Technische Universität München

Department of Mathematics

Variations of the robust network flow problem

Master's Thesis
by
Alexander Eckl

Supervisor:	Prof. Dr. Andreas S. Schulz
Advisor:	Dr. Jannik Matuschke
Submission date:	March 15, 2018

I hereby declare that this thesis is my own work and that no other sources have been used except those clearly indicated and referenced.

Unterschleißheim, March 15, 2018

Alexander Eckl

Abstract

The topic of this Master's thesis is robust optimization in the scope of uncertainty, in particular robust and reroutable flows in networks with failing arcs. In the well-known Maximum Robust Flow problem [DM17], a flow is sent over a directed graph and is then interrupted by the destruction of a known number of arcs. All flow on paths leading through the destroyed arcs is lost. The objective is to maximize the remaining flow after failure of the arcs. We introduce a variation of this problem called Maximum Scaled Robust Flow, where the amount of destroyed flow is scaled with a linear factor. The problem is polynomial if only a single arc is allowed to fail, and NP-hard if the number of failing arcs is unknown. For small absolute values of the scaling factor, all solutions to the problem are maximum network flows. If the scaling factor is larger than a number depending on the connectivity of the underlying graph, the zero-flow is the unique optimal solution. Tightness examples are presented for the proven bounds. A variation with affine scaling is polynomially solvable if only one arc gets destroyed. The same holds for a version defined on arc-based flows for a fixed number of destroyed arcs.

In the Maximum Reroutable Flow problem [MMO17], all flow lost to a single failing arc must be rerouted locally in the graph using available capacities. The objective value of the problem is given by the value of the original flow. The strict version of the problem, Maximum Strictly Reroutable Flow, makes more restrictive assumptions on the capacities available for rerouting. We introduce the variations Maximum Optional Reroutable Flow and Maximum Strictly Optional Reroutable Flow, where rerouting after failure of the arc is possible, but not mandatory. The objective value is computed by determining the arc that yields the worst-case flow value after failure and optional rerouting. We prove that the strict problem can be solved in polynomial time and that there exists a 2-approximation for the non-strict version. Compared to the original Maximum Reroutable Flow problem, making rerouting optional can only improve the objective value of the problem. We prove that Maximum Robust Flow always has a smaller objective value than Maximum Optional Reroutable Flow and that there exists a polynomial reduction from the former problem to the latter. Finally, we solve Maximum Optional Reroutable Flow and its strict version on unit capacity networks, giving an explicit objective value.

Zusammenfassung

Thema dieser Masterarbeit ist robuste Optimierung im Rahmen von Unsicherheit, insbesondere robuste und umleitbare Flüsse in Netzwerken mit ausfallenden Kanten. Im bekannten Problem Maximum Robust Flow [DM17] wird ein Fluss durch einen gerichteten Graphen geschickt und dann durch die Zerstörung einer bekannten Anzahl von Kanten unterbrochen. Der gesamte Fluss auf Pfaden, die durch zerstörte Kanten führen, geht verloren. Ziel ist es, den übrig gebliebenen Fluss nach Ausfall der Kanten zu maximieren. Wir präsentieren eine Variation dieses Problems namens Maximum Scaled Robust Flow, bei dem die Höhe an zerstörtem Fluss mit einem linearen Faktor skaliert wird. Das Problem ist polynomiell lösbar, wenn nur eine einzige Kante ausfällt, und NP-schwer, wenn die Zahl an ausfallenden Kanten unbekannt ist. Für kleine Beträge des Skalierungsfaktors sind alle Lösungen des Problems maximale Netzwerk-Flüsse. Wenn der Skalierungsfaktor größer ist als eine Zahl abhängig von der Konnektivität des zu Grunde liegenden Graphen, dann ist der Null-Fluss die einzige optimale Lösung. Durch Beispiele wird gezeigt, dass die bewiesenen Schranken scharf sind. Eine Variation mit affiner Skalierung ist polynomiell lösbar, wenn nur eine Kante zerstört wird. Das Gleiche gilt für eine Version, die auf kantenbasierten Flüssen für eine feste Anzahl von zerstörten Kanten definiert ist.

Beim Maximum Reroutable Flow-Problem [MMO17] muss der gesamte Fluss, der an den Ausfall einer einzigen Kante verloren geht, lokal im Graphen mit Hilfe der verfügbaren Kapazitäten umgeleitet werden. Der Zielfunktionswert des Problems ist durch den Wert des ursprünglichen Flusses gegeben. Die strikte Version des Problems, Maximum Strictly Reroutable Flow, gibt strengere Bedingungen an die für das Umleiten verfügbaren Kapazitäten vor. Wir führen die Variationen Maximum Optional Reroutable Flow und Maximum Strictly Optional Reroutable Flow ein, bei denen Umleiten nach dem Ausfall der Kante möglich, aber nicht zwingend notwendig ist. Der Zielfunktionswert wird durch die Kante bestimmt, die nach Ausfall und optionalem Umleiten den schlechtesten Flusswert liefert. Wir beweisen, dass das strikte Problem in polynomieller Zeit lösbar ist und dass eine 2-Approximation für das nicht-strikte Problem existiert. Das Umleiten optional zu machen, verbessert im Vergleich zum ursprünglichen Problem, Maximum Reroutable Flow, den Zielfunktionswert. Wir beweisen, dass Maximum Robust Flow immer einen kleineren Zielfunktionswert als Maximum Optional Reroutable Flow hat und dass es eine polynomielle Reduktion von ersterem zu letzterem gibt. Schließlich lösen wir Maximum Optional Reroutable Flow und dessen strikte Version auf Einheitskapazitäts-Netzwerken, indem wir einen expliziten Zielfunktionswert angeben.

Contents

1 Introduction

In an advanced, interconnected society, network infrastructures are an important foundation of modern life. Networks are present everywhere, from communication systems like the internet to transportation systems such as the postal service or public transport. Electricity flows over vast distances across multiple relays to its destination, billions of users connect on sociel networks everyday and navigation systems guide cars from one city to the next.

Unfortunately, all these networks are prone to failure or intentional sabotage. A storm may obstruct air traffic in certain areas, or hackers may attack webpages. In these cases it is desirable to restore failing connections or redirect traffic over the network as quickly as possible.

The mathematical notion of *network flows* is able to encompass most varieties of modern networks. Flow is sent on a directed graph from the designated source to a pre-defined target, subject to capacity constraints on all arcs. For an introduction to network flows, see for example [AMO89], [GTT89] or [CLRS09, Chapter 26].

To capture the failure or sabotage of networks, so-called *robust flows* are used. After declaration of the flow, certain arcs may be destroyed causing all value on them to be substracted from the objective function. Robust flows provide methods to model situations in which the damage and effects of such failures are to be minimized. One basic problem within this framework is the Maximum Robust Flow problem, in which the goal is to maximize the amount of remaining flow after an attack on a number of k arcs in the network. Previous work includes [ACN01], [DC07] and [DM17]. An approximation algorithm was developed by [BNS13] and improved by [BNO16]. A closely related problem is *network flow interdiction*, where the task is to find a set of arcs in a network minimizing the remaining flow after failure of these arcs. For an overview on results regarding this problem, see [CZ17].

In different applications it may be vital to save some or all of the flow lost to failing links. In this case, *reroutable flows* can be employed to model the situation. Reroutable flows utilize alternative paths through the network whenever arcs fail. These reroutings are usually handled locally at the point of failure without affecting other flow paths. The Maximum Reroutable Flow problem investigated by [MMO17] requires the entire lost flow to be rerouted in the given network if any single arc fails. Similar problems include *adaptive flows* [BNS13] and local rerouting schemes focused on minimizing capacity costs [CGK+02],[BOS01].

Robust flows and reroutable flows fall into the framework of *uncertainty*. Two principal methods have been introduced to address data uncertainty in optimization [BS03]: stochastic programming and robust optimization. The first approach models uncertainty by assuming the data occurs with varying probabilities, one of the earliest

works in this direction is [Dan55]. Our problems, on the other hand, are based on robust optimization, where the worst-case problem instances are optimized using min-max objective functions. Research in this framework includes for example [BS03] and [GKL+17]. For an overview of recent research in robust optimization, confer [GMT14].

There exist two basic approaches for robust flow optimization on networks: path-based and arc-based flows [MMO17]. In this thesis, we focus on the path-based approach, but give a few results for arc flows as well.

The aim of this thesis is to introduce the problems Maximum Robust Flow and Maximum Reroutable Flow as well as to analyze some basic hardness results. Afterwards different variations of both problems are established. The most important ones of these are Maximum Scaled Robust Flow, which introduces a scaling factor on the amount of lost flow, and Maximum Optional Reroutable Flow, in which rerouting flow after failure is possible, but not mandatory. Finally, all occuring problems are to be compared with respect to objective values and hardness.

The thesis is structured in the following way: In chapter 2, basic definitions and results important for later chapters are introduced. In chapter 3, we present the Maximum Robust Flow problem as well as different variations including Maximum Scaled Robust Flow. We establish LP formulations and hardness results for both problems and talk about characteristics of solutions for the scaled version. Finally, we take a look at some examples and non-linear scaling functions. Chapter 4 deals with reroutable flows: The Maximum Reroutable Flow problem is introduced in both its normal and strict version. For both of these, we present the variation Maximum Optional Reroutable Flow. We give LP formulations for all problems and highlight some hardness results as well as an approximation algorithm for the strict problem. Additionally, we show some results for unit capacity networks. The presented problems are compared in regards to their objective values as well as their hardness. Finally, we conclude the thesis by talking about open questions and further research in chapter 5.

2 Notation and preliminaries

In this chapter we introduce basic notions of optimization and graph theory necessary for the problems discussed in the thesis. Several graph-theoretical results are established, which will be useful for statements in chapters 3 and 4. Lastly, basic complexity theory and the equivalence between optimization and separation are presented.

2.1 Basic notation and definitions

The graph-theoretical notations and definitions introduced in this section are based on [Die16].

In all our problems we are given a directed multigraph $G = (V, E)$, where V is the set of nodes and the arcs are given by a map $E \to V \times V$. The first and second entry of an element of $V \times V$ describe the *tail* and *tip* of an arc respectively. We do not allow for loops in the graph. For convenience reasons, we write $e = (u, v)$ instead of $e \mapsto (u, v)$.

The graph G has two distinct vertices s and t called *source* and *target*. Every arc has a corresponding *capacity* denoted by the function $u : E \to \mathbb{R}_0^+$. All these notions together define the *network* $N = (G, s, t, u)$.

A *path* P in G is a set of arcs of the form $P = \{(u_0, u_1), (u_1, u_2), \dots, (u_{k-1}, u_k)\}$, where $u_i \in V$ for all $i \in [k]$. The *length* of P is equal to k, i.e. the number of arcs in the path. P is called *simple* if all u_i are distinct. In this case, P contains no cycles. If $v = u_0$ and $w = u_k$, we also refer to P as a v-w-path. The network N is called *trivial*, if there are no s-t-paths in N.

Let $\mathcal{P}$ denote the set of simple s-t-paths in G. We define $\mathcal{P}_e = \{P \in \mathcal{P} : e \in P\}$ as the set of such paths incident to the arc e. Equivalently, $\mathcal{P}_S = \{P \in \mathcal{P} : P \cap S \neq \emptyset\}$ is the set of paths incident to $S \subset E$. For any node v in the graph, the set of arcs with tip v is denoted by $E_{\text{in}}(v)$. Similarly, the set of arcs with tail v is $E_{\text{out}}(v)$. This definition can be extended to node sets in the natural way. For $A \subset V$, the set of arcs with tip in A and tail in $V \setminus A$ is denoted by $E_{\text{in}}(A)$. Analogously, the set of arcs with tail in A and tip in $V \setminus A$ is $E_{\text{out}}(A)$.

A *path-based s-t-flow* x in the network N is defined by $x : \mathcal{P} \to \mathbb{R}_0^+$, where $x(P)$ determines the flow on a specific path $P \in \mathcal{P}$. The total amount of flow on an arc is denoted by $x(e) = \sum_{P \in \mathcal{P}_e} x(P)$. A *feasible* flow must fulfill $x(e) \leq u(e)$ for all $e \in E$. The *value* of the flow is determined by $\text{val}(x) = \sum_{P \in \mathcal{P}} x(P)$.

Flow conservation is fulfilled at node v if $\sum_{e \in E_{\text{in}}(v)} x(e) = \sum_{e \in E_{\text{out}}(v)} x(e)$. The definition of path-based flows satisfies flow conservation at all nodes except s and t.

For *arc-based s-t-flows* we introduce an auxiliary arc $h = (t, s)$ with $u(h) = \infty$. A flow f is then defined by $f : E \cup \{h\} \to \mathbb{R}_0^+$, where $f(e)$ determines the flow on an arc e. To be feasible, flows must fulfill capacity constraints at all arcs and flow conservation at all nodes including s and t. The value of the flow is given by $\text{val}(f) = f(h)$.

It is easy to see that any path-based flow x can be turned into an arc-based flow f of the same value by simply setting $f(e) = x(e)$, $\forall e \in E$ and $f(h) = \text{val}(x)$. To turn an arc-based flow back into a path-based flow, refer to the path decomposition discussed in Proposition 2.7. In this thesis, a flow is always path-based unless specified otherwise.

Definition 2.1. *The problem of finding a feasible flow in N with maximum value is called* Maximum Network Flow *and is denoted by NF.*

As a next step, we want to define cuts in the network N. Let $A \subset V$ such that $s \in A$ and $t \in V \setminus A$. Then, $C = (A, V \setminus A)$ is called an *s-t-cut*. The *capacity* of C is defined by $\text{cap}(C) = \sum_{e \in E_{\text{out}}(A)} u(e)$.

This definition naturally leads to the formulation of the following optimization problem.

Definition 2.2. *The problem of finding a cut in N with minimum capacity is called* Minimum Network Cut.

Since any flow sent from s to t must at some point use the arcs of a given cut, it is obvious that in any network the optimal value of NF is smaller than or equal to the capacity of a minimum cut. The max-flow min-cut theorem asserts the tightness of this bound, see Theorem 2.3.

Generally, we adhere to the following conventions regarding notation throughout the thesis: Networks and graphs are denoted by capital letters N and G respectively. Graphs are defined by their node set V and arc set E. The letters v,w and sometimes u stand for single nodes while e stands for single arcs. In the context of network definitions, u also stands for arc capacities. Nodes s and t are the source and target of a network. Path-based flows are denoted by x and arc-based flows by f. The capital letter C stands for cuts. Paths in networks are usually denoted by P.

Abbreviations for optimization problems are short capital letter acronyms, e.g. NF or ORF. The operator val(.) stands for the value of an instance of a problem (usually a flow), where sometimes an index determines the corresponding problem. For any occuring optimization problem P in this thesis, $\text{opt}_P(N)$ is the value of an optimal solution to the problem on the network N.

A polyhedron is indicated by the capital letter Q. In section 4.5, a capital M sometimes stands for an arbitrarily large natural number. Letters y and z usually denote dual variables.

2.2 Graph-theoretical results

One of the most important statements in network flow theory is the so-called max-flow min-cut theorem. It was proven by Ford and Fulkerson [FF56] in 1956.

Theorem 2.3 (Max-Flow Min-Cut)**.** *Given a network $N = (G, s, t, u)$, the maximum value of a flow is equal to the minimum capacity of a cut.*

A proof can be found in [Die16, Thm. 6.2.2] or directly in [FF56].

This theorem will be used multiple times throughout this thesis. One particular application lies in the following results on path connectivity.

When talking about network flows, it is very important to think about the structure of the underlying graph. In particular it is interesting by how many different paths the two nodes s and t are connected. We introduce the notion of *s-t-path-connectivity* to measure the number of paths connecting s and t.

Definition 2.4. *Let $N = (G, s, t, u)$ be a network, $l \in \mathbb{N}$. We say s and t are l-path-connected if for all sets $F \subset E$ with $|F| < l$ the graph $G - F := (V, E \setminus F)$ has an s-t-path.*

The greatest integer l such that s and t are l-path-connected is the s-t-path-connectivity $\pi(G)$ of the graph G.

To be able to understand this definition more naturally, let us consider the notion of independent paths.

Definition 2.5. *A set of simple s-t-paths in G is called (arc-)independent if $P \cap Q = \emptyset$ for any two paths P and Q in the set.*

In other words, a set of s-t-paths is independent if the paths do not share any arcs. Note that the paths may share nodes, however. Whenever a set of s-t-paths is called (arc-)independent, we always assume that all paths in the set are simple.

The definition of independence is especially interesting for robust flows, which will be introduced in chapter 3. The following lemma establishes the connection between independence and s-t-connectivity.

Lemma 2.6. *$N = (G, s, t, u)$ a network. Then the maximum size of an independent set of s-t-paths is equal to $\pi(G)$.*

Proof. Let m denote the maximum size of an independent set of s-t-paths in G. We define a new capacity function u^* by setting $u^*(e) = 1$ for all $e \in E$.

Consider the maximum flow in the network (G, s, t, u^*). Since there is a set of m independent s-t-paths, we can send at least m units of flow from s to t. Assume we could send more than m. Since the capacity of all arcs is 1, we would need at least $m+1$ independent s-t-paths to send this flow, resulting in a contradiction. Hence the maximum flow with respect to u^* has value m. By the max-flow min-cut theorem, there exists a minimum s-t-cut $C = (A, V \setminus A)$ with capacity m.

By definition of the capacity of a cut, the sum of capacities of outgoing arcs from A is equal to m. Since $u^*(e) = 1$ for all $e \in E$, this means the number of arcs from A to $V \setminus A$ is m. If we remove all these arcs from G, there cannot remain any s-t-path because C is a cut. Hence by definition $\pi(G) \leq m$.

For the inverse inequality, consider any set $F \subset E$ with $|F| < m$. Assume $G - F$ does not contain an s-t-path. In other words, removing F disconnects s and t. This yields a cut with value less than m, which is a contradiction to the minimality of C. Hence $G - F$ must contain an s-t-path and thus $\pi(G) \geq m$. □

It will be useful later to be able to convert path-based flows into arc-based flows and back. As mentioned already, converting from path-based to arc-based is simple. For the reverse direction, we will use path decomposition as defined in the following proposition.

Proposition 2.7 (Path Flow Decomposition)**.** *$N = (G, s, t, u)$ a network, f an arc-based flow in N. Then there exists a collection of feasible flows $f_1, \ldots, f_k$ and simple s-t-paths $p_1, \ldots, p_k$, such that:*

1. *$k \leq |E|$*
2. *$val(f) = \sum_{i=1}^{k} val(f_i)$*
3. *f_i sends positive flow only on arcs of path p_i*
4. *Let the path-based flow x be defined as follows:*

$$x(P) = \begin{cases} val(f_i) & \text{if } P = p_i \text{ for some } i \in [k] \\ 0 & \text{else} \end{cases}$$

Then $val(x) = val(f)$ and $\forall e \in E : f(e) \geq x(e)$.

The flow x constitutes a path-based flow in G with value equal to f, where the number of positive paths is at most $|E|$ and all arcs carry at most as much flow as before.

Proof. This proof expands on the proof of [Tre11, Lemma 11.2]. Let f be an arc-based flow in the network N. We construct the flows f_i and paths p_i using the following procedure:

Set $i = 1, r = f$
While val$(r) > 0$:
  Find a simple s-t-path p_i using only arcs e with $r(e) > 0$
  Let $f_{\min} = \min_{e \in p_i} r(e)$
  Define $f_i(e) = f_{\min}, \ \forall e \in p_i$ and $f_i(e) = 0, \ \forall e \in E \setminus p_i$
  Set $r(e) = r(e) - f_i(e), \ \forall e \in E$
  Set $i = i + 1$
Set $k = i - 1$

The residual flow r can be thought of as the remaining flow that has to be broken down into paths. We first prove that, while r still has positive value, there always exists a simple s-t-path as required in the procedure.

Claim 2.8. *If val$(r) > 0$ in step i, then there exists a simple s-t-path p_i that only uses arcs e with $r(e) > 0$.*

Proof. By contradiction. Assume such a path does not exist and call A the set of nodes reachable from s using only arcs e with $r(e) > 0$. Then $s \in A, t \notin A$.

This means $(A, V \setminus A)$ is an s-t-cut in G and the total amount of flow leaving A is equal to $\mathrm{val}(r)$. Yet by construction of A, all arcs $e \in E_{\mathrm{out}}(A)$ must fulfill $r(e) = 0$, and hence the total amount of flow leaving A is 0. Thus $\mathrm{val}(r) = 0$, giving a contradiction. □

The following claim ensures the feasibility of all flows involved and introduces two invariants that will be needed later.

Claim 2.9. *After step i of the algorithm, r and f_i are feasible flows and the following invariants hold:*

1. *val(f) = val(r) + val(f_1) + $\cdots$ + val(f_i)*
2. *$\forall e \in E : f(e) = r(e) + f_1(e) + \cdots + f_i(e)$*

Proof. By induction. For $i = 0$, that is before the first iteration of the procedure, we trivially have $\mathrm{val}(f) = \mathrm{val}(r)$ as well as $f(e) = r(e)$ for all arcs. Trivially, r is feasible since f is feasible.

Let us consider step $i > 0$. $f_{\min}$ and f_i are defined as in the procedure above. f_i is feasible, since $0 \leq f_i(e) = f_{\min} \leq r(e) \leq f(e) \leq u(e)$ for $e \in p_i$ and $f_i(e) = 0$ for all other arcs. We set $f_i(h) = f_{\min}$. Then flow conservation is trivially fulfilled at all nodes of the path p_i and no other nodes carry any flow w.r.t. f_i.

