \in \mathbb{Z}$ for all $P \in \mathcal{P}$. We say that x is *half-integral* if $2x$ is integral.

For many fundamental flow problems, such as Max Flow or Min Cost Flow, integrality comes for free, i.e., as long as capacities are integral, there exists an optimal integral solution. In the case of reroutable flows, this property does not hold, see, e.g., Figure 3. In fact, it turns out to be *NP*-hard to decide whether there is a non-zero integral reroutable flow in a network.

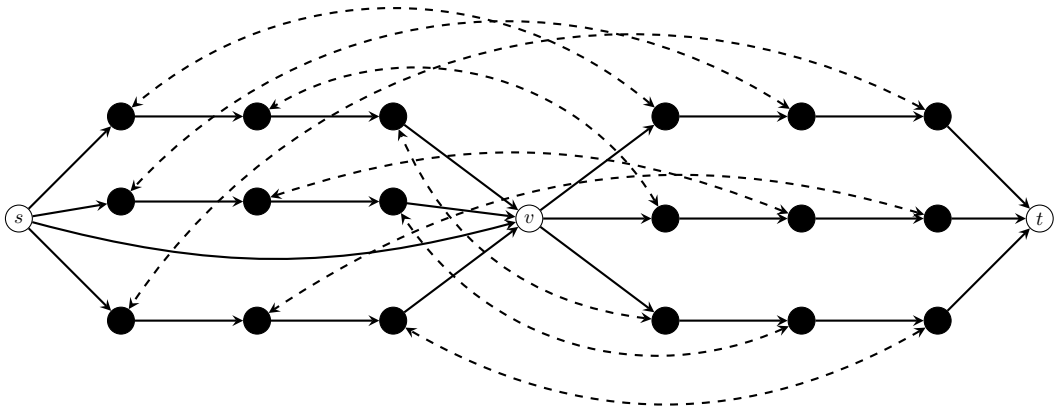

Figure 3 Example network in which no integral or half-integral reroutable flow is optimal. Dashed arcs represent bidirected backup links (see Section 2.4), all arcs have unit capacities. The maximum reroutable flow value is 2. This can only be achieved when $x(s,v) = 1$, the three s-v-paths all carry $1/3$ unit of flow, and the three v-t-paths all carry $2/3$ unit of flow.

▶ **Theorem 14.** *It is NP-hard to decide whether there is an integral (strictly) reroutable flow of value* 1*, even when restricted to instances with* $u \equiv 1$*.*

Note that this problem corresponds to sending a unit of flow along a single s-t-path. The hardness stems from a problem named Forbidden Pairs s-t-Path, which we introduce in Section 5. While it seems that Theorem 14 does not give much space for positive algorithmic results, we can do much better if we relax the integrality requirement slightly.

▶ **Theorem 15.** *Given a network with* $u \equiv 1$*, the algorithm given in Listing1 computes in polynomial time either a half-integral strictly reroutable flow of value* 1*, or correctly determines that no reroutable flow of value* 1 *exists.*

In particular, this implies that if we are interested in sending a single unit of flow, we never need to split our flow in more than two paths. Before we discuss the algorithm from Theorem 15, let us shortly discuss the case of arbitrary capacities. As a consequence of the max flow/min cut result proven in Section 2.3, we obtain the following approximation.

▶ **Theorem 16.** *If* u *is integral, then there is a strictly reroutable half-integral flow* x *with* $\mathrm{val}(x) \geq \mathrm{OPT}/2$*, where* OPT *is the value of a maximum reroutable flow. The flow* x *can be computed in polynomial time.*

Proof. Recall that in the proof of Theorem 7 we computed an s-t-flow x' that was maximal with respect to capacities $u'(a) := \min\{u(a), u(C_a \setminus \{a\})\}$. We then showed that the flow $x := x'/2$ is strictly reroutable and within a factor of 2 of a corresponding R-cut. In particular, $\mathrm{val}(x)$ is within a factor of 2 of the maximum reroutable flow value. Note that if u is integral, also u' is integral, and hence we can choose x' to be integral, ensuring that x is half-integral. ◀

Algorithm for computing a half-integral flow for unit demand

A natural starting point for an algorithm is to identify arcs $a \in A$ such that $\mathrm{tail}(a)$ is disconnected from t in $(V, A \setminus \{a\})$. Obviously, no reroutable flow can send a positive amount

Listing 1 Computing a half-integral reroutable unit demand flow

```
A_0 := ∅, A_1 := ∅
while ∃ a ∈ A \ A_0 : A_1 ∪ {a} is a tail(a)-t-cut in D
  A_0 := A_0 ∪ {a}
  A_1 ← {a' : a' is an s-t-bridge in (V, A \ A_0)}
end while
if A_0 is an s-t-cut in D
  return "No reroutable flow of value 1 exists."
else
  Let P_1, P_2 be two s-t-paths in (V, A \ A_0) such that P_1 ∩ P_2 = A_1.
  Let x be the flow defined by x(P_1) = x(P_2) = 1/2.
  return x
end if
```

of flow along such arcs, as after failure of a, the flow cannot be rerouted to t. Surprisingly, this simple preprocessing step can be generalized to an iterative procedure that solves the problem.