By induction, the remaining flow r was feasible before the i-th iteration. We now decrease r by $f_{\min}$ on all $e \in p_i$. r is still feasible, because before reducing we had $r(e) \geq f_{\min}$ and thus $r(e) \geq 0$ afterwards. For all nodes v on path p_i, there is exactly one incoming and one outgoing arc that is part of p_i. Both are reduced by $f_{\min}$, hence flow conservation is still fulfilled after updating r. Flow running through any nodes that are not on p_i is unchanged.

After updating, we have decreased the value of r by $f_{\min}$ and have gained a new flow f_i with $\mathrm{val}(f_i) = f_{\min}$. All arcs on p_i have been updated accordingly. Using the induction hypothesis, this gives

$$\mathrm{val}(f) = \mathrm{val}(r) + \mathrm{val}(f_1) + \cdots + \mathrm{val}(f_i)$$

as well as

$$\forall e \in E : f(e) = r(e) + f_1(e) + \cdots + f_i(e).$$

□

It remains the be shown that the procedure terminates.

Claim 2.10. *The while loop in the procedure terminates after at most $|E|$ steps.*

Proof. In every step of the procedure, there exists an arc whose entire remaining flow $r(e)$ gets decreased to 0, namely one of the arcs that had minimal amount of flow among the arcs of p_i.

After $|E|$ steps, if $\text{val}(r)$ has not been reduced to 0 before that, there cannot remain any arc with $r(e) > 0$, hence $\text{val}(r) = 0$ must hold and the while condition evaluates to false.

If $\text{val}(r) = 0$ before the $|E|$-th step, then the loop terminates trivially. □

By Claim 2.10, we have $k \leq |E|$ and by the first invariant of Claim 2.9 we have $\text{val}(f) = \sum_{i=1}^{k} \text{val}(f_i)$ after the termination of the procedure. The third property of Proposition 2.7 follows from the construction of the flows f_i.

To proof point 4 of Proposition 2.7, we take a look at the second invariant of Claim 2.9. It might happen that, after termination of the procedure, there are still arcs e with $r(e) > 0$ even though $\text{val}(r) = 0$. This occurs when the paths p_i are constructed in a way such that there remain cycles in G carrying positive amount of flow. By our definition of the value of arc-based flows, these cycles do not add to the value of r.

Considering this special case, we follow from invariant 2 for all arcs $e \in E$:

$$f(e) = r(e) + \sum_{i=1}^{k} f_i(e) \geq \sum_{i=1}^{k} f_i(e) = x(e)$$

The flow x fulfills all arc capacities and has value $\text{val}(x) = \sum_{i=1}^{k} \text{val}(f_i) = \text{val}(f)$ by invariant 1.

This concludes the proof of Proposition 2.7. □

2.3 Flow augmentation

This section aims to prove that the problem NF from Definition 2.1 is solvable in polynomial time. For that we first need to define the notion of the *residual network*:

Definition 2.11. *Let $N = (G, s, t, u)$ be a network, f an arc-based flow in N. The residual network $N_f = (G_f, s, t, u_f)$ is defined via the graph $G_f = (V, E_f)$, where all arcs $e = (v, w) \in E$ are replaced by $e_1 = (v, w)$ and $e_2 = (w, v)$ in E_f. The capacities are defined as follows:*

$$u_f(e_1) = u(e) - f(e)$$
$$u_f(e_2) = f(e)$$

Note that always two arcs are added to G_f for every arc in G as opposed to other common definitions, where arcs between the same two nodes are often combined. We do this because of certain proofs in chapter 3, where this construction is necessary for the argumentation to work.

An *augmenting path* P w.r.t. the flow f is a path in N_f that only contains arcs with positive capacity. Without loss of generality, an augmenting path is always assumed to be simple. If P uses an arc e_1 for some $e \in E$, then e is a *forward arc* of P, if it uses e_2 instead, e is a *backward arc* of P. The *value* of an augmenting path is the smallest capacity among its arcs.

Theorem 2.12. *A flow f is maximum if and only if there exist no augmenting paths in N_f.*

Proof. Let f be a maximum flow and assume there exists an augmenting path P in N_f of value x^+. Then, starting at s, we can send x^+ units of flow along P to t in the original network N. If P uses a forward arc in N_f, then we increase the flow on the corresponding arc in N by x^+. If P uses a backward arc in N_f, then we decrease the flow on the corresponding arc in N by x^+. Changing flow values in N like this does not violate any capacities because of our choice of the capacities of the residual network and by the fact that the value of P is the smallest capacity among its arcs. In total, the amount of flow from s to t gets increased by x^+ and hence the value also gets increased by x^+. But this means f was not maximal, a contradiction.

Now assume f is a flow in N such that there exists no augmenting path in N_f. Let A be the set of nodes reachable from s in N_f using only arcs with positive capacities. Since there exists no augmenting path, we have $s \in A$ and $t \in V \setminus A$ and therefore $C = (A, V \setminus A)$ is an s-t-cut in N. Because there are no arcs with positive capacity leaving A in N_f, the incoming flow of A on N is 0. Because of the same reason, all arcs leaving A in N must be fully saturated, i.e. $f(e) = u(e)$, $\forall e \in E_{\text{out}}(A)$. Hence $\text{val}(f) = \sum_{e \in E_{\text{out}}(A)} f(e) = \sum_{e \in E_{\text{out}}(A)} u(e) = \text{cap}(C)$. But then by the max-flow-min-cut Theorem 2.3, f must be a maximum flow. □

Based on Theorem 2.12, we can now build a polynomial algorithm for NF. For a given flow f, check if N_f contains any augmenting paths. If not, f is optimal. If there exists an augmenting path P, augment along P as seen in the proof.

This method was first introduced by Ford and Fulkerson [FF56] in the same paper as their max-flow-min-cut theorem. Unfortunately, since they did not specify which augmenting path to choose if multiple are available, the running time of the method depended on the size of a maximum flow in the graph, which can be arbitrarily high.

Edmonds and Karp [EK72] improved the method 16 years later by using the shortest available augmenting path. Their approach has a polynomial running time depending only on the size of the underlying graph.

Corollary 2.13. Maximum Network Flow *is solvable in polynomial time.*

A proof can be found in [CLRS09, Theorem 26.8] or directly in [EK72].

2.4 Equivalence of optimization and separation

In the hardness proofs of chapters 3 and 4 we will need the equivalence of optimization and separation, a result first proven by Grötschel, Lovász and Schrijver [GLS88]

in 1988. They asserted that optimization over a given polyhedron is polynomially equivalent to separation over the same polyhedron.

Formally, the optimization and separation problems are defined in [DM17] as follows:

Definition 2.14. *Given a polyhedron $Q \subset \mathbb{R}^n$ and a vector $c \in \mathbb{R}^n$, the problem of Optimization is to either*

- *state $Q = \emptyset$,*
- *find vectors $x, d \in \mathbb{R}^n$ such that $c^\top d > 0$ and $x + \alpha d \in Q$ for all $\alpha > 0$,*
- *or find an $x \in Q$ maximizing $c^\top x$.*

Definition 2.15. *Given a polyhedron $Q \subset \mathbb{R}^n$ and a vector $y \in \mathbb{R}^n$, the problem of Separation is to either*

- *state $y \in Q$,*
- *or find a vector $d \in \mathbb{R}^n$ such that $d^\top x < d^\top y$ for all $x \in Q$.*

These definitions now enable us to state the equivalence theorem.

Theorem 2.16 ([GLS88, Theorem 6.4.9])**.** *For a rational polyhedron $Q \in \mathbb{R}^n$, either one of the above problems is solvable in oracle-polynomial time if given an oracle for the other problem.*

Note that this theorem only applies to rational polyhedra, i.e. polyhedra where the describing system of inequalities has rational coefficients only. For our application this means we may only consider rational values for certain parameters when using this result.

2.5 Complexity theory

During the analysis of our problems, we will often talk about their difficulty in the sense of how long it takes to solve them. Basic notions in complexity theory exist to quantify and compare the runtimes of problems. This section is intended to give a short, non-formal introduction into these matters. For a more thorough overview, confer [AB09]. All definitions and results in this section, unless otherwise indicated, stem from [AB09].

To be able to talk about complexity, we need a basic computational model. Usually, *Turing Machines* are used for this. Informally, a Turing Machine consists of a register with various states and a read/write-head moving over multiple tapes which contain data in the form of symbols from a pre-defined set. When the head reads an input from the tapes, the machine decides based on this input whether to update the data on the current position and whether to move the head right, left or not at all. Turing

Machines can compute functions by receiving data on a special input tape and return the value of the function on an output tape.

A *decision problem* or *language* is a set of instances of a given problem, formally defined over a string of symbols, where the objective is to decide whether a given instance belongs to the set or not. A Turing Machine *decides* such a problem, if it computes its characteristic function over the set of all inputs.

Definition 2.17. *The complexity class P of all polynomially solvable problems is defined as the set of all languages which can be decided by a Turing Machine that runs in polynomial time.*

A Turing Machine is called *non-deterministic*, if for some state of the tapes and register the transition function is not uniquely determined. Such a Turing Machine accepts an instance of a problem if there exists at least one transition sequence leading to a special accepting state. Otherwise it rejects.

Definition 2.18. *The complexity class NP of all non-deterministic polynomially solvable problems is defined as the set of all languages which can be decided by a non-deterministic Turing Machine that runs in polynomial time.*

It is currently unknown whether the problems in NP could also be solved by a deterministic Turing Machine, i.e. whether P = NP. Consent is that this is most likely not the case. Special problems which are at least as hard as all problems in NP play an important role in this question. If an efficient, polynomial algorithm is found for only one of these problems, all problems in NP are solvable in polynomial time.

The most important tool to prove that a problem is at least as hard as another one are reductions. They are used to define the classes of *NP-hard* and *NP-complete* problems.

Definition 2.19. *A problem P_1 is polynomial-time reducible to a problem P_2 if there exists a function f, computable by a Turing Machine in polynomial time, such that for all problem instances x: $x \in P_1 \Leftrightarrow f(x) \in P_2$.*

A problem P' is called NP-hard if every problem $P \in NP$ is polynomial-time reducible to it. P' is called NP-complete if additionally $P' \in NP$.

If a polynomial algorithm is found for any NP-hard problem, then P = NP. Therefore, it is of great interest not only to determine the complexity class of a problem, but also to determine whether it is NP-hard.

All of the problems examined in this thesis are optimization rather than decision problems. A formal definition of optimization problems can be found in [Kel14, Definition 1.12]. If we want to analyze the complexity of an optimization problem, a common approach is to use the so-called *threshold language* [Kel14, Definition 1.13]. Instead of optimizing a value over instances of a problem, we decide whether there exists an instance with a value larger or smaller than a given threshold for maximization or minimization problems respectively. The definitions of complexity classes and

reductions can then be adapted for optimization problems using the threshold language.

Commonly, optimization problems are implicitly designated as 'polynomially solvable' if any computational model, e.g. a Turing Machine or a modern computer, can give an optimal solution in polynomial time. In that case it is clearly possible to decide the corresponding threshold language in polynomial time, too. Hence, for the purposes of this thesis, we will consider this definition to be equivalent.

3 Scaled robust flows

The goal of this chapter is to first describe the Maximum Robust Flow problem and then introduce a variation called Maximum Scaled Robust Flow. Hardness results are presented for some cases of the original problem as well as the scaled version. Various scaling factors are examined and characteristics of solutions are determined for the different factors. Finally, we will take a look at possibilities to scale the problem with non-linear functions.

3.1 The original robust flow problem

Maximum Robust Flow deals with the problem of a network flow which is subjected to uncertainty in form of failing arcs in the graph. We want to maximize the remaining flow after a number of arcs are removed in the worst-case scenario.

This section gives a formal definition and discusses hardness, approximation and other related results from literature.

3.1.1 Definition

We are given a network $N = (G, s, t, u)$ as defined in chapter 2 as well as an integer $k \in \mathbb{N}$ specifying the number of failing arcs. We can assume that $k \leq |E|$, since obviously no more than $|E|$ arcs can be interdicted. As before, $\mathcal{P}$ denotes the set of simple s-t-paths in G and we define $\mathcal{S} = \{S \subset E : |S| = k\}$ as the set of subsets of E with cardinality k.

For this problem, we only consider path-based s-t-flows. Recall that the network flow value for a path-based flow x is

$$\text{val}(x) = \sum_{P \in \mathcal{P}} x(P).$$

We now consider an adversary who destroys k arcs, such that the amount of interrupted flow is as large as possible. This adversary is usually called *interdictor*, and the *interdictor value* is defined as

$$\text{val}_I(x) = \max_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} x(P).$$

Recall that $\mathcal{P}_S = \{P \in \mathcal{P} : P \cap S \neq \emptyset\}$, hence the interdictor maximizes the sum of flow values over all paths intersecting with the set of failing arcs S.

In the Maximum Robust Flow problem, we now consider the *robust value*, which is the difference between network value and interdictor value:

$$\mathrm{val}_r(x) = \sum_{P \in \mathcal{P}} x(P) - \max_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} x(P) \tag{3.1}$$

Definition 3.1. *The problem of finding a feasible flow in N with maximum robust value is called* Maximum Robust Flow *and is denoted by ROB.*

Because of the dual structure of the problem, it is sometimes viewed as a two-player game, the first player being refered to as the *flow player* declaring the flow and the second player being the interdictor. Both players try to maximize their payoff. The next section deals with various results concerning the problem.

3.1.2 Hardness and approximation results

Maximum Robust Flow was first investigated by Aneja, Chandrasekaran and Nair [ACN01] in 2001. It was discovered that for $k = 1$ the problem is solvable in polynomial time by using a parametric linear program. The same holds for a problem variation where the maximum flows must be integral. We will see an alternative approach to proving polynomiality for $k = 1$ in section 3.3 of this chapter, which uses a shortest path computation to solve the separation version of the problem.

Very recently, Disser and Matuschke [DM17] established NP-hardness in the case when k is an unbounded parameter, even when the number of paths in the network is polynomially small. They also show that finding an integral maximum robust flow is NP-hard for $k \geq 2$.

The hardness of the problem remains unknown for $k \geq 2$, but several approximation algorithms have been developed. The approach by Bertsimas, Nasrabadi and Orlin [BNO16] gives a tight approximation factor of $(1 + \lfloor k/2 \rfloor \cdot \lceil k/2 \rceil)/(k + 1)$.

3.2 Definition of scaled robust flows

In the original robust flow problem, all flow going through interdicted arcs is destroyed. For several applications this is reasonable and works as intended. However, in some cases it might happen that the interdictor cannot or does not want to interrupt all flow through a destroyed arc.

For example, we could think of the interdictor as a thief stealing a fixed amount of flow on each arc subject to some maximum carrying capacity. Other interdictors may only want to cripple the flow instead of destroying it outright, interrupting a given factor of the flow through an arc. Further possibilities include greater interdiction if the flow on a given arc is larger, or analogously greater interdiction if the flow is smaller.

To capture these various cases, we consider the following variation of ROB. We introduce a *scaling function* $f : \mathbb{R} \to \mathbb{R}$, and define the *f-scaled robust value* of a

given flow x:

$$\mathrm{val}_f(x) = \sum_{P \in \mathcal{P}} x(P) - \max_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} f(x(P)) \tag{3.2}$$

As can be seen above, f scales directly on the path values. This is, of course, not necessarily the only possibility; one could also apply the scaling on the sum of path values or directly on the interdictor value. For linear functions $f(x) = c \cdot x,\ c \in \mathbb{R}$, linearity enables us to pull out the factor c to the front of the maximization. For the linear case we define the *scaled robust value* as

$$\begin{aligned} \mathrm{val}_c(x) &= \sum_{P \in \mathcal{P}} x(P) - c \cdot \max_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} x(P) \\ &= \mathrm{val}(x) - c \cdot \mathrm{val}_I(x). \end{aligned} \tag{3.3}$$

Linearly scaled robust flows are examined thoroughly in the following sections.

If f is convex, the scaling results in a higher penalty for larger destroyed flow. Equivalently, if f is concave, it gives less penalty for larger destroyed flow. Affine linear functions differ from the linear case only slightly. The added scalar increases or decreases the value depending on the number of paths being interdicted. For an analysis of these other scaling functions, please refer to section 3.7.

If we want to make the penalty dependent on the flow through arcs instead of paths or give specific penalties to certain arcs in the graph, the function f does not suffice. For this task we need to redefine the way the interdictor value is calculated and then scale with a new function. This approach can also be found in section 3.7.

We now want to define scaled robust flows. We are given N, k, $\mathcal{P}$, $\mathcal{S}$ as before when defining ordinary robust flows. Additionally, we are given a scaling factor $c \in \mathbb{R}$. Again, we only consider path-based s-t-flows. Using the scaled robust value from (3.3), the problem is defined as follows:

Definition 3.2. *The problem of finding a feasible flow in N with maximum scaled robust value is called* Maximum Scaled Robust Flow *and is denoted by ROB_c.*

It is obvious that ROB_1 corresponds to the original problem ROB. In the same way, ROB_0 is the same as NF. Especially interesting are therefore all values $0 \leq c \leq 1$. We take a close look at how solutions transition from network flows to robust flows when c gets increased from 0 to 1.

$c > 1$ corresponds to high penalties for interdicted arcs. We will see that if c gets large enough, the best possible course for the flow player is to send no flow at all. On the other hand, $c < 0$ implies that the interdictor actually supports the flow player in the sense that arcs with highest flow count more towards the objective value.

In one of the most important results of this chapter, we will argue that for small absolute values of c, any optimal solution to ROB_c is an optimal solution to NF. Surprising and perhaps a little counter-intuitive is the fact that this result does not hold for arbitrarily large negative values of c. A counter-example for this can be found in section 3.6.

Below, we introduce some general results for scaled robust flows which will be important tools later on.

Lemma 3.3. *Let x and y be path-based flows in a network N, s.t. the flow $x+y$ is feasible. Then for all $c \in \mathbb{R}_0^+$ it holds that $val_c(x+y) \geq val_c(x) + val_c(y)$.*

Proof. Let $S_{x+y}, S_x, S_y \in \mathcal{S}$ be the optimal set of arcs for the interdiction of flows $x+y, x, y$ respectively. Then

$$\begin{aligned}
\mathrm{val}_I(x+y) &= \sum_{P \in \mathcal{P}_{S_{x+y}}} (x+y)(P) \\
&= \sum_{P \in \mathcal{P}_{S_{x+y}}} x(P) + y(P) \\
&= \sum_{P \in \mathcal{P}_{S_{x+y}}} x(P) + \sum_{P \in \mathcal{P}_{S_{x+y}}} y(P) \\
&\leq \sum_{P \in \mathcal{P}_{S_x}} x(P) + \sum_{P \in \mathcal{P}_{S_y}} y(P) \\
&= \mathrm{val}_I(x) + \mathrm{val}_I(y).
\end{aligned}$$

Using this, we conclude

$$\begin{aligned}
\mathrm{val}_c(x+y) &= \mathrm{val}(x+y) - c \cdot \mathrm{val}_I(x+y) \\
&= \mathrm{val}(x) + \mathrm{val}(y) - c \cdot \mathrm{val}_I(x+y) \\
&\geq \mathrm{val}(x) + \mathrm{val}(y) - c \cdot \mathrm{val}_I(x) - c \cdot \mathrm{val}_I(y) \\
&= \mathrm{val}_c(x) + \mathrm{val}_c(y).
\end{aligned}$$

□

Fact 3.4. *Let $N = (G, s, t, u)$ be a network and let $c_1 < c_2$ be given penalty factors. Then for any flow x on N it holds that $val_{c_1}(x) \geq val_{c_2}(x)$. If x is non-zero, it holds that $val_{c_1}(x) > val_{c_2}(x)$.*

Lemma 3.5. *Let $N = (G, s, t, u)$ be a network and let $c_1 \leq c_2$ be given penalty factors. Then $opt_{ROB_{c_1}}(N) \geq opt_{ROB_{c_2}}(N)$.*

Proof. Assume x_1 and x_2 are optimal flows for ROB_{c_1} and ROB_{c_2} respectively. By Fact 3.4 and the optimality of x_1, we know that $\mathrm{val}_{c_2}(x_2) \leq \mathrm{val}_{c_1}(x_2) \leq \mathrm{val}_{c_1}(x_1)$. □

The rest of this chapter is structured in the following way: In section 3.3, we take a look at the primal and dual LP formulation of the problem and try to derive hardness results. In section 3.4 several different scaling factors are examined as well as how the solutions depend on these factors. The subsequent sections introduce important examples which prove tightness for several results. Finally, we consider non-linear and arc-based scaling functions in the last section of this chapter.

3.3 Hardness results

The original ROB is solvable in polynomial time if only a single arc is being interdicted. It is NP-hard if the number of arcs is unbounded. In this section, we want to adapt these results for the scaled version of the problem. We assume $c \in \mathbb{R}_0^+$ throughout the entire section.