The algorithm, which is formally given in Listing 1, maintains two sets A_0 and A_1. In every iteration, it identifies an arc that cannot carry any flow in any reroutable flow and adds it to A_0. The set A_1 contains the s-t-bridges in the graph $(V, A \setminus A_0)$, i.e., all arcs whose removal disconnects s from t in that graph. Clearly, if $x(a) = 0$ for all $a \in A_0$, then every arc in A_1 must carry 1 unit of flow. If at some point A_0 becomes an s-t-cut, we know that no reroutable flow of value 1 exists. On the other hand, if the algorithm finds no more arcs to add to A_0 while s and t are still connected in $(V, A \setminus A_0)$, it computes two paths P_1, P_2 that only intersect at the bridges, and sends $1/2$ units of flow along each of them.

Proof of Theorem 15. To see that Algorithm 1 terminates in polynomial time, observe that $|A_0|$ is increased in every iteration of the while-loop and the loop thus terminates after at most $|A|$ iterations, each of which can be carried out in polynomial time.

Case 1: No flow exists. We now show that if Algorithm 1 denies the existence of a reroutable flow of value 1, this is indeed correct. By contradiction assume A_0 contains an s-t-cut but there exists a reroutable flow x of value 1. We prove by induction that at any step of algorithm the set A_0 fulfills the property that $x(a) = 0$ for all $a \in A_0$, yielding a contradiction. The claim is clearly true initially, when $A_0 = \emptyset$. Now consider any iteration of the while-loop, considering arc a. By induction hypothesis, every s-t-path P with $x(P) > 0$ must be a path in $(V, A \setminus A_0)$. Note that there is an order $a_1, \ldots, a_\ell$ of the set A_1 of s-t-bridges of $(V, A \setminus A_0)$ such that every such flow-carrying path contains all of these bridges in exactly that order. In particular $x(a_1) = \ldots = x(a_\ell) = 1$. Now consider the next arc a added to A_0 and assume by contradiction that $x(a) > 0$. By choice of a there is a tail(a)-t-cut $C \subseteq A_1 \cup \{a\}$ in D. Note that if $C \cap A_1 = \emptyset$, there is no rerouting of x in case of failure of arc a, as there is no tail(a)-t-path in $(V, A \setminus \{a\})$. Thus, let $a_k \in C \cap A_1$ be the bridge with the highest index k on the cut. We distinguish two cases:

- **(i)** Assume a appears before a_k on every flow-carrying path. Note that C is a tail(a_k)-t-cut because $a_k \in C$ and that $\sum_{a' \in C} \bar{u}_{x,a_k}(a') = 1 - x(a) < 1$. Therefore, the one unit of flow on a_k cannot be rerouted when a_k fails.
- **(ii)** Now assume a occurs after a_k on every flow-carrying path. But then, when a fails, the flow on a cannot be rerouted as all edges in $C \setminus \{a\} \subseteq A_1$ occur before a on every flow-carrying path and thus $\sum_{a' \in C \setminus \{a\}} \bar{u}_{x,a}(a') = 0$.

We thus deduce that $x(a) = 0$, completing the induction.

Case 2: Algorithm returns flow. Finally, we show that if $(V, A \setminus A_0)$ contains an s-t-path after completing the while-loop, then the flow x returned by the algorithm is a strictly reroutable flow. First observe that two s-t-paths P_1, P_2 in $(V, A \setminus A_0)$ with $P_1 \cap P_2 = A_1$ exist by the max flow/min cut theorem, as A_1 contains exactly the bridges of $(V, A \setminus A_0)$. Now consider the failure of any arc $\bar{a} \in A \setminus A_0$. Let C be a tail$(\bar{a})$-t-cut in D minimizing $U(C) := \sum_{a \in C \setminus \{\bar{a}\}} \bar{u}_x(a)$. We show that $U(C) \geq x(\bar{a})$, which by max flow/min cut implies that there is a rerouting of x in case of failure of $\bar{a}$. By termination condition of the while-loop, there is at least one arc $a' \in C \setminus (A_1 \cup \{\bar{a}\})$. Note that $x(a') \in \{0, 1/2\}$ and thus $U(C) \geq 1/2$. If $\bar{a} \notin A_1$, then $x(\bar{a}) \leq 1/2 \leq U(C)$. If $\bar{a} \in A_1$, we distinguish two cases.

(i) If $x(a') = 0$ then $U(C) \geq 1$ and the one unit of flow on $\bar{a}$ can be rerouted.

(ii) If $x(a') = 1/2$, then $a' \notin A_0$. Note that C is a tail(a')-t-cut in D and thus there is $a'' \in C \setminus A_1 \cup \{a'\}$ by termination condition of the while-loop. Note that, because $a'' \notin A_1$, we have $a'' \neq a$ and $x(a'') \leq 1/2$. Thus $U(C) \geq 1$ also in this last case.

We conclude that x is indeed strictly reroutable. ◀

▶ Remark. Note that our proof of Theorem 15 does not make use of Theorem 13. Instead, it gives a simple alternative argument for the equivalence of reroutable and strictly reroutable flows in unit capacity networks, for the special case of unit value flows.

▶ Remark. Theorem 15 implies that, for networks with $u \equiv 1$, if there exists any reroutable flow of value 1, then there exists a half-integral strictly reroutable flow of value 1. The example given in Figure 3, however, reveals that this is no longer true for flows of higher value, as the unique maximum reroutable flow uses paths with flow value 1/3.

5 Hardness results

In this section, we give hardness results for Max RF and some variants of the problem.

Paths avoiding forbidden pairs. Our hardness results are based on reductions from Forbidden Pairs s-t-Path, which is defined as follows: We are given a digraph $D' = (V', A')$, two nodes $s', t' \in V'$, and a set of forbidden arc pairs $\mathcal{F} \subseteq \{\{a, \bar{a}\} : a, \bar{a} \in A\}$. The task is to find an s'-t'-path P that does not contain both arcs of any pair, i.e., $|S \cap P| \leq 1$ for all $S \in \mathcal{F}$. It is not hard to see that Forbidden Pairs s-t-Path is *NP*-hard [12].

5.1 General capacities

▶ **Theorem 3.** Max RF *is NP-hard, even when* $u(a) \in \{1, 2\}$ *for all* $a \in A$.

Sketch of Proof. We construct a gadget that allows us to introduce forbidden pairs for flow paths in the network. Starting with an instance of Forbidden Pairs s-t-Path, we append a sequence of parallel length-2 paths (g_i, h_i) and $(\bar{g}_i, \bar{h}_i)$, one pair of paths for each pair $\{a_i, \bar{a}_i\} \in \mathcal{F}$, leading from t' to the new sink t. For each i, we connect a_i and h_i, and $\bar{a}_i$ and $\bar{h}_i$, respectively, with bidirected backup links. The construction is depicted in Figure 4. In a reroutable s-t-flow of value 2, both h_i and $\bar{h}_i$ are saturated. When a_i fails, the only path for rerouting leads via h_i, hence all flow that traverses a_i must be on paths in $\mathcal{P}_{a_i \to h_i}$. Likewise all flow that traverses $\bar{a}_i$ must be on paths in $\mathcal{P}_{\bar{a}_i \to \bar{h}_i}$. As $\mathcal{P}_{h_i} \cap \mathcal{P}_{\bar{h}_i} = \emptyset$, no flow-carrying path can use both a_i and $\bar{a}_i$ for any i. Thus if a reroutable flow of value 2 exists, there is a path avoiding the forbidden pairs. For the converse of this argument, it is important that $u(a_i) = u(\bar{a}_i) = 2$. This allows for a rerouting when h_i or $\bar{h}_i$ fails. ◀