3.3.1 LP formulation

To be able to talk about the hardness of the problem, let us first take a look at the primal LP formulation. We construct the program by extending the LP for NF. There, we maximized the value of a given flow under the constraint that all arc capacities are fulfilled. Additionally, all flow had to be non-negative. This yields:

$$\begin{aligned} \max_{x} \quad & \sum_{P \in \mathcal{P}} x(P) \\ \text{s.t.} \quad & \sum_{P \in \mathcal{P}_e} x(P) \leq u(e) \qquad \forall e \in E \\ & x \geq 0 \end{aligned}$$

To transform this into an LP for scaled robust flow, let us introduce a variable $\lambda \in \mathbb{R}$ measuring the amount of interdicted flow. In order to do that, we would like to have

$$\lambda = \max_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} x(P). \tag{3.4}$$

To achieve this, we introduce the constraints

$$\lambda \geq \sum_{P \in \mathcal{P}_S} x(P) \quad \forall S \in \mathcal{S},$$

and substract $c \cdot \lambda$ from the objective function. Since the LP is maximizing, it will choose the smallest λ fulfilling the inequalities, which is exactly the desired value (3.4).

The finalized LP looks like this:

$$\begin{aligned} \max_{x,\lambda} \quad & \sum_{P \in \mathcal{P}} x(P) - c \cdot \lambda \\ \text{s.t.} \quad & \sum_{P \in \mathcal{P}_e} x(P) \leq u(e) \qquad \forall e \in E \\ & \sum_{P \in \mathcal{P}_S} x(P) \leq \lambda \qquad \forall S \in \mathcal{S} \\ & x \geq 0 \end{aligned} \tag{3.5}$$

This program can easily be transformed into its corresponding dual. For that purpose, let $y(e),\ \forall e \in E$ and $z(S),\ \forall S \in \mathcal{S}$ be the dual variables corresponding the first and second set of constraints in the primal respectively.

The second set of constraints in (3.5) can be rewritten as

$$\sum_{P \in \mathcal{P}_S} x(P) - \lambda \leq 0 \quad \forall S \in \mathcal{S}.$$

Thus, the right-hand side of these constraints is 0, and the dual objective function is simply

$$\sum_{e \in E} u(e)y(e).$$

The first set of constraints in the dual corresponds to the variables $x(P)$ in the primal. Their factor in the primal objective is 1 and they appear in both constraints of (3.5). This results in the following constraint for the dual:

$$\sum_{e \in P} y(e) + \sum_{S \in \mathcal{S}: P \cap S \neq \emptyset} z(S) \geq 1 \quad \forall P \in \mathcal{P}$$

λ, however, only appears in the constraints pertaining to the variables $z(e)$. Additionally, λ is not constrained to non-negative numbers in the primal. This produces the constraints

$$\sum_{S \in \mathcal{S}} z(S) = c.$$

Both dual variables appear in the non-negativity constraints of the dual, since both corresponding constraints in the primal were inequalities. Altogether, the dual LP of ROB_c is

$$\begin{aligned} \min_{y,z} \quad & \sum_{e \in E} u(e)y(e) \\ \text{s.t.} \quad & \sum_{e \in P} y(e) + \sum_{S \in \mathcal{S}: P \cap S \neq \emptyset} z(S) \geq 1 \qquad \forall P \in \mathcal{P} \\ & \sum_{S \in \mathcal{S}} z(S) = c \\ & y, z \geq 0. \end{aligned}$$

3.3.2 Computing the hardness

Let us first concentrate on the case $k = 1$. Then, the set $\mathcal{S}$ is simply the set of all arcs E. Thus, we can rewrite the variables $z(S)$ as $z(e)$ for all $e \in E$. Consequently, $S \in \mathcal{S}$ is rewritten as $e \in E$ and $S \in \mathcal{S} : P \cap S \neq \emptyset$ as $e \in P$. This yields the dual LP in the case $k = 1$:

$$\begin{aligned} \min_{y,z} \quad & \sum_{e \in E} u(e)y(e) \\ \text{s.t.} \quad & \sum_{e \in P} y(e) + z(e) \geq 1 \qquad \forall P \in \mathcal{P} \\ & \sum_{e \in E} z(e) = c \\ & y, z \geq 0 \end{aligned}$$

Lemma 3.6. *The dual LP of* Maximum Scaled Robust Flow *can be solved in polynomial time if* $k = 1$ *and* $c \in \mathbb{Q}$.

Proof. Consider the polyhedron Q of all feasible solutions of this dual:

$$Q := \left\{ (y, z) \in \mathbb{R}^{E \times E}_{\geq 0} : \quad \sum_{e \in P} y(e) + z(e) \geq 1 \ \forall P \in \mathcal{P}, \quad \sum_{e \in E} z(e) = c, \right\} \tag{3.6}$$

If we can find a polynomial oracle to solve the separation problem 2.15 for Q, then by Theorem 2.16 any corresponding linear program is solvable in polynomial time.

The following is an oracle for the separation of Q:

Input: $(y, z) \in \mathbb{R}^{E \times E}_{\geq 0}$, Network N, $c \in \mathbb{R}^+_0$
If not $\sum_{e \in E} z(e) = c$:
  Output $(y, z) \notin Q$
Compute the shortest s-t-path P' in N w.r.t. arc costs $c(e) = y(e) + z(e)$
If not $\sum_{e \in P'} c(e) \geq 1$:
  Output $(y, z) \notin Q$
Output $(y, z) \in Q$

If the first if-clause is true, then a violated inequality is given by $\sum_{e \in E} z(e) \geq c$ or $\sum_{e \in E} z(e) \leq c$. If the second if-clause is true, a violated inequality is given by $\sum_{e \in P'}(y(e) + z(e)) \geq 1$. If (y, z) fulfills the inequality $\sum_{e \in P'} c(e) \geq 1$ for the shortest path P', then, by our choice of the arc costs, all path inequalities are fulfilled as well. In particular, the oracle outputs $(y, z) \in Q$ correctly.

The first if-clause $\sum_{e \in E} z(e) = c$ is only a single equation to be checked, which can be done in polynomial time. Computing a shortest path for given arc-weights is polynomial, using for example Dijkstra's algorithm [Dij59]. The second if-clause is again a single statement to be checked. Hence the above oracle runs in polynomial time.

Q is a rational polyhedron, because $c \in \mathbb{Q}$. Therefore by Theorem 2.16, the dual of ROB_c for $k = 1$ is solvable in polynomial time. □

Note, that the rationality of Q is the only point at which we require $c \in \mathbb{Q}$. It is unlikely that an irrational scaling factor is chosen, but if that is the case, we may for example approximate with a rational number close to c. The resulting error is at most linear w.r.t the deviation from the original scaling factor.

Given a solution to the dual as well as the set of inequalities generated during separation, we can restrict the primal program to these inequalities and then solve it, as indicated in the proof of Theorem 1 in [MMO17]. Hence ROB_c is solvable in polynomial time for $k = 1$. It is interesting to note that, if we set $c = 1$, the proof above works just as fine, giving an alternative proof for the polynomiality of the original ROB problem for $k = 1$.

Unfortunately, proving a similar result for values of k larger than 1 is not as easy. The polyhedron of all feasible solutions of the dual is given by

$$Q := \left\{ (y,z) \in \mathbb{R}^{E \times \mathcal{S}}_{\geq 0} : \quad \sum_{e \in P} y(e) + \sum_{S \in \mathcal{S}: P \cap S \neq \emptyset} z(S) \geq 1 \quad \forall P \in \mathcal{P}, \right. \\ \left. \sum_{S \in \mathcal{S}} z(S) = c \right\} \tag{3.7}$$

The approach via shortest path computations does not work in this case, since for any $k \geq 2$ the first group of constraints is not a sum over the arcs of a single path anymore and the arc costs cannot be defined in a reasonable way.

In 2007, Du and Chandrasekaran [DC07] proven that separation of (3.7) is NP-hard for all $k \geq 2$, even when constricted to the first group of constraints in Q. However, as pointed out by [DM17], this does not necessarily imply NP-hardness for the dual of ROB, since it is only a special case of the optimization problem over Q where all coefficients corresponding to the z-variables are 0 in the objective function. The same holds for ROB_c.

As a consequence, the hardness of ROB and ROB_c remains an open problem for all $k \geq 2$.

If k is unbounded, [DM17] have shown that ROB is NP-hard. If we could solve ROB_c polynomially for unbounded c, then clearly we could also solve ROB by setting $c = 1$. Hence ROB_c is also NP-hard for unbounded parameters k and c.

	ROB	ROB_c (for unbounded $c \in \mathbb{Q}_0^+$)
$k = 1$	solvable in polynomial time	solvable in polynomial time
$k \geq 2$	open	open
k unbounded	NP-hard	NP-hard

Table 1: Overview of the complexity of ROB and ROB_c

3.4 Characterization of solutions

In this section, we want to examine different scaling factors $c \in \mathbb{R}$ and analyze the effect on the corresponding solutions to ROB_c. We start out by giving an important result for small factors $0 < c \ll 1$, then we consider large and negative cases. Some path connectivity results are presented and finally we give an overview over the solution characteristics for all scaling factors.

3.4.1 Solutions for small scaling factors

Obviously, all solutions to ROB_c with $c = 0$ are solutions to the original network flow problem, since ROB_0 is the same as NF. In this section, we want to investigate

whether this result can be extended to small non-zero values for c. This would give us a range of values for which the solutions of ROB_c can be characterized as maximum network flows.

The following theorem is one of the most important ones in this thesis and establishes a bound of $c < \frac{1}{|E|}$ below which all solutions are network flows.

Theorem 3.7. *Let $N = (G, s, t, u)$ be a network and assume that $0 \leq c < \frac{1}{|E|}$. Then any optimal solution $\bar{x} \in \mathbb{R}^{\mathcal{P}}$ to ROB_c is an optimal solution to NF.*

Proof. Let N be defined as above, $\bar{x} \in \mathbb{R}^{\mathcal{P}}$ an optimal solution to ROB_c. Hence, we have $\forall x \in \mathbb{R}^{\mathcal{P}}$:

$$\mathrm{val}_c(x) \leq \mathrm{val}_c(\bar{x})$$

Assume for contradiction that $\bar{x}$ is not optimal w.r.t. the original network flow problem NF.

We can transform $\bar{x}$ into an arc-based flow $\bar{f}$ of the same value by setting $\bar{f}(e) = \bar{x}(e)$ for any arc e. Then by Theorem 2.12, there exists an augmenting path $\bar{P}$ of value $\bar{v} > 0$ in the residual graph $G_{\bar{f}}$. Without loss of generality we assume that $\bar{P}$ is a simple path. We define

$$x^+ := \min\{\bar{v}, \min_{P \in \mathcal{P}, \bar{x}(P) > 0} \bar{x}(P)\}.$$

The scalar x^+ will be the amount of increasement we use when augmenting our path. The definition of x^+ ensures that the increasement is smaller than or equal to any positive amount of flow sent over any path in $\mathcal{P}$, as well as smaller than or equal to $\bar{v}$.

We augment $\bar{f}$ by x^+ units along $\bar{P}$. This yields an arc-based flow $\hat{f}$ with $\mathrm{val}(\hat{f}) = \mathrm{val}(\bar{f}) + x^+$. We now need the following claim, which will be proven later.

Claim 3.8. *There exists a path decomposition $\hat{x}$ of $\hat{f}$, s.t. for all paths $P \in \mathcal{P}$ we have $\hat{x}(P) - \bar{x}(P) \leq x^+$ and the number of paths with $\hat{x}(P) - \bar{x}(P) \geq 0$ is at most $|E|$.*

Using the results of this claim, we can sum up the current situation in our graph as follows: $\mathrm{val}(\hat{x}) = \mathrm{val}(\hat{f}) = \mathrm{val}(\bar{f}) + x^+ = \mathrm{val}(\bar{x}) + x^+$, where $\bar{f}$ and $\hat{f}$ are arc-based flows and $\hat{x}$ and $\bar{x}$ are path decompositions of $\hat{f}$ and $\bar{f}$ respectively.

In the next step, we define $m := \mathrm{val}_I(\hat{x}) - \mathrm{val}_I(\bar{x})$ and consider another claim.

Claim 3.9. *For the choice of $0 \leq c < \frac{1}{|E|}$ it holds that $c \cdot m < x^+$.*

Using this result and the fact that $\mathrm{val}(\hat{x}) = \mathrm{val}(\bar{x}) + x^+$, we can deduce that

$$\begin{aligned}
\mathrm{val}_c(\hat{x}) &= \mathrm{val}(\hat{x}) - c \cdot \mathrm{val}_I(\hat{x}) \\
&= \mathrm{val}(\bar{x}) + x^+ - c \cdot (\mathrm{val}_I(\hat{x}) - \mathrm{val}_I(\bar{x}) + \mathrm{val}_I(\bar{x})) \\
&= \mathrm{val}(\bar{x}) + x^+ - c \cdot (m + \mathrm{val}_I(\bar{x})) \\
&> \mathrm{val}(\bar{x}) + x^+ - x^+ - c \cdot \mathrm{val}_I(\bar{x}) \\
&= \mathrm{val}(\bar{x}) - c \cdot \mathrm{val}_I(\bar{x}) \\
&= \mathrm{val}_c(\bar{x}),
\end{aligned}$$

which is a contradiction to the optimality of $\bar{x}$ w.r.t. ROB_c. Hence the flow $\bar{x}$ is optimal w.r.t. the problem NF. □

Proof of Claim 3.8. We assume we are given flows $\bar{x}$ and $\bar{f}$ as in the proof of Theorem 3.7. We augment $\bar{f}$ by x^+ units along the path $\bar{P}$ to receive the arc-based flow $\hat{f}$. The goal of this proof is to establish a path decomposition of $\hat{f}$ with the given properties of Claim 3.8. The following construction is based on the ideas of [McC96].

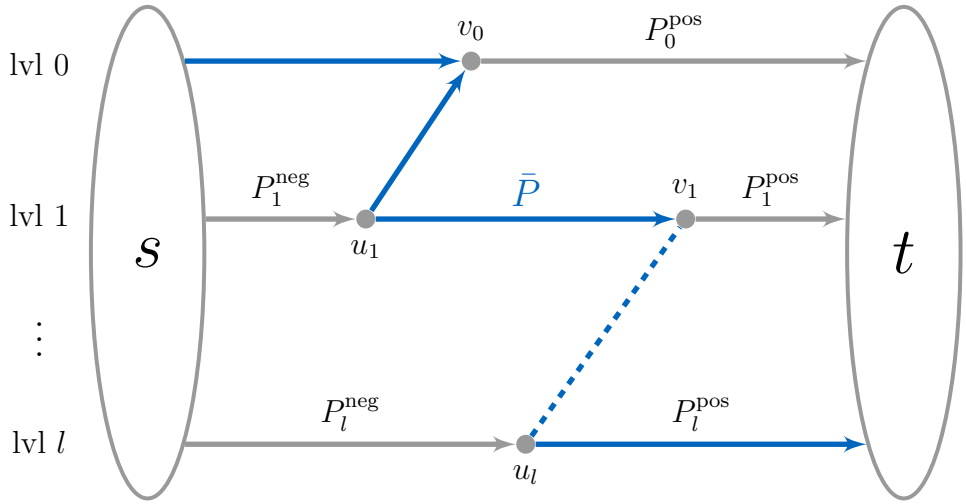

Figure 1: Path decomposition around the augmenting path $\bar{P}$

We start by considering the augmenting path $\bar{P}$. Starting from s, $\bar{P}$ must first use a forward arc. If it were to use a backward arc, there would be a positive amount of flow flowing into s. This means there would be some s-t-path carrying flow that contains a cycle. But this was excluded from our model, where we only considered simple paths.

At some node $v_0 \neq s$, $\bar{P}$ might start using one or several backward arcs. There must exist at least one simple s-t-path containing one or more of these backward arcs of $\bar{P}$ and continuing on over v_0 to t. Since the amount by which we augment is $x^+ \leq \min_{P \in \mathcal{P}, \bar{x}(P)>0} \bar{x}(P)$ and there have to be at least x^+ units of flow along the backward arcs of $\bar{P}$, there must exist at least one such path with flow value at least x^+.

Among the set of paths with this property we now consider one which uses the most consecutive backward arcs of $\bar{P}$ up to v_0. We call this path P_1^{neg}. It carries at least x^+ units of flow. The first node where backward arcs of $\bar{P}$ coincide with P_1^{neg} is called u_1. The path starting in s, then following along $\bar{P}$ up to v_0 and then continuing along P_1^{neg} to t, will be called P_0^{pos}. It may not carry any flow before augmenting.

From the node u_1 the augmenting path $\bar{P}$ uses an unknown amount of forward arcs until at some node v_1 it changes back to backward arcs. Using the same reasoning as above, we obtain a path P_2^{neg} with flow value at least x^+ which uses at least one backward arc of $\bar{P}$ until reaching v_1. Again, we define the path starting in s, following

P_1^{neg} to u_1, then running along $\bar{P}$ to v_1 and finally going along P_2^{neg} to t as P_1^{pos}. Note that it might happen that $u_1 = v_1$ and $\bar{P}$ has no forward arcs between the two nodes. As before, we define the first node where backward arcs of $\bar{P}$ coincide with P_2^{neg} as u_2.

We continue this construction until at some point a forward arc of $\bar{P}$ reaches t. Note that it cannot be a backward arc reaching t, because this would again result in a cyclic path carrying flow. We describe all paths P_i^{pos} and P_i^{neg} as well as nodes u_i and v_i as belonging to *level* i in this construction. We also let $l \in \mathbb{N}$ be the number of levels. Consult Figure 1 for an overview.

Note that all paths P_i^{pos} and P_j^{neg} are simple paths by construction. Also note that u_0, v_l and P_0^{neg} do not exist. Furthermore, all nodes u_i and v_i are part of $\bar{P}$ and do not coincide with either s or t, since this would immediately create a cycle. Conversely, the sections of $\bar{P}$ between the nodes u_i and v_i might be empty. The various sections of P_i^{pos} or P_i^{neg} not lying on $\bar{P}$ might be overlapping arbitrarily, which is not captured by the simplified picture of Figure 1.

With this deconstruction of the paths around the augmenting path $\bar{P}$, we may construct a path decomposition $\hat{x}$ of the augmented flow $\hat{f}$. To do this, we start from the flow $\bar{x}$ and update as follows:

- For all $i \in \{0, \dots, l\}$ increase the flow on P_i^{pos} by x^+
- For all $j \in \{1, \dots, l\}$ decrease the flow on P_i^{neg} by x^+
- All other flows remain unchanged

We have to argue why $\hat{x}$ is a valid flow decomposition of $\hat{f}$. Recall that $\hat{f}$ arises from $\bar{f}$ by augmenting along $\bar{P}$ with a value of x^+. For all arcs $e \notin \bar{P}$ it holds that $\hat{f}(e) = \bar{f}(e)$ and conversely for all arcs $e \in \bar{P}$ it holds that $\hat{f}(e) = \bar{f}(e) + x^+$.

We observe that by our construction for any arc e_1 on some P_i^{pos} but not on $\bar{P}$, there is a $j \in \{i, i+1\}$, s.t. e_1 also lies on P_j^{neg}. Since we increase P_i^{pos} by x^+ and simultaneously decrease P_j^{neg} by x^+, the total amount of flow on e_1 is unchanged. No matter how many different P_i^{pos} run through e_1, there is always exactly one single P_j^{neg} corresponding to each such path. Hence the flow through e_1 w.r.t. $\hat{x}$ is the same as with respect to $\bar{x}$, meaning at e_1 we have the correct amount of flow $\hat{f}(e_1) = \bar{f}(e_1)$.

If we consider an arc e_2 on some P_i^{pos} *and* on $\bar{P}$, then there are two possibilities. Either e_2 does not lie on any P_j^{neg}, then the flow through e_2 is increased by x^+. Since e_2 must be a forward arc of $\bar{P}$, this is correct in terms of the flow augmentation from $\bar{f}$ to $\hat{f}$. On the other hand, if e_2 lies on some P_j^{neg}, then for every such path there must be some $i_j \neq i$, s.t. e_2 lies on $P_{i_j}^{\text{pos}}$, meaning the paths overlap at e_2. All pairs of flows on paths $P_{i_j}^{\text{pos}}$ and P_j^{neg} cancel themselves out as above and the total amount of flow is still increased by x^+ because of P_i^{pos}.

For arcs e_3 on some P_j^{neg} and on $\bar{P}$ we again have two possibilites. If we know that e_3 does not lie on any P_i^{pos}, then the flow through e_3 is decreased by x^+, which is correct since e_3 is a backward arc of $\bar{P}$. Else, if e_3 lies on some P_i^{pos}, then analogous to above, there is a path $P_{j_i}^{\text{neg}}$ with $j_i \neq j$ going through e_3 for every such P_i^{pos}.

Again, all pairs P_i^{pos}, $P_{j_i}^{\text{neg}}$ cancel and the total amount of flow on e_3 is decreased by x^+ because of P_j^{neg}.

Finally, any arc laying neither on $\bar{P}$, P_i^{pos} nor on P_j^{neg} remains entirely unchanged in the sense that all paths going through it carry the same amount of flow as before.

Summarizing, the sum of flow going over any arc in the graph is equal to the flow of that arc w.r.t. $\hat{f}$. Since $\hat{f}$ was a feasible flow, the flow $\hat{x}$ is also feasible. All in all $\hat{x}$ is a path decomposition of $\hat{f}$.

By our construction it obviously holds that $\hat{x}(P) - \bar{x}(P) \leq x^+$, since any path P gets increased by at most x^+. We can also easily see that the number of paths getting increased in this way is $l+1$. In every level $i \geq 1$ of our construction it holds that there must be at least one (backward) arc of $\bar{P}$ between v_{i-1} and u_i. Since there also must be an arc between s and v_0, the number of arcs on $\bar{P}$ must be at least $l+1$. Note that for $l = 0$ we can assume that $v_0 = t$. Using this, we can bound the number of paths that are increased by $l+1 \leq |\bar{P}| \leq |E|$, where the second inequality follows from the fact that $\bar{P}$ is a simple path. This means the number of paths with $\hat{x}(P) - \bar{x}(P) \geq 0$ is at most $|E|$, and that concludes the proof of Claim 3.8. □

Proof of Claim 3.9. We assume that $0 \leq c < \frac{1}{|E|}$ and we are given flows $\bar{x}$ and $\hat{x}$ as defined in the proof of Theorem 3.7. Also, we again set $x^+ = \text{val}(\hat{x}) - \text{val}(\bar{x})$ and $m := \text{val}_I(\hat{x}) - \text{val}_I(\bar{x})$.

We define the set $P_\Delta := \{P \in \mathcal{P} : \hat{x}(P) - \bar{x}(P) \geq 0\}$. By Claim 3.8, we have $|P_\Delta| \leq |E|$. Let $\bar{S} \in \mathcal{S}$ be a set of arcs, s.t.

$$\bar{S} \in \operatorname*{argmax}_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} \bar{x}(P).$$

Analogously, let $\hat{S} \in \mathcal{S}$ be a set of arcs, s.t.

$$\hat{S} \in \operatorname*{argmax}_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} \hat{x}(P).$$

Then:

$$\begin{aligned} m &= \text{val}_I(\hat{x}) - \text{val}_I(\bar{x}) \\ &= \sum_{P \in \mathcal{P}_{\hat{S}}} \hat{x}(P) - \sum_{P : \in \mathcal{P}_{\bar{S}}} \bar{x}(P) \\ &\leq \sum_{P \in \mathcal{P}_{\hat{S}}} (\hat{x}(P) - \bar{x}(P)) \\ &\leq \sum_{P \in P_\Delta \cap \mathcal{P}_{\hat{S}}} (\hat{x}(P) - \bar{x}(P)) \\ &\leq \sum_{P \in P_\Delta \cap \mathcal{P}_{\hat{S}}} x^+ \\ &\leq |E| \cdot x^+, \end{aligned} \tag{3.8}$$

where we used the optimality of the interdictor problem $\sum_{P\in\mathcal{P}_S} \bar{x}(P) \leq \sum_{P:\in\mathcal{P}_{\bar{S}}} \bar{x}(P)$ $\forall S \in \mathcal{S}$ in the first inequality and the fact $\hat{x}(P) - \bar{x}(P) \leq x^+ \ \forall P \in \mathcal{P}$ from Claim 3.8 in the third inequality.

The estimate in (3.8) illustrates the main argument of the proof of the theorem. The difference in interdiction m can only be $|E|$ times as large as the augmentation value x^+. Thus, if we choose c smaller than $\frac{1}{|E|}$, we counter large changes in the interdiction value and make sure the new flow has a larger robust value.

With (3.8), we can write

$$c \cdot m \leq c \cdot |E| \cdot x^+ < \frac{1}{|E|} \cdot |E| \cdot x^+ = x^+,$$

which proves the claim. □

A counterexample to the theorem for values of c larger than $\frac{1}{k}$ can be found in section 3.6.

3.4.2 Large and negative scaling factors

This section deals with very large scaling factors, such that the flow player is forced to send no flow at all. We also take a look at negative scaling values where the interdictor actually supports the flow player.

This first result establishes that for $c > |E|$ all solutions are equal to the zero-flow.

Lemma 3.10. *Let $N = (G, s, t, u)$ be a network and $c \geq |E|$. Then $\bar{x} \equiv 0$ is an optimal ROB_c-flow. If $c > |E|$, $\bar{x}$ is the unique solution.*

Proof. Let $c \geq |E|$ and x an arbitrary flow on G. Further, let $e \in E$ be any arc. If e is being interdicted, then clearly $x(e) \leq \mathrm{val}_I(x)$. If, on the other hand, e is not interdicted, assume $x(e) > \mathrm{val}_I(x)$. Then any set $S \in \mathcal{S}$ with $e \in S$ gives an interdiction value of at least $x(e)$, a contradiction. Hence for all $e \in E : x(e) \leq \mathrm{val}_I(x)$.

This enables us to estimate the flow value of x against its interdictor value. In particular: $\mathrm{val}(x) \leq \sum_{e\in E} x(e) \leq |E| \cdot \mathrm{val}_I(x)$. Then:

$$\begin{aligned}
\mathrm{val}_c(x) &= \mathrm{val}(x) - c \cdot \mathrm{val}_I(x) \\
&\leq |E| \cdot \mathrm{val}_I(x) - c \cdot \mathrm{val}_I(x) \\
&= (|E| - c) \cdot \mathrm{val}_I(x) \\
&\leq 0
\end{aligned}$$

Since x was arbitrary, all flows have a scaled robust value of at most 0. Hence $\bar{x}$ is optimal.

In case of $c > |E|$, the last inequality above is strict if and only if x has non-zero interdiction value. The only flow with interdiction value equal to zero is $\bar{x}$, hence it is the unique optimum. □

The following lemma gives a result very similar to Theorem 3.7 for negative values of c.

Lemma 3.11. *$N = (G, s, t, u)$ a network, $-\frac{1}{|E|} < c \leq 0$. Then every solution to ROB_c is a solution to NF.*

Proof. Let $\bar{x}$ be an optimal ROB_c-flow and assume $\bar{x}$ is not optimal w.r.t. NF. Strictly following the proof of Theorem 3.7, we construct the flows $\bar{f}$ and $\hat{f}$ as well as the corresponding augmenting path $\bar{P}$. x^+ is also defined in the same way.

Our goal is to proof $c \cdot m < x^+$ as in Claim 3.9. We use the following adapted claims:

Claim 3.12. *There exists a path decomposition $\hat{x}$ of $\hat{f}$, s.t. $\hat{x}(P) - \bar{x}(P) \geq -x^+$ $\forall P \in \mathcal{P}$ and the number of paths with $\hat{x}(P) - \bar{x}(P) \leq 0$ is at most $|E|$.*

Again, we set $m := \text{val}_I(\hat{x}) - \text{val}_I(\bar{x})$.

Claim 3.13. *For the choice of $-\frac{1}{|E|} < c \leq 0$ it holds that $c \cdot m < x^+$.*

Now we can argue further along the lines of the proof of Theorem 3.7. The exact same chain of inequalities can be used to deduce $\text{val}_c(\hat{x}) > \text{val}_c(\bar{x})$, which gives the contradiction. Thus $\bar{x}$ is optimal w.r.t. NF. □

Proof of Claim 3.12. We use the same construction as in the proof of Claim 3.8. Hence it immediately follows that a path decomposition exists. Additionally, $\hat{x}$ from Claim 3.8 fulfills that any path only gets decreased by at most x^+ and their number is $l \leq |\bar{P}| \leq |E|$. □

Proof of Claim 3.13. This proof is along the lines of the proof of Claim 3.9. We define the set $P'_\Delta := \{P \in \mathcal{P} : \hat{x}(P) - \bar{x}(P) \leq 0\}$. By Claim 3.12 we have $|P'_\Delta| \leq |E|$. We define $\hat{S}$ and $\bar{S}$ as before.

This yields:

$$\begin{aligned}
m &= \text{val}_I(\hat{x}) - \text{val}_I(\bar{x}) \\
&= \sum_{P \in \mathcal{P}_{\hat{S}}} \hat{x}(P) - \sum_{P \in \mathcal{P}_{\bar{S}}} \bar{x}(P) \\
&\geq \sum_{P \in \mathcal{P}_{\bar{S}}} (\hat{x}(P) - \bar{x}(P)) \\
&\geq \sum_{P \in P'_\Delta \cap \mathcal{P}_{\bar{S}}} (\hat{x}(P) - \bar{x}(P)) \\
&\geq \sum_{P \in P'_\Delta \cap \mathcal{P}_{\bar{S}}} -x^+ \\
&\geq -|E| \cdot x^+,
\end{aligned}$$

Now, using our choice of $-\frac{1}{|E|} < c \leq 0$, we can write

$$c \cdot m \leq -c \cdot |E| \cdot x^+ < -\left(-\frac{1}{|E|}\right) \cdot |E| \cdot x^+ = x^+.$$

This concludes the proof. □

3.4.3 Path connectivity results

In the previous section we have seen that a value of $c > |E|$ is enough to prevent the flow player from sending any flow at all. It is possible to improve this result to an even smaller bound. Recall the definition of the s-t-path-connectivity from Definition 2.4. It is intuitive that the size of the required scaling factor to prevent positive flow size depends on the s-t-path-connectivity, since less connectivity means easier interdiction of the entire flow. We want to introduce a theorem giving the exact dependency of the solution of ROB_c on the s-t-path-connectivity.

For that, we first need to take care of a special case, namely if $\pi(G) = 0$, i.e. there exists no path between s and t. Remember that a network in which the graph has no s-t-path was called trivial. In such a network, $\bar{x} \equiv 0$ obviously is the only solution to ROB_c for any $c \in \mathbb{R}$.

Theorem 3.14. *Let $N = (G, s, t, u)$ be a non-trivial network. Then*

$$\bar{x} \equiv 0 \text{ is } ROB_c\text{-optimal} \quad \Longleftrightarrow \quad c \geq \frac{\pi(G)}{\min(k, \pi(G))}$$

Proof. We begin with the forward implication. Let $\bar{x} \equiv 0$ be an optimal ROB_c-flow.

First consider the case where $k \geq \pi(G)$. We need to show $c \geq \frac{\pi(G)}{\pi(G)} = 1$. Assume for contradiction that $c < 1$. Since the network is non-trivial, there exists an s-t-path P' in G. Set $x^+ := \min_{e \in E} u(e)$ and define the flow x' by

$$x'(P) = \begin{cases} x^+ & \text{if } P = P' \\ 0 & \text{else.} \end{cases}$$

Clearly, x' is feasible and has robust value

$$\text{val}_c(x') = x^+ - c \cdot x^+ = (1 - c) \cdot x^+ > 0,$$

where we used $x^+ > 0$, $c < 1$ and that we can always interdict the flow on a single path entirely. This constitutes a contradiction to the optimality of $\bar{x}$ and thus $c \geq 1$.

In the second case we have $k < \pi(G)$ and need to show $c \geq \frac{\pi(G)}{k}$. Again, assume for contradiction that $c < \frac{\pi(G)}{k}$. Let $\mathcal{P}_M$ be a maximal set of independent s-t-paths in G in compliance with Definition 2.5. By Lemma 2.6, the size of $\mathcal{P}_M$ is equal to $\pi(G)$. We again set $x^+ = \min_{e \in E} u(e)$ and define the flow x'' as follows:

$$x''(P) = \begin{cases} x^+ & \forall P \in \mathcal{P}_M \\ 0 & \text{else.} \end{cases}$$

x'' is feasible, since all paths in $\mathcal{P}_M$ are independent and x^+ is smaller than any single arc capacity. It holds $\text{val}(x'') = |\mathcal{P}_M| \cdot x^+ = \pi(G) \cdot x^+$ and, since $k < \pi(G)$,

the interdictor value is $\mathrm{val}_I(x'') = k \cdot x^+$. Thus

$$\begin{aligned}\mathrm{val}_c(x'') &= \mathrm{val}(x'') - c \cdot \mathrm{val}_I(x'') \\ &= \pi(G) \cdot x^+ - c \cdot k \cdot x^+ \\ &> \pi(G) \cdot x^+ - \frac{\pi(G)}{k} \cdot k \cdot x^+ \\ &= 0.\end{aligned}$$

Therefore $\bar{x}$ is not optimal, a contradiction yielding $c \geq \frac{\pi(G)}{k}$ as required.

For the reverse implication, assume $c \geq \frac{\pi(G)}{\min(k,\pi(G))}$. We again divide the problem into two cases. First, consider $k \geq \pi(G)$, giving $c \geq 1$. We need to show that $\bar{x} \equiv 0$ is ROB_c-optimal.

By definition of $\pi(G)$ there exists a set F of $\pi(G)$ arcs that, if removed, disconnect s and t, meaning there are no s-t-paths remaining in $G - F$. Since $k \geq \pi(G)$, the interdictor can always disrupt the entire flow by choosing the arcs in F.

Hence we have $\mathrm{val}_I(x) = \mathrm{val}(x)$ for any given flow x. We can use this for the following estimation:

$$\begin{aligned}\mathrm{val}_c(x) &= \mathrm{val}(x) - c \cdot \mathrm{val}(x) \\ &\leq \mathrm{val}(x) - 1 \cdot \mathrm{val}(x) \\ &= 0\end{aligned}$$

This holds for all possible flows x. Hence we can conclude that $\bar{x} \equiv 0$ is optimal with $\mathrm{val}_c(\bar{x}) = 0$.

In the case $k < \pi(G)$, we obtain $c \geq \frac{\pi(G)}{k}$. Again, let $F \subset E$ be a set of $\pi(G)$ arcs that disconnect s and t. Removing F splits the graph G into two components A and B, w.l.o.g. we assume $s \in A$, $t \in B$. By the disconnecting property of F, (A, B) is an s-t-cut in G. Hence, for given $x \in \mathbb{R}^{\mathcal{P}}$, all flow must go through the arcs in F. Therefore $\mathrm{val}(x) = \sum_{P \in \mathcal{P}_F} x(P)$.

We now remove $\pi(G) - k$ arcs from F to receive $F^* \subsetneq F$ with $|F^*| = k$, such that the remaining flow over F^* is maximum. In other words:

$$F^* \in \operatorname*{argmax}_{\substack{F' \subset F, \\ |F'| = k}} \sum_{P \in \mathcal{P}_{F'}} x(P)$$

An interdictor for the network N will always interdict at least the value of paths through F^*. Hence, $\mathrm{val}_I(x) \geq \sum_{P \in \mathcal{P}_{F^*}} x(P)$.

Because of the maximizing property of F^*, removing the $\pi(G) - k$ arcs not in F^* from F can only increase the average over the interdicted paths. Therefore we have

$$\frac{\sum_{P \in \mathcal{P}_{F^*}} x(P)}{|F^*|} \geq \frac{\sum_{P \in \mathcal{P}_F} x(P)}{|F|}.$$

Using this, we can write

$$\sum_{P \in \mathcal{P}_{F^*}} x(P) \geq \frac{|F^*|}{|F|} \sum_{P \in \mathcal{P}_F} x(P) = \frac{k}{\pi(G)} \sum_{P \in \mathcal{P}_F} x(P).$$

We have collected everything we need to prove the robust value of x is at most 0:

$$\begin{aligned}
\mathrm{val}_c(x) &= \mathrm{val}(x) - c \cdot \mathrm{val}_I(x) \\
&\leq \sum_{P \in \mathcal{P}_F} x(P) - c \cdot \sum_{P \in \mathcal{P}_{F^*}} x(P) \\
&\leq \sum_{P \in \mathcal{P}_F} x(P) - c \cdot \frac{k}{\pi(G)} \sum_{P \in \mathcal{P}_F} x(P) \\
&\leq \sum_{P \in \mathcal{P}_F} x(P) - \frac{\pi(G)}{k} \cdot \frac{k}{\pi(G)} \sum_{P \in \mathcal{P}_F} x(P) \\
&= 0
\end{aligned}$$

Therefore, $\bar{x}$ is optimal with robust value $\mathrm{val}_I(\bar{x}) = 0$. □

Theorem 3.14 presents an easy-to-check approach to test for optimality of the zero-flow. It also guarantees that for problems with $c < \frac{\pi(G)}{\min(k,\pi(G))}$, the zero-flow is *not* optimal. From the theorem we can now derive the following strict version:

Corollary 3.15. *Let $N = (G, s, t, u)$ be a non-trivial network. Then*

$$\bar{x} \equiv 0 \text{ is uniquely } ROB_c\text{-optimal} \quad \Longleftrightarrow \quad c > \frac{\pi(G)}{\min(k, \pi(G))}$$

Proof. Let x be any non-zero flow. By Theorem 3.14, x has at most ROB_c-value 0 if $c = \pi(G)/\min(k, \pi(G))$. If on the other hand $c > \pi(G)/\min(k, \pi(G))$, then by Fact 3.4, x has negative ROB_c-value. Therefore $\bar{x}$ is uniquely ROB_c-optimal.

Let $\bar{x}$ be uniquely ROB_c-optimal. Assume $c \leq \pi(G)/\min(k, \pi(G))$. By Theorem 3.14, equality must hold. But the flow x'' from the proof of the theorem, which sends $x^+ = \min_{e \in E} u(e)$ units along every path in a maximum independent set has scaled robust value

$$\begin{aligned}
\mathrm{val}_c(x'') &= \pi(G) \cdot x^+ - c \cdot \min(k, \pi(G)) \cdot x^+ \\
&= \pi(G) \cdot x^+ - \frac{\pi(G)}{\min(k, \pi(G))} \cdot \min(k, \pi(G)) \cdot x^+ \\
&= 0,
\end{aligned}$$

and thus $\bar{x}$ is not uniquely ROB_c-optimal, a contradiction. □

The following corollary is technically part of the proof of Theorem 3.14, but is important enough to stand on its own.

Corollary 3.16. *$N = (G, s, t, u)$ a network, $k \geq \pi(G)$, $c \geq 1$. Then $\bar{x} \equiv 0$ is an optimal ROB_c-flow. If $c > 1$, $\bar{x}$ is the unique optimal ROB_c-flow.*

Proof. For trivial networks, all of the above obviously holds. So let N be non-trivial. If $c \geq 1$, this follows immediately from Theorem 3.14. If $c > 1$, any non-zero flow gets interdicted completely and since $c > 1$, the penalty is higher than the original flow. Thus $\bar{x}$ is the unique optimum. □

The above corollary intuitively tells us that, if the interdictor can always interdict all flow, $c \geq 1$ is enough to guarantee optimality of $\bar{x} \equiv 0$. This can be applied analogously to cases where $c \leq 1$, guaranteeing optimality of NF-optimal flows.

Corollary 3.17. *$N = (G, s, t, u)$ a network, $k \geq \pi(G)$, $c \leq 1$. Then any optimal NF-solution x^* is an optimal ROB_c-flow. If $c < 1$, only the optimal NF-solutions are optimal ROB_c-flows.*

Proof. As in the proof of Theorem 3.14, we can argue $\text{val}_I(x) = \text{val}(x)$ for all flows x, since the interdictor can always remove the entire flow. Let x^* be an arbitrary optimal NF-flow.

Using $c \leq 1$ and the optimality of x^*, we deduce for all flows x:

$$\begin{aligned}
\text{val}_c(x) &= \text{val}(x) - c \cdot \text{val}(x) \\
&= (1 - c)\ \text{val}(x) \\
&\leq (1 - c)\ \text{val}(x^*) \\
&= \text{val}(x^*) - c \cdot \text{val}(x^*) \\
&= \text{val}_c(x^*)
\end{aligned}$$

This implies optimality of x^* w.r.t. ROB_c with $\text{val}_c(x^*) = (1 - c)\ \text{val}(x^*)$.

If $c < 1$, we know that $(1 - c)$ is strictly larger than 0 and we can improve the above estimation for any flow x with $\text{val}(x) < \text{val}(x^*)$:

$$\begin{aligned}
\text{val}_c(x) &= (1 - c)\ \text{val}(x) \\
&< (1 - c)\ \text{val}(x^*) \\
&= \text{val}_c(x^*)
\end{aligned}$$

Hence no non-NF-optimal flow can be ROB_c-optimal. This concludes the proof. □

With Corollaries 3.16 and 3.17, we can characterize all solutions to ROB_c if $k \geq \pi(G)$: If $c > 1$ then the only solution is the zero-flow. If $c < 1$ then all solutions are maximum network flows. Finally, if $c = 1$ then all flows have robust value 0 and are optimal.

3.4.4 Further characterization results

The findings in this section are special cases of Theorem 3.7. We present one result for the case $k = 1$, which gives a better bound than the theorem, as well as a version for *maximal* flows.

Lemma 3.18. *Let $N = (G, s, t, u)$ be a network with interdictor number $k = 1$ and assume that $0 \leq c < 1$. Then any optimal solution $\bar{x}$ to ROB_c is an optimal solution to NF.*

Proof. Let N be defined as above, $\bar{x}$ an optimal solution to ROB_c. We have $\forall x \in \mathbb{R}^{\mathcal{P}}$:

$$\text{val}_c(x) \leq \text{val}_c(\bar{x})$$

Assume for contradiction that $\bar{x}$ is not optimal w.r.t. the original flow-problem NF. The following line of arguments is equivalent to that of the proof of Theorem 3.14. We transform $\bar{x}$ into an arc-based flow $\bar{f}$ of the same value by setting $\bar{f}(e) = \bar{x}(e)$ for any arc e. Then there exists an augmenting path $\bar{P}$ of value $x^+ > 0$ in the residual graph $G_{\bar{f}}$. Without loss of generality we assume that $\bar{P}$ is a simple path.

We augment $\bar{f}$ by x^+ units along $\bar{P}$. This yields an arc-based flow $\hat{f}$ with $\text{val}(\hat{f}) = \text{val}(\bar{f}) + x^+$.

By Proposition 2.7, there is a path-based flow $\hat{x}$ with $\text{val}(\hat{x}) = \text{val}(\hat{f})$ and $\hat{f}(e) \geq \hat{x}(e)$ for all $e \in E$.