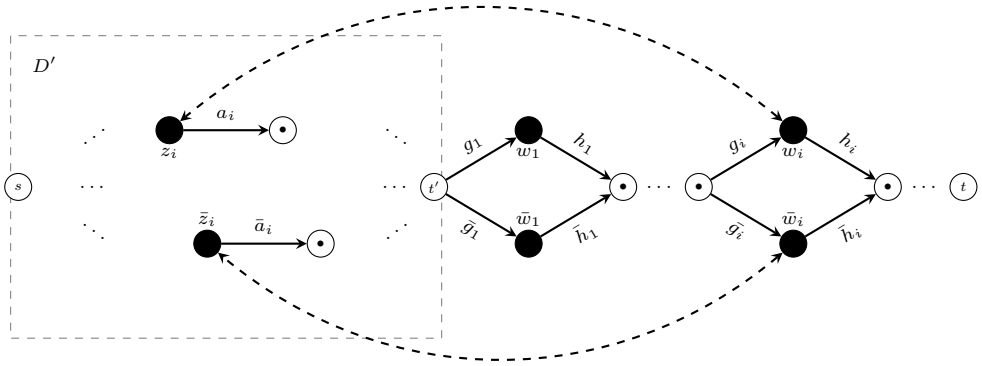

Figure 4 Construction for the proof of Theorem 3. The dashed box contains the graph D' from the Forbidden Pairs s-t-Path instance. The arcs $a_i, \bar{a}_i$ have capacity 2 for all i, all other arcs have unit capacity. In a reroutable flow of value 2, the arcs h_i and $\bar{h}_i$ must be saturated for all i. Any rerouting for a_i has to traverse h_i and any rerouting for $\bar{a}_i$ has to traverse $\bar{h}_i$.

5.2 Multiple arc failures

A natural generalization of Max RF and Max SRF allows multiple simultaneous arc failures. When a set of arcs S fails, flow is interrupted where it first encounters an arc from S and has to be rerouted from that point to the sink. A flow is (strictly) k-reroutable, if there is a rerouting for any failure of a set $S \subseteq A$ with $|S| \leq k$. We denote the corresponding problem of finding a (strictly) k-reroutable flow of maximum value by Max (Strictly) k-Reroutable Flow. It turns out that dealing even with only 2 arc failures in unit capacity networks is *NP*-hard in both cases.

▶ **Theorem 17.** Max (Strictly) k-Reroutable Flow *is NP-hard, even when restricted to instances with* $k = 2$ *and* $u \equiv 1$.

Acknowledgments. We thank David Adjiashvili and Marco Senatore for helpful discussions.

References

1 David Adjiashvili, Gianpaolo Oriolo, and Marco Senatore. The online replacement path problem. In *Algorithms – ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2013.
2 Charu C. Aggarwal and James B. Orlin. On multiroute maximum flows in networks. *Networks*, 39(1):43–52, 2002.
3 Y. P. Aneja, R. Chandrasekaran, and K. P. K. Nair. Maximizing residual flow under an arc destruction. *Networks*, 38(4):194–198, 2001.
4 Dimitris Bertsimas, Ebrahim Nasrabadi, and James B. Orlin. On the power of randomization in network interdiction. *Operations Research Letters*, 44(1):114–120, 2016.
5 Dimitris Bertsimas, Ebrahim Nasrabadi, and Sebastian Stiller. Robust and adaptive network flows. *Operations Research*, 61:1218–1242, 2013.
6 Graham Brightwell, Gianpaolo Oriolo, and F. Bruce Shepherd. Reserving resilient capacity in a network. *SIAM Journal on Discrete Mathematics*, 14(4):524–539, 2001.
7 Chandra Chekuri, Anupam Gupta, Amit Kumar, Joseph Naor, and Danny Raz. Building edge-failure resilient networks. *Algorithmica*, 43(1-2):17–41, 2005.

8 Stephen R. Chestnut and Rico Zenklusen. Hardness and approximation for network flow interdiction. *Networks*, 2017.

9 Amaro de Sousa and Gil Soares. Improving load balance and minimizing service disruption on ethernet networks with IEEE 802.1 S MSTP. In *Workshop on IP QoS and Traffic Control*, pages 25–35, 2007.

10 Yann Disser and Jannik Matuschke. The complexity of computing a robust flow, 2017.

11 Lester R. Ford and Delbert R. Fulkerson. *Flows in networks*. Princeton Univ. Press, 1962.

12 Harold N. Gabow, Shachindra N. Maheshwari, and Leon J. Osterweil. On two problems in the generation of program test paths. *IEEE Transactions on Software Engineering*, SE-2(3):227–231, 1976.

13 Fabrizio Grandoni, Gaia Nicosia, Gianpaolo Oriolo, and Laura Sanità. Stable routing under the spanning tree protocol. *Operations Research Letters*, 38(5):399–404, 2010.

14 Alan J. Hoffman. A generalization of max flow—min cut. *Mathematical Programming*, 6(1):352–359, 1974.

15 Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, 1999.

16 Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, 1976.

17 Steven J. Phillips and Jeffery R. Westbrook. Approximation algorithms for restoration capacity planning. In *Algorithms – ESA'99*, volume 1643 of *Lecture Notes in Computer Science*, pages 101–115. Springer, 1999.

18 F. Bruce Shepherd. Single-sink multicommodity flow with side constraints. In *Research Trends in Combinatorial Optimization*, pages 429–450. Springer, 2009.