Since we consider $k = 1$, any $S \in \mathcal{S}$ only has one element. Let $\bar{e}$ be the only arc in the optimal set $\bar{S}$ chosen by the interdictor for flow $\bar{x}$. Hence,

$$\bar{x}(e) \leq \bar{x}(\bar{e}) = \text{val}_I(\bar{x}) \quad \forall e \in E.$$

Similarly, let $\hat{e}$ be the only arc in the optimal set $\hat{S}$ chosen by the interdictor for flow $\hat{x}$. Hence,

$$\hat{x}(e) \leq \hat{x}(\hat{e}) = \text{val}_I(\hat{x}) \quad \forall e \in E.$$

We now distinguish between the following cases:

Case 1. $\hat{e} \notin \bar{P}$. This implies $\hat{f}(\hat{e}) = \bar{f}(\hat{e})$. Then we can estimate the interdiction value as follows:

$$\hat{x}(\hat{e}) \leq \hat{f}(\hat{e}) = \bar{f}(\hat{e}) = \bar{x}(\hat{e}) \leq \bar{x}(\hat{e}) + x^+,$$

where we used the fact $\hat{x}(e) \leq \hat{f}(e)$ for all $e \in E$ from Proposition 2.7 for the first inequality.

Case 2a. $\hat{e}$ is a forward arc of $\bar{P}$. Hence $\hat{f}(\hat{e}) = \bar{f}(\hat{e}) + x^+$. We can write

$$\hat{x}(\hat{e}) \leq \hat{f}(\hat{e}) = \bar{f}(\hat{e}) + x^+ = \bar{x}(\hat{e}) + x^+.$$

Case 2b. $\hat{e}$ is a backward arc of $\bar{P}$. Then we have $\hat{f}(\hat{e}) = \bar{f}(\hat{e}) - x^+$. Thus

$$\hat{x}(\hat{e}) \leq \hat{f}(\hat{e}) = \bar{f}(\hat{e}) - x^+ \leq \bar{f}(\hat{e}) + x^+ = \bar{x}(\hat{e}) + x^+.$$

In all cases we obtain the inequality $\hat{x}(\hat{e}) \leq \bar{x}(\hat{e}) + x^+$. We can now insert this inequality into the robust value of $\hat{x}$:

$$\begin{aligned}
\text{val}_c(\hat{x}) &= \text{val}(\hat{x}) - c \cdot \text{val}_I(\hat{x}) \\
&= \text{val}(\bar{x}) + x^+ - c \cdot \hat{x}(\hat{e}) \\
&\geq \text{val}(\bar{x}) + x^+ - c \cdot (\bar{x}(\hat{e}) + x^+) \\
&= \text{val}(\bar{x}) - c \cdot \bar{x}(\hat{e}) + (1 - c) \cdot x^+ \\
&> \text{val}(\bar{x}) - c \cdot \bar{x}(\hat{e}) \\
&\geq \text{val}(\bar{x}) - c \cdot \bar{x}(\bar{e}) \\
&= \text{val}_c(\bar{x})
\end{aligned}$$

We also used $c < 1$ and the optimality of $\bar{e}$ with respect to the interdiction of flow $\bar{x}$. With the above argumentation, we get a contradiction to the optimality of $\bar{x}$ w.r.t. ROB_c. Hence $\bar{x}$ is optimal w.r.t. NF. □

For the second special case, we first need to introduce the notion of maximal flows. To define maximal flows, we need to refine the definition of augmenting paths from section 2.3.

Definition 3.19. *An augmenting path in the residual graph of a network N is called plain if it uses no backward arcs.*

An arc-based flow $\bar{f}$ is called maximal if there exists no plain augmenting path in $G_{\bar{f}}$. A path-based flow is called maximal if the corresponding arc-based flow is maximal.

Lemma 3.20. *Let $N = (G, s, t, u)$ a network and assume $0 \leq c < 1$. Then any optimal solution $\bar{x}$ to ROB_c is a maximal flow.*

Proof. Let N be defined as above, $\bar{x}$ an optimal solution to ROB_c. Since $\bar{x}$ is optimal, we have $\forall x \in \mathbb{R}^{\mathcal{P}}$:

$$\text{val}_c(x) \leq \text{val}_c(\bar{x}).$$

Once again, we transform $\bar{x}$ into an arc-based flow $\bar{f}$ by setting $\bar{f}(e) = x(e)$ as before. By definition, the flow $\bar{x}$ is maximal if there exists no plain augmenting path in $G_{\bar{f}}$.

Assume for contradiction that $\bar{x}$ is not maximal. Thus there exists an augmenting path $\bar{P}$, which contains no backward arcs. Let x^+ denote the value of this path. Again, w.l.o.g. $\bar{P}$ is a simple path. We augment $\bar{f}$ by x^+ units along $\bar{P}$. This yields an arc-based flow $\hat{f}$ with $\text{val}(\hat{f}) = \text{val}(\bar{f}) + x^+$.

We can now construct a path decomposition $\hat{x}$ of the flow $\hat{f}$ in the following way: For all paths $P \in \mathcal{P}$ set $\hat{x}(P) = \bar{x}(P)$ and then update $\hat{x}(\bar{P}) = \bar{x}(\bar{P}) + x^+$. This is a valid decomposition, since the augmenting path $\bar{P}$ contained no backward arcs and thus we can simply increase the flow on this single path to get from $\bar{x}$ to $\hat{x}$.

In the next step we take a closer look at $m := \text{val}_I(\hat{x}) - \text{val}_I(\bar{x})$. Let $\bar{S} \in \mathcal{S}$ be a set of arcs, s.t.

$$\bar{S} \in \operatorname*{argmax}_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} \bar{x}(P)$$

and analogously let $\hat{S} \in \mathcal{S}$ be a set of arcs, s.t.

$$\hat{S} \in \operatorname*{argmax}_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} \hat{x}(P).$$

Then:

$$\begin{aligned} m &= \mathrm{val}_I(\hat{x}) - \mathrm{val}_I(\bar{x}) \\ &= \sum_{P \in \mathcal{P}_{\hat{S}}} \hat{x}(P) - \sum_{P \in \mathcal{P}_{\bar{S}}} \bar{x}(P) \\ &\leq \sum_{P \in \mathcal{P}_{\hat{S}}} (\hat{x}(P) - \bar{x}(P)) \\ &\leq \hat{x}(\bar{P}) - \bar{x}(\bar{P}) \\ &= x^+, \end{aligned}$$

where we used the fact that $\hat{x}(P) = \bar{x}(P)$ for all $P \neq \bar{P}$ in the last inequality.

We can now use the above estimation to argue the following:

$$\begin{aligned} \mathrm{val}_c(\hat{x}) &= \mathrm{val}(\hat{x}) - c \cdot \mathrm{val}_I(\hat{x}) \\ &= \mathrm{val}(\bar{x}) + x^+ - c \cdot (\mathrm{val}_I(\hat{x}) - \mathrm{val}_I(\bar{x}) + \mathrm{val}_I(\bar{x})) \\ &= \mathrm{val}(\bar{x}) + x^+ - c \cdot (m + \mathrm{val}_I(\bar{x})) \\ &\geq \mathrm{val}(\bar{x}) - c \cdot \mathrm{val}_I(\bar{x}) + (1 - c) \cdot x^+ \\ &> \mathrm{val}(\bar{x}) - c \cdot \mathrm{val}_I(\bar{x}) \\ &= \mathrm{val}_c(\bar{x}) \end{aligned}$$

We used $c < 1$ for the last inequality. This is a contradiction to the NF-optimality of $\bar{x}$. Hence $\bar{x}$ is a maximal flow. □

3.4.5 Summary of results

We now want to give an overview of the different solutions found for ROB_c depending on the parameters k and c.

First, let us consider an arbitrary value $k \in \mathbb{N}$. Table 2 lists results for this case. A question mark in the table signals values of c for which a characterizing solution could not be found.

Small values of $|c| < \frac{1}{|E|}$ can be characterized as optimal NF-flows, which has been proven in Theorem 3.7 and Lemma 3.11. Theorem 3.14 and in particular Corollary 3.15 state that for all values of $c > \pi(G)/\min(k, \pi(G))$, the zero-flow is the unique solution.

Table 3 lists results for the case $k \geq \pi(G)$. With Corollaries 3.16 and 3.17 we can characterize all solutions for $c < 1$ and $c > 1$ respectively. In the case $c = 1$, any given flow on the network is ROB_c-optimal with scaled robust value 0.

In Table 4, the results for the case $k = 1$ can be found. The only difference to the results for unbounded k is that we know all solutions are network flows for values of c up to 1. We have proven this in Lemma 3.18. Compared to $\frac{1}{|E|}$, this result gives a much better bound.

We conclude the section on solution characterizations with two examples highlighting the unknown intervals in the tables. First, let us consider a simple example with

$c \in$	$\left(-\infty, -\frac{1}{\lvert E\rvert}\right]$	$\left(-\frac{1}{\lvert E\rvert}, \frac{1}{\lvert E\rvert}\right)$	$\left[\frac{1}{\lvert E\rvert}, \frac{\pi(G)}{\min(k,\pi(G))}\right]$	$\left(\frac{\pi(G)}{\min(k,\pi(G))}, \infty\right)$
Solution of ROB_c	?	NF-optimal	?	0

Table 2: Characterization of solutions of ROB_c for $k \in \mathbb{N}$ unbounded

$c \in$	$(-\infty, 1)$	$\{1\}$	$(1, \infty)$
Solution of ROB_c	NF-optimal	arbitrary	0

Table 3: Characterization of solutions of ROB_c for $k \geq \pi(G)$

$c \in$	$\left(-\infty, -\frac{1}{\lvert E\rvert}\right]$	$\left(-\frac{1}{\lvert E\rvert}, 1\right)$	$\left[1, \frac{\pi(G)}{\min(k,\pi(G))}\right]$	$\left(\frac{\pi(G)}{\min(k,\pi(G))}, \infty\right)$
Solution of ROB_c	?	NF-optimal	?	0

Table 4: Characterization of solutions of ROB_c for $k = 1$

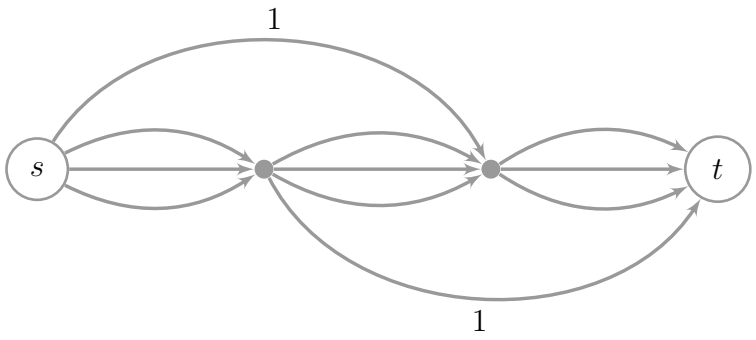

Figure 2: First example highlighting solution characterizations; all unlabeled arcs have capacity $\frac{1}{3}$

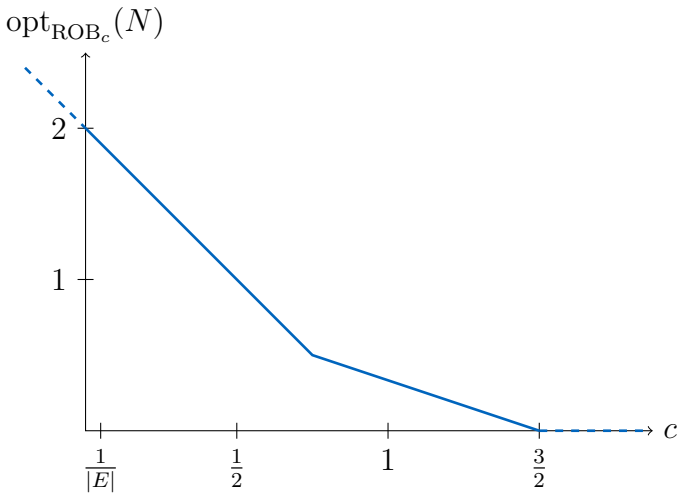

Figure 3: Optimal ROB_c-values as a function of scaling parameter c for the example in Figure 2

$k = 2, \pi(G) = 3$. The network is given in Figure 2. Figure 3 depicts the optimal value of ROB_c as a function of c.

For all values $c \leq \frac{3}{4}$ the following flow x_1 is optimal: send 1 unit of flow along the top arc with capacity 1 and then further to t, and send 1 unit of flow from s to its adjacent node and then along the bottom arc with capacity 1. This flow has scaled robust value $\text{val}_c(x_1) = 2 - c \cdot 2$.

For values $\frac{3}{4} \leq c \leq \frac{3}{2}$, the flow x_2 that sends 1 unit of flow evenly distributed over the three parallel center paths is optimal. It has value $\text{val}_c(x_2) = 1 - c \cdot \frac{2}{3}$.

Finally, for all values $c \geq \pi(G)/\min(k, \pi(G)) = \frac{3}{2}$, the zero-flow is optimal, as predicted by Theorem 3.14.

The reader may feel free to compare the graph in Figure 3 with the bounds from Table 2. It is worth noting that the transition at $c = \frac{3}{4}$ does not happen at one of the bounds, but rather in between them. It is therefore very much possible that the optimal solution to ROB_c stays a network flow for values much higher than $\frac{1}{|E|}$.

The second set of examples shows that there may exist infinitely many different other solutions between the NF-optimal and the zero-flow solution. It can be constructed for all values of $\pi(G)$, but we only show the case with parameters $\pi(G) = 3$ and $k = 1$. Figure 4 shows the network and Figure 5 shows the corresponding coordinate system.

Again, every linear segment in the graph belongs to a separate solution. For example, the optimal flow x_{1-2} for $c \in [1, 2]$ sends 2 units on the top and middle arc, and 1 unit on the bottom arc, giving a ROB_c-value of $\text{val}_c(x_{1-2}) = 5 - c \cdot 2$.

Compare this example with Tables 2 and 4. We can see that extending the result on NF-optimality to $c = 1$ for $k = 1$ makes an actual difference in this example. At $c = \pi(G)/\min(k, \pi(G)) = 3$, we can see once again that the zero-flow is optimal for all larger values in accordance to our theorem.

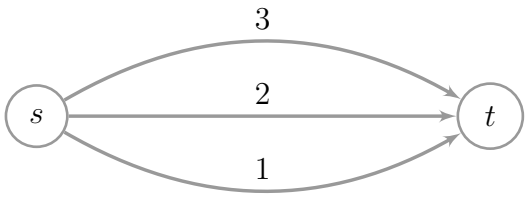

Figure 4: Second example highlighting solution characterizations

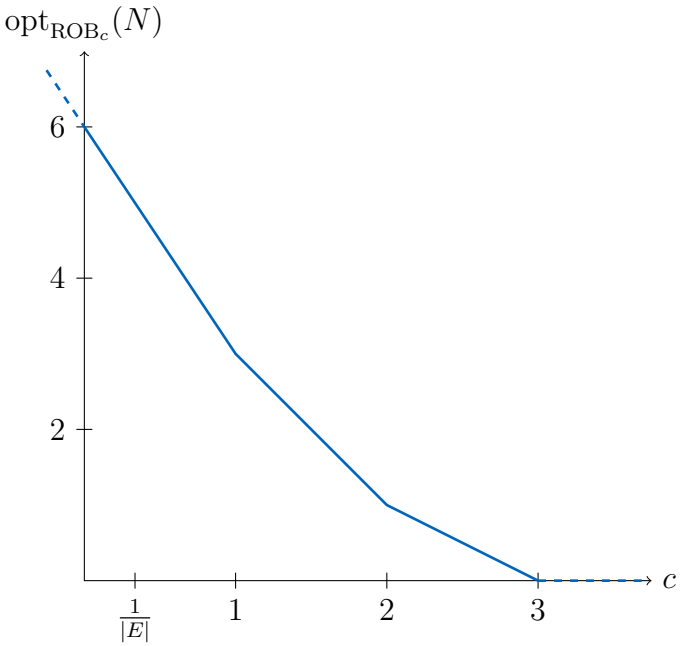

Figure 5: Optimal ROB_c-values as a function of scaling parameter c for the example in Figure 4

As mentioned already, this example can be extended to any value of $\pi(G)$ by simply adding arcs with increasing integral capacities. This yields examples where the number of separate flows between the NF-optimal and the zero-flow solution is unbounded.

3.5 Networks with non-interdictable arcs

In our original model, both for ROB and ROB_c, every arc is generally eligible for interdiction. However, it can make sense in certain applications to be able to block selected arcs from interdiction entirely.

For hardness computations, making certain arcs non-interdictable makes no large difference. The only change is that all arc sets containing non-interdictable arcs are removed from the set $\mathcal{S}$. Of course, if we make enough arcs non-interdictable, the problem may become easier, even equivalent to the Network Flow problem if all arcs are denied interdiction.

This section shows that, if we introduce non-interdictable arcs, the value of ROB_c-flows may only change within an ε-margin compared to certain network constructions. Hence we may use non-interdictable arcs for counterexamples and tightness proofs, c.f. Section 3.6.

We use the following definitions to make clear how networks with non-interdictable arcs are going to be handled.

Definition 3.21. *Let $N = (G, s, t, u)$ be a network with possibly non-interdictable arcs. We define the multi-arc network $N_M = (G_M{=}(V, E_M), s, t, u_M)$ of N for $M \in \mathbb{N}$ in the following way:*

Copy all nodes, arcs and capacities of G. For any non-interdictable arc $e = (u, v) \in E$, delete e and instead add M arcs $e_1, \dots, e_M$ with $e_i = (u, v)$ and capacities $u_M(e_i) = \frac{1}{M}u(e)$.

Clearly, every arc in G_M is interdictable and $\forall e \in E : u(e) = \sum_{i=1}^{M} u_M(e_i)$. In the next step we describe what happens with flows on networks with non-interdictable arcs.

Definition 3.22. *Let N be a network and N_M the corresponding multi-arc network, x a flow on N. The multi-arc flow x_M of x on the network N_M is defined by setting $x_M(e_i) = \frac{1}{M}x(e)$ for all non-interdictable arcs e and their corresponding copies e_i in G_M and $x_M(e) = x(e)$ for all other arcs.*

Obviously, x_M is feasible for N_M and $\text{val}(x_M) = \text{val}(x)$, with values being determined on the appropriate networks.

The following lemma establishes that, for any flow on a network with non-interdictable arcs, there exists a corresponding multi-arc network and multi-arc flow, such that the scaled robust values of the flows are arbitrarily close. In other words, barring an epsilon-error, we may construct examples of robust flows on networks where non-interdictable arcs are allowed.

Lemma 3.23. *Let $N = (G, s, t, u)$ be a network with possibly non-interdictable arcs. Let $\varepsilon > 0$, $c \in \mathbb{R}_0^+$. Then for any flow x in N, there exists $M \in \mathbb{N}$, s.t. for x_M on N_M it holds that $|val_c(x) - val_c(x_M)| < \varepsilon$.*

Proof. Let x be an arc-based flow in N. We choose $M > c \cdot k \cdot \max_{e \in E} u(e) \cdot \frac{1}{\varepsilon}$.

In a first step we want to bound the interdiction value of the multi-arc flow. x_M can only be interdicted as much as x plus at most k times on the new multi-arcs of previously non-interdictable arcs. Every such multi-arc has at most a flow-value of $\max_{e \in E} u(e) \cdot \frac{1}{M}$ by construction. Hence

$$\mathrm{val}_I(x_M) \leq \mathrm{val}_I(x) + k \cdot \max_{e \in E} u(e) \cdot \frac{1}{M}.$$

We will use this estimation to bound the difference between the scaled robust values of x and x_M:

$$\begin{aligned}
\mathrm{val}_c(x) &= \mathrm{val}(x) - c \cdot \mathrm{val}_I(x) \\
&= \mathrm{val}(x_M) - c \cdot \mathrm{val}_I(x) \\
&\leq \mathrm{val}(x_M) - c \cdot \mathrm{val}_I(x_M) + c \cdot k \cdot \max_{e \in E} u(e) \cdot \frac{1}{M} \\
&= \mathrm{val}_c(x_M) + c \cdot k \cdot \max_{e \in E} u(e) \cdot \frac{1}{M} \\
&< \mathrm{val}_c(x_M) + \varepsilon
\end{aligned}$$

Since it is clear that $\mathrm{val}_c(x_M) \leq \mathrm{val}_c(x)$, we can combine it with the above result to receive $|\mathrm{val}_c(x_M) - \mathrm{val}_c(x)| < \varepsilon$. □

3.6 Tightness proofs and important examples

Example 3.24. *Let $k \in \mathbb{N}$ be the interdiction number of ROB_c. We define the network $N = (G, s, t, u)$ with $V = \{s, t, u_1, \ldots, u_k, v_1, \ldots, v_k\}$ where we have arcs from s to u_i, from u_i to v_i and from v_i to t for all $i \in [k]$. Additionally, there are arcs leading from u_i to v_{i-1} for all $2 \leq i \leq k$. All arcs have capacity 1. Only the arcs $e_i := (u_i, v_i)$ are interdictable. Consult Figure 6 for an overview of the network. Non-interdictable arcs are marked with orange colour in the picture.*

Using the above example, we can show an upper bound to possible values of c in Theorem 3.7. There, we used $c < \frac{1}{|E|}$. The following argumentation provides a counterexample for all $c > \frac{1}{k}$. We can assume that $k \leq |E|$, because additional interdiction above $|E|$ yields no benefits. This gives $\frac{1}{|E|} \leq \frac{1}{k}$, but there may be a gap between the two values.

Example 3.24 includes non-interdictable arcs. We first prove its correctness as a counterexample before using the notion of multi-arc networks to construct a version where all arcs are interdictable.

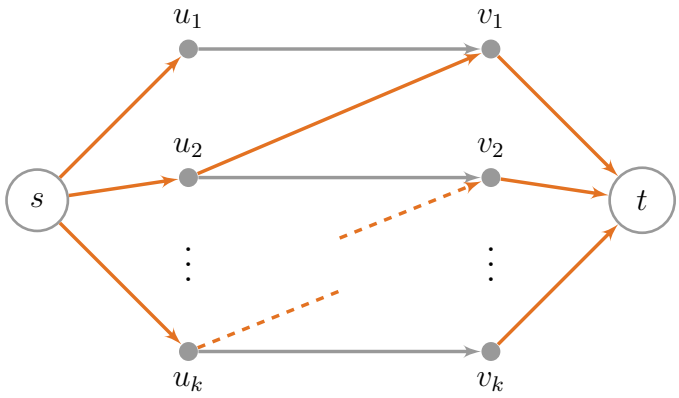

Figure 6: Counterexample to Theorem 3.7 for $c \geq \frac{1}{k}$, non-interdictable arcs are marked with orange colour

Lemma 3.25. *Let $c = \frac{1}{k}$. Then the optimal value of NF in the network N of Example 3.24 is k. The optimal value of ROB$_c$ is $k-1$. Additionally there exists a flow $\bar{x}$ with $val(\bar{x}) = k-1$ and $val_c(\bar{x}) = k-1$.*

Since $\bar{x}$ is optimal w.r.t. ROB$_c$, but not NF, this constitutes a counterexample to Theorem 3.7 for $c = \frac{1}{k}$.

Proof. We define the following paths in G: $P_i := \{(s,u_i),(u_i,v_i),(v_i,t)\}$ for $i \in [k]$ and $\bar{P}_j := \{(s,u_{j+1}),(u_{j+1},v_j),(v_j,t)\}$ for $j \in [k-1]$. These paths form the entire set of viable paths in the network.

Consider first the flow x^* which sends 1 unit of flow along every P_i and 0 units along $\bar{P}_j$. Obviously, this flow has NF-value k. To prove its optimality, consider the cut $(\{s\}, V \setminus \{s\})$, which has also value k. By the max-flow-min-cut theorem, x^* is an optimal flow.

We define the flow $\bar{x}$ in the following way: $\bar{x}$ sends 0 units of flow along the paths P_i and 1 unit along the paths $\bar{P}_j$. Since all edges on the paths $\bar{P}_j$ are non-interdictable, we can deduce that $\text{val}_c(\bar{x}) = \text{val}(\bar{x}) = k-1$.

Lastly, we need to prove that the optimal value of ROB$_c$ is $k-1$. We use the following linear program, which is equivalent to ROB$_c$ on N and has a global maximum of value $k-1$:

$$\begin{aligned} \max_{x(P_i),x(\bar{P}_j)} \quad & \sum_{i=1}^{k} \left(1 - \frac{1}{k}\right) x(P_i) + \sum_{j=1}^{k-1} x(\bar{P}_j) & & \\ \text{s.t.} \quad & x(P_i) + x(\bar{P}_i) \leq 1 & & \forall i \in [k-1] \\ & x(P_{i+1}) + x(\bar{P}_i) \leq 1 & & \forall i \in [k-1] \\ & 0 \leq x(P_i), x(\bar{P}_j) \leq 1 & & \forall i \in [k], j \in [k-1] \end{aligned} \tag{3.9}$$

The objective function sums up the values of all available paths in the network minus any interdiction happening on the paths P_i. The flow on any pair of paths

$P_i, \bar{P}_i$ as well as $P_{i+1}, \bar{P}_i$ may not exceed 1, since these pairs share exactly one arc with capacity 1. This is taken care of by the first two set of constraints. The remaining constraints make sure all flows are within the given capacities.

The above program is clearly linear. This means that in every point a so-called constraint qualification holds [UU12, Theorem 16.18]. Therefore, any point in which the Karush-Kuhn-Tucker conditions are fulfilled is a global maximum [UU12, Theorems 16.14, 16.26].

We define the vectors $x = (x(P_1), \ldots, x(P_k), x(\bar{P}_1), \ldots, x(\bar{P}_{k-1}))^\top \in \mathbb{R}^{2k-1}$ as well as $d = (1-\frac{1}{k}, \ldots, 1-\frac{1}{k}, 1, \ldots, 1)^\top \in \mathbb{R}^{2k-1}$ with consistent dimensions. We transform the maximization problem above to an equivalent minimization problem with the objective function $f(x) = -d^\top x$.

The inequality constraints $g\colon \mathbb{R}^{2k-1} \to \mathbb{R}^{6k-4}$ are given by:

$$g(x) = \left(\begin{array}{c|c|c} \mathbb{I}_{k-1} & \begin{matrix}\vdots\\0\\\vdots\end{matrix} & \mathbb{I}_{k-1} \\ \hline \begin{matrix}\vdots\\0\\\vdots\end{matrix} & \mathbb{I}_{k-1} & \mathbb{I}_{k-1} \\ \hline \multicolumn{3}{c}{-\mathbb{I}_{2k-1}} \\ \hline \multicolumn{3}{c}{\mathbb{I}_{2k-1}} \end{array}\right) \cdot x \quad - \quad \left(\begin{array}{c} \vdots \\ 1 \\ \vdots \\ \hline \vdots \\ 1 \\ \vdots \\ \hline \vdots \\ 0 \\ \vdots \\ \hline \vdots \\ 1 \\ \vdots \end{array}\right)$$

Let $\lambda \in \mathbb{R}^{6k-4}_{\geq 0}$ be the corresponding Lagrange multipliers and $L(x, \lambda) = f(x) + \lambda^\top g(x)$ the Lagrangian function. Consider again $x^* = (1, \ldots, 1, 0, \ldots, 0)$ and choose

$$\lambda^* = (\underbrace{1 - 1/k, 1 - 2/k, \ldots, 1/k}_{k-1}, \quad \underbrace{1/k, 2/k, \ldots, 1 - 1/k}_{k-1}, \quad \underbrace{0, \ldots, 0}_{4k-2})^\top.$$

This yields $\nabla f(x^*) = (-(1-\frac{1}{k}), \ldots, -(1-\frac{1}{k}), -1, \ldots, -1)^\top$ as well as $\nabla g(x^*)\lambda^* = (1 - \frac{1}{k}, \ldots, 1 - \frac{1}{k}, 1, \ldots, 1)^\top$ and thus $\nabla L(x^*, \lambda^*) \equiv 0$. This means the first KKT-condition, stationarity, holds.

Since $g(x^*)$ is 0 for all entries from 1 to $2k-2$ and λ^* is 0 for the rest, we get $(\lambda^*)^\top g(x^*) \equiv 0$, fulfilling the second KKT-condition, complementary slackness. Obviously, x^* is also feasible and $\lambda^* \geq 0$.

Thus, x^* is a KKT-point and global optimum. The optimal value of the original linear program as well as the underlying flow problem is $-f(x^*) = k - 1$. □

Since we defined scaled robust flows on networks without non-interdictable arcs, the above example is technically not permitted. We will now use the notions of multi-arc networks and flows to show that for $c > \frac{1}{k}$ we can still get the same result, even when all arcs are interdictable.

Lemma 3.26. *Let $c > \frac{1}{k}$ and let N be the network from Example 3.24. Then there exists an $M \in \mathbb{N}$, s.t. N_M has an optimal NF-value of k and the optimal ROB_c-flow has an NF-value less than k.*

Proof. Any flow in this proof with index M is assumed to run through the network N_M. Any flow without index runs through the network N. Any values are calculated on the appropriate networks.

First consider an arbitrary $M \in \mathbb{N}$ to be determined later. Consider again the NF-optimal flow x^*, which sends 1 unit of flow along every path P_i and 0 units of flow along $\bar{P}_j$. We want to prove that x^*_M is uniquely NF-optimal on N_M with value k.

Since NF-values are unchanged by switching to multi-arc networks, x^*_M is still NF-optimal on N_M with value k. Thus we only have to prove uniqueness.

Let $x_M \neq x^*_M$ be any flow different from x^*_M on N_M. Consider first the case that x_M does not send any flow along the paths $\bar{P}_j$. Since $x_M \neq x^*_M$, the flow over the paths P_i must be strictly smaller than that of x^*_M. Hence $\mathrm{val}(x_M) < k$.

The second case is that x_M sends at least some amount of flow along some path $\bar{P}_j$. But then the two adjacent paths P_j and P_{j+1} both lose this amount of available capacity. This gives a smaller total possible flow value, which cannot be recovered through other paths because of the structure of the network. Hence again $\mathrm{val}(x_M) < k$.

Thus x^*_M is uniquely NF-optimal on N_M. Since the flow can be interdicted entirely, its scaled robust value is

$$\begin{aligned}\mathrm{val}_c(x^*_M) &= k - c \cdot k \\ &= k - \frac{1}{k} \cdot k - (c - \frac{1}{k}) \cdot k \\ &= k - 1 - (c - \frac{1}{k}) \cdot k.\end{aligned}$$

For the next step, we define $\varepsilon := (c - \frac{1}{k}) \cdot k > 0$. Consider again the flow $\bar{x}$, which sends 0 units of flow along every path P_i and 1 unit of flow along $\bar{P}_j$. Recall that $\mathrm{val}_c(\bar{x}) = k - 1$.

By Lemma 3.23, we know that for the above ε there exists an $\bar{M} \in \mathbb{N}$, s.t. $|\mathrm{val}_c(\bar{x}) - \mathrm{val}_c(\bar{x}_{\bar{M}})| < \varepsilon$. We now consider only the network $N_{\bar{M}}$ for the rest of the proof.

We have already proven that $x^*_{\bar{M}}$ is uniquely NF-optimal with value k and that $\mathrm{val}_c(x^*_{\bar{M}}) = k - 1 - (c - \frac{1}{k}) \cdot k$. But $\bar{x}_{\bar{M}}$ has scaled robust value

$$\begin{aligned}\mathrm{val}_c(\bar{x}_{\bar{M}}) &> \mathrm{val}_c(\bar{x}) - \varepsilon \\ &= k - 1 - \varepsilon \\ &= k - 1 - (c - \frac{1}{k}) \cdot k.\end{aligned}$$

Therefore, $x^*_{\bar{M}}$ is not ROB_c-optimal on N_M. But, since $x^*_{\bar{M}}$ is the unique NF-optimum on N_M, we know that the ROB_c-optimum is not NF-optimal. This concludes the proof. □

A counter-example to Lemma 3.11 for values of c smaller than $-\frac{1}{k}$ can be constructed very similarly to Example 3.24. We use the same graph, except only the arcs (u_{i+1}, v_i) are interdictable now. With careful argumentation it can be shown that the flow $\bar{x}$ is ROB_c-optimal, but not NF-optimal. The result can then be extended to networks without non-interdictable arcs.

3.7 Non-linear and arc-based scaling functions

We originally introduced a variation to Maximum Robust Flow by defining a scaling function f and the f-scaled robust value as in (3.2). So far, we have concentrated on the case when f is linear. Now, we want to start considering general functions $f : \mathbb{R} \to \mathbb{R}$.

Definition 3.27. *The problem of finding a feasible flow in N with maximum f-scaled robust value is called* Maximum F-Scaled Robust Flow *and is denoted by ROB_f.*

Analogously to how we constructed the LP for linearly scaled robust flows, we can build an optimization program for ROB_f. Again, let $\lambda \in \mathbb{R}$ be a variable measuring the amount of interdicted flow, i.e.

$$\lambda = \max_{S \in \mathcal{S}} \sum_{P \in \mathcal{P}_S} x(P).$$

The optimization program for ROB_f is given by:

$$\begin{aligned} \max_{x,\lambda} \quad & \sum_{P \in \mathcal{P}} x(P) - \lambda & & \\ \text{s.t.} \quad & \sum_{P \in \mathcal{P}_e} x(P) \leq u(e) & & \forall e \in E \\ & \sum_{P \in \mathcal{P}_S} f(x(P)) \leq \lambda & & \forall S \in \mathcal{S} \\ & x \geq 0 & & \end{aligned} \tag{3.10}$$

If f is not linear, we cannot move the scaling operator to the objective function as we have done with the factor c in (3.5). What is more, if f is not linear, the above program is not even an LP, making hardness conclusions difficult.

One case which can be analyzed a bit more thorougly is when f is affine linear, i.e. $f(x) = c \cdot x + d$, with $c \in \mathbb{R}_0^+, d \in \mathbb{R}$. The second set of inequalities of (3.10) can

be written as

$$\begin{aligned}
&\sum_{P\in\mathcal{P}_S}(c\cdot x(P)+d)\le\lambda \quad \forall S\in\mathcal{S}\\
\iff & c\cdot\sum_{P\in\mathcal{P}_S}x(P)+d\cdot|\mathcal{P}_S|\le\lambda \quad \forall S\in\mathcal{S}\\
\iff & c\cdot\sum_{P\in\mathcal{P}_S}x(P)-\lambda\le -d\cdot|\mathcal{P}_S| \quad \forall S\in\mathcal{S}
\end{aligned}$$

We introduce the dual variables $y(e), e\in E$ and $z'(S), S\in\mathcal{S}$. The dual is constructed very similarly to the linear case, only giving a slightly more complicated objective function and differing coefficients:

$$\begin{aligned}
\min_{y,z}\quad & \sum_{e\in E}u(e)y(e)-d\cdot\sum_{S\in\mathcal{S}}|\mathcal{P}_S|\cdot z'(S)\\
\text{s.t.}\quad & \sum_{e\in P}y(e)+c\cdot\sum_{S\in\mathcal{S}:P\cap S\neq\emptyset}z'(S)\ge 1 \qquad \forall P\in\mathcal{P}\\
& \sum_{S\in\mathcal{S}}z'(S)=1\\
& y,z\ge 0
\end{aligned}$$

To make it easier to compare this program to previous cases, we define $z(S):=c\cdot z'(S)$, which yields

$$\begin{aligned}
\min_{y,z}\quad & \sum_{e\in E}u(e)y(e)-\frac{d}{c}\cdot\sum_{S\in\mathcal{S}}|\mathcal{P}_S|\cdot z(S)\\
\text{s.t.}\quad & \sum_{e\in P}y(e)+\sum_{S\in\mathcal{S}:P\cap S\neq\emptyset}z(S)\ge 1 \qquad \forall P\in\mathcal{P}\\
& \sum_{S\in\mathcal{S}}z(S)=c\\
& y,z\ge 0.
\end{aligned}$$

The set of feasible solutions of this program is equivalent to the set of feasible solutions (3.6) of the dual of ROB_c. Hence by the proof of Lemma 3.6, the above LP is solvable in polynomial time for $k=1$ and $c\in\mathbb{Q}$. Therefore, ROB_f with affine linear scaling function f and $c\in\mathbb{Q}$ is solvable in polynomial time for $k=1$.

For the final batch of results in this chapter, we take a look at arc-based interdiction. To be able to talk about this problem, we have to change perspectives from path-based to arc-based flows. Recall that the value of an arc-based flow f was determined by an auxiliary arc $h=(t,s)$ with infinite capacity:

$$\mathrm{val}(f)=f(h)$$

The *arc-based robust value* of f is then

$$\mathrm{val}_{\mathrm{arc}}(f)=f(h)-\max_{S\in\mathcal{S}}\sum_{e\in S}f(e).$$

Note that this definition is not equivalent to the original robust value (3.1). Even though arc-based and path-based flows can be converted into each other without losing value, it might happen in the above definition that paths get interdicted multiple times on different arcs. In the original robust value definition, these paths only counted once towards the interdiction value.

Definition 3.28. *The problem of finding a feasible flow in N with maximum arc-based robust value is called* Maximum Arc-Based Robust Flow *and is denoted by ROB_{arc}.*

We now want to construct the LP describing this problem. For that, we again introduce a variable λ measuring interdicted flow. The rest of the constraints below are necessary to guarantee capacity compliance and flow conservation.

$$\begin{aligned} \max_{f} \quad & f(h) - \lambda & & \\ \text{s.t.} \quad & f(e) \leq u(e) & & \forall e \in E \\ & \sum_{e \in E_{\text{in}}(v)} f(e) = \sum_{e \in E_{\text{out}}(v)} f(e) & & \forall v \in V \\ & \sum_{e \in S} f(e) \leq \lambda & & \forall S \in \mathcal{S} \\ & f \geq 0 & & \end{aligned} \tag{3.11}$$

We do not need to introduce the slightly more complicated dual of this problem, since we can prove compactness of the above LP. The ellipsoid method [Kha79], introduced by Khachiyan in 1979, then implies polynomiality of the problem.

Lemma 3.29. *ROB_{arc} for fixed parameter k is solvable in polynomial time.*

Proof. The number of variables in the LP (3.11) is $|E|+1$ and therefore polynomial. The number of constraints is $2|E| + |V| + |\mathcal{S}|$. If k is a fixed natural number, then $|\mathcal{S}| = \binom{|E|}{k}$ is a polynomial in $|E|$ and therefore the number of constraints is also polynomial. Hence (3.11) is a compact LP.

A compact linear program can be solved in polynomial time using the ellipsoid method [Kha79]. □

We can of course also introduce scaling on the interdictor value of arc-based robust flows. For all linear scaling functions, polynomiality can be achieved the same way as above using a compact LP formulation.

4 Optional reroutable flows

In this chapter we aim to first introduce Maximum Reroutable Flow and Maximum Strictly Reroutable Flow, then describe variations in which rerouting is optional. These problems will be called Maximum Optional Reroutable Flow and Maximum Optional Strictly Reroutable Flow respectively. Again, hardness results are presented for both the original as well as the variations. We investigate an approximation algorithm for the optional case and present some results for unit capacity networks. Lastly, we compare reroutable flows with some other problems from the thesis. Especially ordinary robust flows are of high interest since they yield lower bounds and a reduction result.

4.1 The original reroutable flow problem

In the first section of this chapter, we establish the problem of rerouting flows in a network where a single arc has failed. A given flow will be called reroutable if, after failure of the arc, the lost flow can be rerouted along some alternative path in the network subject to capacity constraints. Strictly reroutable flows strengthen this definition by restricting the available capacities for rerouting further. The goal is to maximize the value of a given (strictly) reroutable flow. The problems were first examined by Matuschke, McCormick and Oriolo [MMO17].

The following section gives formal definitions for both versions of the problem as well as several hardness results.

4.1.1 Definition

Once again, we are given a network $N = (G, s, t, u)$ as defined in chapter 2. As before, $\mathcal{P}$ denotes the set of s-t-paths in G, and we only consider path-based s-t-flows. Recall that the network flow value for a path-based flow x is

$$\text{val}(x) = \sum_{P \in \mathcal{P}} x(P).$$

Similarly, recall the definition of the set of all paths intersecting with a given arc $e \in E$:

$$\mathcal{P}_e := \{P \in \mathcal{P} : e \in P\}$$

Given two arcs $\bar{e}, e \in E$, we introduce the more detailed notion of all paths intersecting $\bar{e}$ before e:

$$\mathcal{P}_{\bar{e} \to e} := \{P \in \mathcal{P} : \ \bar{e}, e \in P, \ P \text{ traverses } \bar{e} \text{ before } e\}$$

Note, that if $\bar{e} = e$, we assume $\mathcal{P}_{\bar{e}\to e} = \emptyset$.

Let x be a feasible s-t-flow. If an arc $\bar{e} = (v, w)$ fails, any flow on paths containing this arc is interrupted, as in the problem ROB from chapter 3. Unlike before though, we now must reroute the interrupted flow starting at the tip v of the destroyed arc. For that, we define the *available capacity* $\bar{u}_{x,\bar{e}}(e)$ of an arc $e \in E \setminus \{\bar{e}\}$ after failure of $\bar{e}$:

$$\bar{u}_{x,\bar{e}}(e) := u(e) - \sum_{P \in \mathcal{P} \setminus \mathcal{P}_{\bar{e}\to e}} x(P) \tag{4.1}$$

Now consider the network $\bar{N} = ((V, E \setminus \{\bar{e}\}), v, t, \bar{u}_{x,\bar{e}})$. A v-t-flow $x_{\bar{e}}$ in $\bar{N}$ is called *rerouting* of x for the failing arc $\bar{e}$ if it has value $x(\bar{e})$. The flow x is *reroutable* if there exists a rerouting $x_{\bar{e}}$ for all failing arcs $\bar{e} \in E$.

Definition 4.1. *The problem of finding a feasible reroutable flow in N with maximum value is called* Maximum Reroutable Flow *and is denoted by RF.*

A slightly more restrictive version of this definition are strictly reroutable flows. In the previous case, the capacities of flow that was interrupted by the interdictor were available for rerouting. This can be seen in (4.1), where the paths in $\mathcal{P}_{\bar{e}\to e}$ are not substracted from the arc capacity. Strictly reroutable flows do not allow this anymore. The available capacity of an arc $e \in E \setminus \{\bar{e}\}$ after failure of $\bar{e}$ is

$$\bar{u}_x(e) := u(e) - \sum_{P \in \mathcal{P}_e} x(P) = u(e) - x(e). \tag{4.2}$$

A v-t-flow $x_{\bar{e}}$ in the network $\bar{N}_{\text{strict}} = ((V, E \setminus \{\bar{e}\}), v, t, \bar{u}_x)$ is called *strict rerouting* of x for the failing arc $\bar{e}$ if it has value $x(\bar{e})$. The flow x is *strictly reroutable* if there exists a strict rerouting $x_{\bar{e}}$ for all failing arcs $\bar{e} \in E$.

Definition 4.2. *The problem of finding a feasible strictly reroutable flow in N with maximum value is called* Maximum Strictly Reroutable Flow *and is denoted by SRF.*

4.1.2 Hardness results

In [MMO17], it is proven that SRF is solvable in polynomial time via a separation approach. This approach is especially interesting since it can be adapted to prove polynomiality of our variation, see section 4.3. It is also proven that RF, on the other hand, is NP-hard, even when restricted to capacities in $\{1, 2\}$. Another result introduced is a tight 2-approximation for RF, which can be adapted to our variation of the problem as well.

[MMO17] also show that, for unit capacity networks, RF and SRF are equivalent and every reroutable flow can be converted to a strictly reroutable flow of the same value. Finally, they prove that RF and SRF are NP-hard when considering multiple arc failures, even when only 2 arcs fail and all capacities are equal to 1.

4.2 Definition of optional reroutable flows

The problems RF and SRF are characterized by the fact that rerouting interdicted flow is mandatory. Applications for this model include things such as shipping unique goods to households, as in the postal system, or generally any situation where nothing sent may get lost to interdiction.

We now want to introduce a new model where some amount of the interdicted flow is allowed to be lost. After the interdiction happened, we may reroute flow as before, but if there exists no possibility to do that or there are alternatives without rerouting, we may refrain partly or entirely from doing it. Possible applications for this model could be electricity or water supply networks, where some flow is allowed to get lost as long as the consumer's needs are fulfilled.

In order for this model to work, we will have to introduce a new objective function which depends on the arc selected for interdiction, since that will now influence the value of the flow. The new objective function is similar to the robust value introduced in chapter 3. Thus, we can view optional reroutable flows as a mixture of robust flows and reroutable flows.

Formally, we are again given a network N and the set of simple paths $\mathcal{P}$ in N. Let x be a feasible s-t-flow. If an arc $\bar{e} = (v, w)$ fails, we define the available capacity $u_{x,\bar{e}}(e)$ for all arcs $e \in E \setminus \{\bar{e}\}$ as before in equation (4.1).

A v-t-flow $x_{\bar{e}}$ in the network $\bar{N} = ((V, E \setminus \{\bar{e}\}), v, t, \bar{u}_{x,\bar{e}})$ is called *optional rerouting* of x for the failing arc $\bar{e}$ if its value lies in $[0, x(\bar{e})]$. Let $\mathcal{F}(x, \bar{e})$ be the set of all optional reroutings of x if $\bar{e}$ fails.

The *optional reroutable value* of x is then composed of the flow value minus the worst possible interdiction, taking into account possibly saved flow by rerouting. Formally:

$$\mathrm{val}_{\mathrm{ORF}}(x) = \sum_{P \in \mathcal{P}} x(P) \; - \; \max_{\bar{e} \in E} \left(x(\bar{e}) - \max_{x_{\bar{e}} \in \mathcal{F}(x,\bar{e})} \mathrm{val}(x_{\bar{e}}) \right),$$

where as always $x(e) = \sum_{P \in \mathcal{P}_e} x(P)$.

Definition 4.3. *The problem of finding a feasible flow in N with maximum optional reroutable value is called* Maximum Optional Reroutable Flow *and is denoted by ORF.*

For the strict version, consider the alternate definition of the available capacity $\bar{u}_x(e)$ for $e \in E \setminus \{\bar{e}\}$ from equation (4.2). A v-t-flow $x_{\bar{e}}$ in the network $\bar{N}_{\mathrm{strict}} = ((V, E \setminus \{\bar{e}\}), v, t, \bar{u}_x)$ is called *optional strict rerouting* of x for the failing arc $\bar{e}$ if its value lies in $[0, x(\bar{e})]$. Let $\mathcal{F}_{\mathrm{strict}}(x, \bar{e})$ be the set of all optional strict reroutings of x if $\bar{e}$ fails.

The *optional strictly reroutable value* of x is defined by

$$\mathrm{val}_{\mathrm{OSRF}}(x) = \sum_{P \in \mathcal{P}} x(P) \; - \; \max_{\bar{e} \in E} \left(x(\bar{e}) - \max_{x_{\bar{e}} \in \mathcal{F}_{\mathrm{strict}}(x,\bar{e})} \mathrm{val}(x_{\bar{e}}) \right).$$

Definition 4.4. *The problem of finding a feasible flow in N with maximum optional strictly reroutable value is called* Maximum Optional Strictly Reroutable Flow *and is denoted by OSRF.*

The rest of this chapter is organized in the following way: In section 4.3 we talk about hardness results that can be adopted from the original reroutable flows to the optional version. In the next section, an approximation algorithm for the non-strict version is presented. Lastly, we take a look at results in unit capacity networks.

4.3 Hardness results

4.3.1 LP formulation

For the analysis of the complexity of the problems, we first need to establish the LP formulations. We start with the strict version of the problem, since the LP will be slightly easier to construct.

First, consider the program for the non-optional SRF problem. For that, let $\bar{e}$ be the failing arc of the network and let $\mathcal{R}(\bar{e})$ be the set of all v-t-paths in the network $\bar{N}$, i.e. all paths that a rerouting can use when $\bar{e}$ fails. Note, that $\mathcal{R}(\bar{e})$ is defined on the non-strict version of the rerouting network. This is deliberate to simplify notation, it can be used for the strict version without any further adjustments.

Let $\bar{x} = (x_{e_1}, \dots, x_{e_{|E|}})$ be a vector containing the (strict) rerouting variables for all arcs $e \in E$. The following LP for SRF is taken directly from [MMO17]:

$$\begin{aligned}
\max_{x,\bar{x}} \quad & \sum_{P \in \mathcal{P}} x(P) \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_e} x(P) + \sum_{R \in \mathcal{R}(\bar{e}) : e \in R} x_{\bar{e}}(R) \leq u(e) && \forall e, \bar{e} \in E \\
& \sum_{P \in \mathcal{P}_{\bar{e}}} x(P) - \sum_{R \in \mathcal{R}(\bar{e})} x_{\bar{e}}(R) = 0 && \forall \bar{e} \in E \\
& x, x_{\bar{e}} \geq 0 && \forall \bar{e} \in E
\end{aligned} \tag{4.3}$$

The first set of constraints make sure any rerouting $x_{\bar{e}}$ fulfills capacity constraints on all arcs $e \in E$. Note that if $\bar{e} = e$, the set $\{R \in \mathcal{R}(\bar{e}) : e \in R\}$ is empty and the second term becomes 0, ensuring feasibility of the original flow x. The second set of constraints ensure that the rerouting $x_{\bar{e}}$ has value $x(\bar{e})$.

To transform this into an LP for the optional case, we again introduce a variable $\lambda \in \mathbb{R}$, this time measuring the maximum difference between the value of the failing arc $\bar{e}$ and the corresponding value of $x_{\bar{e}}$. In other words, we want to have

$$\lambda = \max_{\bar{e} \in E} \left(\sum_{P \in \mathcal{P}_{\bar{e}}} x(P) - \sum_{R \in \mathcal{R}(\bar{e})} x_{\bar{e}}(R) \right). \tag{4.4}$$

To achieve this, we update the second set of constraints of (4.3):

$$\sum_{P\in\mathcal{P}_{\bar{e}}} x(P) - \sum_{R\in\mathcal{R}(\bar{e})} x_{\bar{e}}(R) \leq \lambda \quad \forall \bar{e} \in E \tag{4.5}$$

Additionally, we substract λ from the objective function. Since the LP is a maximization, it will choose the smallest λ fulfilling the inequalities, which is exactly the value of (4.4). Simultaneously, it will try to choose the values $x_{\bar{e}}(R)$ as large as possible, thus making λ potentially even smaller.

Altogether, the LP for OSRF looks like this:

$$\begin{aligned}
\max_{x,\bar{x},\lambda} \quad & \sum_{P\in\mathcal{P}} x(P) - \lambda \\
\text{s.t.} \quad & \sum_{P\in\mathcal{P}_e} x(P) + \sum_{R\in\mathcal{R}(\bar{e}):e\in R} x_{\bar{e}}(R) \leq u(e) && \forall e, \bar{e} \in E \\
& \sum_{P\in\mathcal{P}_{\bar{e}}} x(P) - \sum_{R\in\mathcal{R}(\bar{e})} x_{\bar{e}}(R) \leq \lambda && \forall \bar{e} \in E \\
& x, x_{\bar{e}}, \lambda \geq 0 && \forall \bar{e} \in E
\end{aligned} \tag{4.6}$$

Observe that, if we set all variables in $\bar{x}$ to 0, i.e. if we allow for no rerouting at all, the above LP is in fact equal to the primal LP of ROB (3.5). For a more rigorous comparison of the two problems, see section 4.5.

The only thing not captured explicitly by (4.6) is the fact that we need to make sure the rerouting for an arc $\bar{e} \in E$ does not send more than $x(\bar{e})$ of flow value. One way to ensure this restriction is to introduce an additional constraint, i.e.

$$\sum_{P\in\mathcal{P}_{\bar{e}}} x(P) - \sum_{R\in\mathcal{R}(\bar{e})} x_{\bar{e}}(R) \geq 0. \tag{4.7}$$

It turns out that, because of $\lambda \geq 0$, this is unnecessary. Assume $(x, \bar{x}, \lambda)$ is a feasible point of the LP (4.6) with $\sum_{P\in\mathcal{P}_{\bar{e}}} x(P) - \sum_{R\in\mathcal{R}(\bar{e})} x_{\bar{e}}(R) < 0$ for some $\bar{e} \in E$. $\lambda \geq 0$ makes sure that, despite this, there can not be any 'rerouting overflow' affecting the objective function. Thus, we can simply reduce an appropriate set of paths in the rerouting, s.t. $\sum_{P\in\mathcal{P}_{\bar{e}}} x(P) - \sum_{R\in\mathcal{R}(\bar{e})} x'_{\bar{e}}(R) = 0$, where $x'_{\bar{e}}$ is the new rerouting. Then $(x, \bar{x}', \lambda)$ is a feasible point with the same objective value as $(x, \bar{x}, \lambda)$ and additionally fulfilling (4.7).

Since larger values for $x_{\bar{e}}(R)$ make it more difficult to fulfill the first set of constraints in (4.6), it is unlikely that the solution of the LP does not fulfill (4.7). But if that is the case, we can use the above procedure to compute a correct solution easily.

To transform this LP into its dual form, let $y_{\bar{e}}(e), \ \forall \bar{e}, e \in E$ and $z(e), \ \forall e \in E$ be the dual variables corresponding to the first and second set of constraints of (4.6). The second set of constraints can be rewritten as

$$\sum_{P\in\mathcal{P}_{\bar{e}}} x(P) - \sum_{R\in\mathcal{R}(\bar{e})} x_{\bar{e}}(R) - \lambda \leq 0 \quad \forall \bar{e} \in E.$$

Thus, the objective function of the dual does not depend on $z(e)$. It is simply

$$\sum_{e,\bar{e}\in E} u(e)y_{\bar{e}}(e).$$

The first set of constraints in the dual corresponds to the variables $x(P)$. They have a factor of 1 in the primal objective and appear in both sets of constraints. Carefully identifying the exact inequalities they appear in yields the following constraint for the dual:

$$\sum_{e\in P}\left(\sum_{\bar{e}\in E} y_{\bar{e}}(e) + z(e)\right) \geq 1 \quad \forall P \in \mathcal{P}$$

The variables $x_{\bar{e}}(R)$ also appear in both set of constraints. They appear multiple times in the first set, and only once for fixed $\bar{e}$ and $R \in \mathcal{R}(\bar{e})$ in the second set. Their associated factor in the primal objective is 0.

$$\sum_{e\in R} y_{\bar{e}}(e) - z(\bar{e}) \geq 0 \quad \forall \bar{e} \in E, R \in \mathcal{R}(\bar{e})$$

Finally, the variables λ appear only in every constraint of the second set and have an associated factor of -1. This gives

$$-\sum_{\bar{e}\in E} z(\bar{e}) \geq -1.$$

Overall, the dual LP of OSRF is given by

$$\begin{aligned}
\min_{y,z} \quad & \sum_{e,\bar{e}\in E} u(e)y_{\bar{e}}(e) \\
\text{s.t.} \quad & \sum_{e\in P}\left(\sum_{\bar{e}\in E} y_{\bar{e}}(e) + z(e)\right) \geq 1 && \forall P \in \mathcal{P} \\
& \sum_{e\in R} y_{\bar{e}}(e) - z(\bar{e}) \geq 0 && \forall \bar{e} \in E, R \in \mathcal{R}(\bar{e}) \\
& -\sum_{\bar{e}\in E} z(\bar{e}) \geq -1 \\
& y, z \geq 0.
\end{aligned} \tag{4.8}$$

For the non-strict version of the problem, the only thing changing is the capacities available for rerouting. This becomes apparent in the capacity constraints of (4.6), which are altered to exclude the paths $\mathcal{P}_{\bar{e}\to e}$.

The primal LP of ORF is therefore

$$\begin{aligned}
\max_{x,\bar{x},\lambda} \quad & \sum_{P\in\mathcal{P}} x(P) - \lambda \\
\text{s.t.} \quad & \sum_{P\in\mathcal{P}_e\setminus\mathcal{P}_{\bar{e}\to e}} x(P) + \sum_{R\in\mathcal{R}(\bar{e}):e\in R} x_{\bar{e}}(R) \le u(e) && \forall e,\bar{e}\in E \\
& \sum_{P\in\mathcal{P}_{\bar{e}}} x(P) - \sum_{R\in\mathcal{R}(\bar{e})} x_{\bar{e}}(R) \le \lambda && \forall \bar{e}\in E \\
& x, x_{\bar{e}}, \lambda \ge 0 && \forall \bar{e}\in E.
\end{aligned} \tag{4.9}$$

Again, $\lambda \ge 0$ implicitly makes sure that no more than $x(\bar{e})$ units are sent in every rerouting $x_{\bar{e}}$. The dual of ORF is constructed equivalently to that of OSRF, the only difference being once more the missing paths $\mathcal{P}_{\bar{e}\to e}$:

$$\begin{aligned}
\min_{y,z} \quad & \sum_{e,\bar{e}\in E} u(e) y_{\bar{e}}(e) \\
\text{s.t.} \quad & \sum_{e\in P}\left(\sum_{\bar{e}\in E:P\notin\mathcal{P}_{\bar{e}\to e}} y_{\bar{e}}(e) + z(e)\right) \ge 1 && \forall P\in\mathcal{P} \\
& \sum_{e\in R} y_{\bar{e}}(e) - z(\bar{e}) \ge 0 && \forall \bar{e}\in E, R\in\mathcal{R}(\bar{e}) \\
& -\sum_{\bar{e}\in E} z(\bar{e}) \ge -1 \\
& y, z \ge 0
\end{aligned} \tag{4.10}$$

4.3.2 Hardness of OSRF

In this section we prove that OSRF can be solved in polynomial time. We will again use the equivalence of optimization and separation as well as a shortest path computation based on [MMO17]. We will also investigate why this method does not work to prove polynomiality for the non-strict version.

Lemma 4.5. *The dual LP of* Maximum Optional Strictly Reroutable Flow *can be solved in polynomial time.*

Proof. Consider the polyhedron Q of all feasible solutions of the dual program (4.8):

$$\begin{aligned}
Q := \Bigg\{ (y,z)\in\mathbb{R}^{E^2\times E}_{\ge 0} : \quad & \sum_{e\in P}\left(\sum_{\bar{e}\in E} y_{\bar{e}}(e) + z(e)\right) \ge 1 \quad \forall P\in\mathcal{P}, \\
& \sum_{e\in R} y_{\bar{e}}(e) - z(\bar{e}) \ge 0 \qquad \forall \bar{e}\in E, R\in\mathcal{R}(\bar{e}), \\
& -\sum_{\bar{e}\in E} z(\bar{e}) \ge -1 \qquad\qquad \Bigg\}
\end{aligned}$$

If we are able to find a polynomial oracle that solves the separation problem 2.15 for Q, then, by Theorem 2.16, the corresponding program is also solvable in polynomial time.

Consider the following oracle for the separation of Q:

Input: $(y, z) \in \mathbb{R}_{\geq 0}^{E^2 \times E}$, Network N
If not $-\sum_{\bar{e} \in E} z(\bar{e}) \geq -1$:
  Output $(y, z) \notin Q$
Compute the shortest s-t-path P' in N w.r.t. arc costs $c(e) = \sum_{\bar{e} \in E} y_{\bar{e}}(e) + z(e)$
If not $\sum_{e \in P'} c(e) \geq 1$:
  Output $(y, z) \notin Q$
For all $\bar{e} \in E$:
  Compute the shortest tail$(\bar{e})$-t-path R' in $\bar{N}$ w.r.t. arc costs $d(e) = y_{\bar{e}}(e)$
  If not $\sum_{e \in R'} d(e) \geq z(\bar{e})$:
    Output $(y, z) \notin Q$
Output $(y, z) \in Q$

If the first if-clause in the oracle is false, $-\sum_{\bar{e} \in E} z(\bar{e}) \geq -1$ already represents a violated inequality. If, for some path P' or R', the second or third if clauses in the oracle evaluate to false, then a violated inequality is given by $\sum_{e \in P'}(\sum_{\bar{e} \in E} y_{\bar{e}}(e) + z(e)) \geq 1$ or $\sum_{e \in R'} y_{\bar{e}}(e) \geq z(\bar{e})$ respectively. If, on the other hand, (y, z) fulfills $\sum_{e \in P'} c(e) \geq 1$ for the shortest s-t-path P' and $\sum_{e \in R'} d(e) \geq z(\bar{e})$ for all shortest paths R', then by the choice of the arc costs in the procedure, all path inequalities are fulfilled as well and the oracle outputs $(y, z) \in Q$ correctly.

We have to argue that the above procedure runs in polynomial time. All if-statements are single equations that can be checked in constant time. Computing a shortest path is polynomial, we can use Dijkstra's algorithm again [Dij59]. In the loop we do this polynomially many times, resulting in a polynomial overall runtime.

By Theorem 2.16, the dual program (4.8) is solvable in polynomial time. □

Given a solution for the dual and the set of inequalities generated during separation, we can solve the primal program as well. Therefore, OSRF is solvable in polynomial time.

In [MMO17], Matuschke, McCormick and Oriolo also present an alternative proof for the polynomiality of SRF. They remark that in the strict case, all capacities available for rerouting only depend on arc flow values. Thus, SRF can be formulated as a compact LP with arc flow variables. Using the ellipsoid method [Kha79], the LP can be solved in polynomial time.

This works for OSRF as well. The resulting LP has capacity constraints, flow conservation constraints for the original flow as well as the reroutings and one set of λ-constraints corresponding to (4.5), which ensure the correct amount of flow is substracted from the objective function. The number of variables and constraints is at most quadratic in the number of arcs and nodes. To obtain a maximum optional strictly reroutable flow, all that is left to do is to decompose the resulting arc flow as in Proposition 2.7.

Unfortunately, the approach via shortest path computations does not work for the non-strict version of the problem. In the polyhedron of all feasible solutions of the dual (4.10), the constraint

$$\sum_{e \in P} \left(\sum_{\bar{e} \in E : P \notin \mathcal{P}_{\bar{e} \to e}} y_{\bar{e}}(e) + z(e) \right) \geq 1 \quad \forall P \in \mathcal{P}$$

appears. We cannot define reasonable arc costs for a given arc e, since the argument inside the outer sum now depends on the path variable.

[MMO17] prove that the non-optional, non-strict version of the problem, RF, is NP-hard, even when restricted to instances with capacities in $\{1, 2\}$. They use so-called *backup-links* as a central component of their proof. By construction, backup-links can only be used for rerouting and have as much capacity as needed. Since rerouting is mandatory in RF, two consecutive arcs suffice to create a backup-link because then rerouting from the middle node is impossible and no flow may be sent along the link.

Unfortunately, making rerouting optional ruins the idea of backup-links, since there is no way to guarantee that the flow player does not send any flow along the links. Therefore, the NP-hardness proof for RF could not be adapted to the optional version of the problem.

4.4 An approximation algorithm for optional reroutable flows

In the strict version of our problems, we do not free up capacities of interrupted flow for rerouting. Since that is the only difference to the non-strict problems, it is obvious that the optimal values of OSRF and SRF are at most as large as the optimal values of SRF and RF respectively. [MMO17] prove that the gap between SRF and RF may not be larger than a factor of 2. This result can be adopted for OSRF and ORF as well, yielding a 2-approximation for the latter.

Lemma 4.6. *Let $(x, \bar{x}, \lambda)$ be a feasible point of the primal LP of OSRF (4.6). Then $(x, \bar{x}, \lambda)$ is feasible for the primal LP of ORF (4.9) with the same objective value.*

Let $(x, \bar{x}, \lambda)$ be a feasible point of the primal LP of ORF. Then there exists a point $(x', \bar{x}', \lambda')$ which is feasible for the primal LP of OSRF with an objective value of at least half that of $(x, \bar{x}, \lambda)$.

Proof. If $(x, \bar{x}, \lambda)$ is a feasible point of (4.6), then clearly $(x, \bar{x}, \lambda)$ is also feasible for (4.9), since all constraints of the non-strict version are relaxations of their strict counterparts.

On the other hand, let $(x, \bar{x}, \lambda)$ be a feasible point of the non-strict LP. Define $(x', \bar{x}', \lambda') = \frac{1}{2}(x, \bar{x}, \lambda)$ and let $e, \bar{e} \in E$. Then, using the feasibility of $(x, \bar{x}, \lambda)$, we

have

$$\begin{aligned}\sum_{R\in\mathcal{R}(\bar{e}):e\in R} x'_{\bar{e}}(R) &= \frac{1}{2}\sum_{R\in\mathcal{R}(\bar{e}):e\in R} x_{\bar{e}}(R)\\ &\leq \frac{1}{2}\left(u(e) - \sum_{P\in\mathcal{P}_e\setminus\mathcal{P}_{\bar{e}\to e}} x(P)\right)\\ &= \frac{1}{2}\left(u(e) - \sum_{P\in\mathcal{P}_e} x(P) + \sum_{P\in\mathcal{P}_{\bar{e}\to e}} x(P)\right).\end{aligned}$$

Since the flow x fulfills flow capacities, we know $\sum_{P\in\mathcal{P}_{\bar{e}\to e}} x(P) \leq u(e)$. Thus we can continue the above line of inequalities:

$$\begin{aligned}\frac{1}{2}\left(u(e) - \sum_{P\in\mathcal{P}_e} x(P) + \sum_{P\in\mathcal{P}_{\bar{e}\to e}} x(P)\right) &\leq \frac{1}{2}\left(u(e) - \sum_{P\in\mathcal{P}_e} x(P) + u(e)\right)\\ &= u(e) - \frac{1}{2}\sum_{P\in\mathcal{P}_e} x(P)\\ &= u(e) - \sum_{P\in\mathcal{P}_e} x'(P)\end{aligned}$$

Altogether, this proves that the first set of constraints of the LP for OSRF is fulfilled for $(x', \bar{x}', \lambda')$. Similarly, we can prove that the second set of constraints hold:

$$\begin{aligned}\sum_{P\in\mathcal{P}_{\bar{e}}} x'(P) - \sum_{R\in\mathcal{R}\bar{e}} x'_{\bar{e}}(R) &= \frac{1}{2}\left(\sum_{P\in\mathcal{P}_{\bar{e}}} x(P) - \sum_{R\in\mathcal{R}\bar{e}} x_{\bar{e}}(R)\right)\\ &\leq \frac{1}{2}\cdot\lambda\\ &= \lambda'\end{aligned}$$

Obviously, all entries of $(x', \bar{x}', \lambda')$ are non-negative, since the original point was non-negative. Hence, $(x', \bar{x}', \lambda')$ is feasible for the primal LP of OSRF. It remains to be shown that the objective value is at least half that of the original point. This actually holds with equality:

$$\mathrm{val}(x', \bar{x}', \lambda') = \sum_{P\in\mathcal{P}} x'(P) - \lambda' = \frac{1}{2}\left(\sum_{P\in\mathcal{P}} x(P) - \lambda\right) = \frac{1}{2}\cdot\mathrm{val}(x, \bar{x}, \lambda)$$

□

Corollary 4.7. *There exists a 2-approximation for ORF.*

Proof. Given an instance of ORF on a network N, we solve OSRF on the same network, yielding an optimal solution $(x, \bar{x}, \lambda)$ with objective value $x^* := \mathrm{val}_{\mathrm{OSRF}}(x, \bar{x}, \lambda)$. By Lemma 4.5, this can be done in polynomial time. By Lemma 4.6, $(x, \bar{x}, \lambda)$ is feasible for ORF with the same objective value.

Let $\text{opt}_{\text{ORF}}(N)$ be the optimal value of ORF on N. Assume for contradiction that $x^* < \frac{1}{2} \cdot \text{opt}_{\text{ORF}}(N)$. Let $(x', \bar{x}', \lambda')$ be an optimal solution to ORF. Then, again by Lemma 4.6, there exists $(x'', \bar{x}'', \lambda'')$ feasible to OSRF with $\text{val}_{\text{OSRF}}(x'', \bar{x}'', \lambda'') \geq \frac{1}{2} \cdot \text{opt}_{\text{ORF}}(N)$. Hence $x^* < \frac{1}{2} \cdot \text{opt}_{\text{ORF}}(N) \leq \text{val}_{\text{OSRF}}(x'', \bar{x}'', \lambda'')$, a contradiction to the optimality of x^*.

Therefore, $(x, \bar{x}, \lambda)$ is a feasible solution to ORF with value at least half of the optimum. □

4.5 Comparing optimal values

This section compares the optimal values of optional reroutable flows with various problems we have encountered so far.

In the first part, we compare optional reroutable flows to robust flows. These problems qualify for such a comparison, since in both cases any feasible flow on the network is a valid input with an objective value. For example in the case of non-optional reroutable flows, there exist flows which are not permissable and have no clearly defined value, making comparisons hard. We present some results for this case in the second part. Finally, we state the relation between the optional and non-optional versions of reroutable flows.

4.5.1 Comparing Robust Flows and Optional Reroutable Flows

Since we only defined optional reroutable flows for the failure of one arc, we assume $k = 1$ throughout this section for the number of failing arcs in the robust flow problem.

Let us first take a look at the value functions of the involved problems. The robust value for $k = 1$ is defined by

$$\text{val}_r(x) = \sum_{P \in \mathcal{P}} x(P) - \max_{\bar{e} \in E} x(\bar{e}),$$

whereas the optional strictly reroutable value amounts to

$$\text{val}_{\text{OSRF}}(x) = \sum_{P \in \mathcal{P}} x(P) \; - \; \max_{\bar{e} \in E} \left(x(\bar{e}) - \max_{x_{\bar{e}} \in \mathcal{F}_{\text{strict}}(x, \bar{e})} \text{val}(x_{\bar{e}}) \right).$$

Finally, the optional reroutable value is

$$\text{val}_{\text{ORF}}(x) = \sum_{P \in \mathcal{P}} x(P) \; - \; \max_{\bar{e} \in E} \left(x(\bar{e}) - \max_{x_{\bar{e}} \in \mathcal{F}(x, \bar{e})} \text{val}(x_{\bar{e}}) \right).$$

It is easy to see that for any flow x, the robust value is always less than or equal to the optional reroutable values. By definition we also have $\mathcal{F}_{\text{strict}}(x, \bar{e}) \subset \mathcal{F}(x, \bar{e})$ for any failing arc $\bar{e}$ and thus the non-strict version is at least as large as the strict one.

Fact 4.8. *Let $N = (G, s, t, u)$ be a network, x a given feasible network flow in N. Then $val_r(x) \leq val_{OSRF}(x) \leq val_{ORF}(x)$.*

Recall that $\text{opt}_P(N)$ denotes the optimal value of problem P on the network N. As a direct consequence of the above fact, it holds that $\text{opt}_{\text{ROB}}(N) \leq \text{opt}_{\text{OSRF}}(N) \leq \text{opt}_{\text{ORF}}(N)$ for all networks N.

Both inequalities may be strict. Consider the network N_1 in Figure 7. The construction on the left-hand side is equivalent to the multi-arc construction introduced in section 3.5. It consists of M arcs with capacity $\frac{1}{M}$ each. The flow x sends 1 unit of flow from s to the middle node uniformly distributed over all left-hand side arcs and then sends $\frac{1}{2}$ units of flow along both right-hand side arcs. If M becomes large, the interdictor in the robust flow problem will always interdict one of the two right-hand side arcs in the network. This gives a ROB-value of $\text{val}_r(x) = \frac{1}{2}$.

Any other flow sends more value along one of the two right-hand side arcs, which then gets interdicted, resulting in a smaller value. Hence $\text{opt}_{\text{ROB}}(N_1) = \frac{1}{2}$.

In OSRF, rerouting is allowed and thus all flow interdicted on the right-hand side of the network can always be saved. Hence the interdictor always chooses one of the arcs on the left-hand side. But when M gets large, the flow on any of these arcs gets arbitrarily small and the OSRF-value is $1 - \frac{1}{M}$. Hence $\text{opt}_{\text{ROB}}(N_1) < \text{opt}_{\text{OSRF}}(N_1)$.

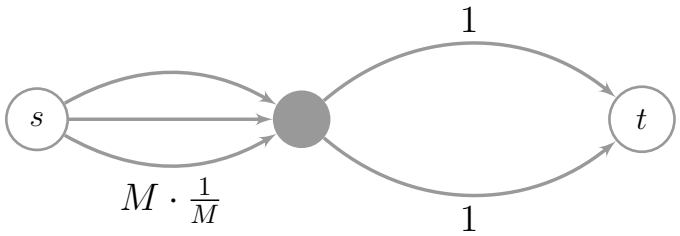

Figure 7: Network N_1; the optimal robust value is $\frac{1}{2}$, the optimal (optional) (strictly) reroutable value is $1 - \frac{1}{M}$

For the strictness of the second inequality, consider the network N_2 in Figure 8. The flow sending 1 unit along the center path directly to t has optional reroutable value $1 - \frac{1}{M}$, since any interdiction (except at the left-hand side multi-arc construction) can be countered by rerouting. This flow is optimal, since the left-hand side multi-arc construction yields an upper bound of $1 - \frac{1}{M}$ for any optional reroutable flow in the network.

If we now consider the same flow in context of the strict version of our problem, we can see that rerouting after failure of the last arc of the path is no longer possible, since the capacities on the central path are not freed up after interdiction.

Generally, the network admits no optional strictly reroutable flow with value larger than $\frac{1}{2}$: Any flow sent over one of the smaller nodes can be interdicted entirely, since these nodes only have one outgoing arc. As we have seen above, no more than $\frac{1}{2}$ units can be sent along the center path, otherwise the flow is blocking its own rerouting paths.

To actually achieve a flow with value $\frac{1}{2}$, we can for example send $\frac{1}{2}$ units along the center path and then reroute, or send $\frac{1}{2}$ units along the top and bottom smaller nodes, losing one of the paths to interdiction in the process.

As a result, the optimal optional strictly reroutable value in N_2 is equal to $\frac{1}{2}$, which proves that the second inequality of Fact 4.8 may be strict. Note that this additionally proves that the 2-approximiation given by Corollary 4.7 is, in fact, tight.

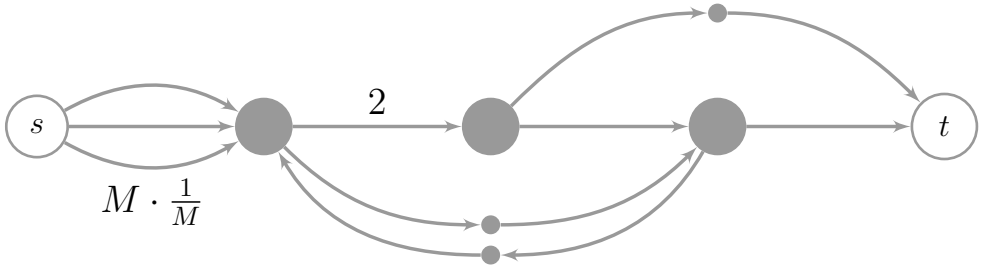

Figure 8: Network N_2; all unlabeled capacities are 1, the optimal optional reroutable value is $1 - \frac{1}{M}$, the optimal optional strictly reroutable value is $\frac{1}{2}$

4.5.2 Comparing Robust Flows and Reroutable Flows

Again, we assume $k = 1$ throughout this section. Opposed to before, we now consider rerouting to be non-optional. That means there may exist flows which are not feasible because there are arcs where flow cannot be rerouted after failure. We will see that this makes comparing the problem to robust flows impossible, at least w.r.t. objective values.

We recall the definitions of reroutable flows and strictly reroutable flows. A flow x is reroutable if there exists a rerouting for every failing arc $\bar{e}$ w.r.t. to capacities $\bar{u}_{x,\bar{e}}$ as defined in (4.1). Analogously, a flow x is strictly reroutable if there exists a rerouting for every failing arc $\bar{e}$ w.r.t. to capacities $\bar{u}_x$ as defined in (4.2).

Since $\bar{u}_x(e) \leq \bar{u}_{x,\bar{e}}(e)$ for all arcs $e \in E$, every strictly reroutable flow is reroutable and we can immediately infer the following fact.

Fact 4.9. *Let $N = (G, s, t, u)$ be a network. Then $opt_{SRF}(N) \leq opt_{RF}(N)$ for all networks N.*

This inequality can be strict for certain networks, confer for example Figure 1 of [MMO17].

Unfortunately, it turns out that robust flows and non-optional (strictly) reroutable flows cannot be compared w.r.t. the objective value.

Let us once again consider the example in Figure 7. Then the following flow x' is SRF-optimal: send $1 - \frac{1}{M}$ units of flow along the left-hand side arcs from s to the middle node, leaving only a single arc without any flow. Then send $1 - \frac{1}{M}$ units from the middle node to t along the top arc on the right-hand side. If any flow-bearing arc fails, there is always a possibility to reroute destroyed flow. Therefore, x' is

reroutable. It is SRF-optimal, since there exists a cut of capacity 1 and one arc with capacity $\frac{1}{M}$ always must remain without flow to allow for rerouting on the left-hand side. Hence, $\text{opt}_{\text{SRF}}(N_1) = 1 - \frac{1}{M}$.

As seen in the previous section, we have $\text{opt}_{\text{ROB}}(N_1) = \frac{1}{2}$, and hence $\text{opt}_{\text{ROB}}(N_1) < \text{opt}_{\text{SRF}}(N_1)$.

Now consider the network N_3 in Figure 9. The optimal robust value here is obviously 1, since the interdictor simply destroys one of the two paths.

In the network, two nodes exist that only have a single outgoing arc. Such an arc cannot carry any flow if the resulting flow is to be reroutable. Since there exists no path without such arcs in the network, the only reroutable flow is the zero-flow. Hence, $\text{opt}_{\text{SRF}}(N_3) = 0$ and $\text{opt}_{\text{ROB}}(N_3) > \text{opt}_{\text{SRF}}(N_3)$.

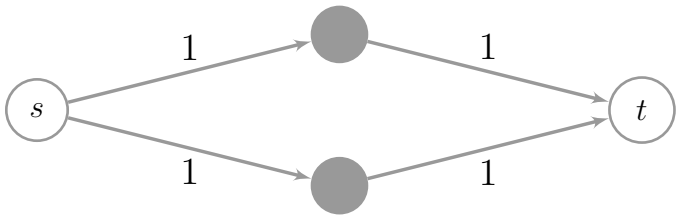

Figure 9: Network N_3; the optimal robust value is 1 and the optimal (strictly) reroutable value is 0

From the above examples we gather that there exist networks where the optimal robust value is larger than the optimal strictly reroutable value and the other way around. Therefore we cannot give any comparisons between the two problems.

4.5.3 Comparing Reroutable Flows and Optional Reroutable Flows

It is easy to see that making rerouting optional increases the objective value of the problems, since any reroutable flow is optionally reroutable with the same value. We state as much in the following fact:

Fact 4.10. *Let $N = (G, s, t, u)$ be a network. Then $opt_{RF}(N) \leq opt_{ORF}(N)$ and $opt_{SRF}(N) \leq opt_{OSRF}(N)$ for all networks N.*

This estimate is of course not always tight. Consider Network N_3 of Figure 9 again. The optimal strictly reroutable value is 0, the optimal robust value is 1 and by Fact 4.8, $\text{opt}_{\text{ROB}}(N_3) \leq \text{opt}_{\text{OSRF}}(N_3)$, hence $\text{opt}_{\text{SRF}}(N_3) < \text{opt}_{\text{OSRF}}(N_3)$. For the non-strict version, the same example works as well.

To receive an overview of all estimates in this section, please consult Figure 10.

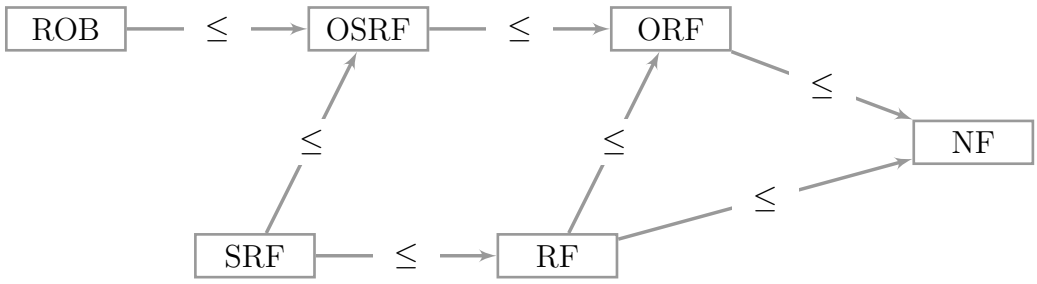

Figure 10: Overview of optimal value comparisons

4.6 Unit capacity networks

In unit capacity networks, all arcs e have the same capacity $u(e) = 1$. We will prove that, in this case, the optimal value of ORF on N is given by $\text{opt}_{\text{ORF}}(N) = \min_{C \in \mathcal{C}} \text{cap}(C) - 1$, where $\mathcal{C}$ is the set of all cuts in N. An optimal flow is given by any maximum network flow. The only exception to this result is the case where N is trivial. But then the zero-flow obviously is the only optimal flow.

All findings in this section are presented for ORF, but the same always holds for OSRF, too. In consequence of the above result, ORF and OSRF are solvable in polynomial time on unit capacity networks.

The following lemma already ensures that all flow values of ORF are smaller than or equal to the given bound. If we additionally find a flow with that exact value, we are done.

Lemma 4.11. *Let $N = (G, s, t, u)$ be a non-trivial unit capacity network. Then $opt_{ORF}(N) \leq \min_{C \in \mathcal{C}} cap(C) - 1$*

Proof. Let C be a minimum cut in N, x a maximum optional reroutable flow. Assume x gets interdicted on an arc in C. Any rerouting happening after destruction of that arc must also go through the cut C. Therefore, even including rerouting, only at most $\text{cap}(C) - 1$ units of flow can be sent through C. This gives $\text{val}_{\text{ORF}}(x) \leq \text{cap}(C) - 1$.

Now assume x is interdicted on some arc not in C. Because of the optimality of the interdiction value, the reroutable value of x must be smaller than or equal to the case where x was interdicted on C. But then, as we have just seen, the value is at most $\text{cap}(C) - 1$. Therefore $\text{val}_{\text{ORF}}(x) \leq \text{cap}(C) - 1$.

In both cases, we have $\text{val}_{\text{ORF}}(x) \leq \text{cap}(C) - 1$ and thus because of the optimality of x and C:

$$\text{opt}_{\text{ORF}}(N) \leq \min_{C \in \mathcal{C}} \text{cap}(C) - 1$$

□

Now consider again a minimal cut C. By the max-flow-min-cut Theorem 2.3, there exists a maximum flow x, s.th. $\text{val}(x) = \text{cap}(C)$. Only at most one arc with capacity 1 is interdicted (and possibly rerouted), hence $\text{val}_{\text{ORF}}(x) \geq \text{cap}(C) - 1$. But combining this with Lemma 4.11, we get $\text{val}_{\text{ORF}}(x) = \text{opt}_{\text{ORF}}(x) = \text{cap}(C) - 1$.

Corollary 4.12. *Let $N = (G, s, t, u)$ be a non-trivial unit capacity network. Then $opt_{ORF}(N) = \min_{C \in \mathcal{C}} cap(C) - 1$ and an optimal optional reroutable flow is given by any maximum network flow.*

It is interesting to note that this bound also holds for ordinary robust flows with single arc failure. As we have seen in section 4.5, it always holds that $\text{opt}_{\text{ROB}}(N) \leq \text{opt}_{\text{ORF}}(N)$. Additionally, any maximum network flow with optional reroutable value $\min_{C \in \mathcal{C}} \text{cap}(C) - 1$ also has a robust value of $\min_{C \in \mathcal{C}} \text{cap}(C) - 1$. Combining this yields $\text{opt}_{\text{ROB}}(N) = \min_{C \in \mathcal{C}} \text{cap}(C) - 1$.

The result can be extended to the case where multiple arcs fail in either ROB or ORF. Multiple arc failures for ORF are defined analogously to chapter 5.3 in the extended version of [MMO17]. In this case we get an optimal value of

$$\text{opt}_{\text{ROB}}(N) = \text{opt}_{\text{ORF}}(N) = \max\{\min_{C \in \mathcal{C}} \text{cap}(C) - k, 0\},$$

where k is the number of failures.

4.7 A hardness reduction

We have already seen some basic polynomial reductions throughout this thesis. For example robust flows can be reduced to their scaled version with unbounded c by simply setting $c = 1$. In the previous section we have seen that the optimal solution of O(S)RF on unit capacity networks always is a robust flow. This gives a very simple reduction from O(S)RF to ROB, where we just have to solve ROB to immediately receive a solution for O(S)RF.

In this section, we want to present a reduction from ROB to O(S)RF on general networks, which works by slightly adapting the underlying graph.

We assume $k = 1$ for all instances of occuring robust flows. We again present this result for the non-strict version of reroutable flows, but note that everything also works for OSRF.

Theorem 4.13. *There exists a polynomial reduction from ROB with $k = 1$ to ORF.*

Proof. All values occuring in this proof are defined on the networks of their respective flows. Let $N = (G, s, t, u)$ be an instance of robust flow problem for $k = 1$. We construct an new instance $N' = (G', s, t, u')$ for ORF in the following way.

Let $G' = (V', E')$ be the newly constructed graph. $V' = V \cup \{v_e : e \in E\}$ consists of all old nodes plus one node for every arc in G. For all arcs $e = (u, w) \in E$, we define $e_l = (u, v_e)$, $e_r = (v_e, w)$ and thus $E' = \bigcup_{e \in E} \{e_l, e_r\}$. Finally, we set $u'(e_l) = u'(e_r) = u(e)$ for all $e \in E$.

In short, N' describes the network N where all arcs have been doubled artificially and the copies have the same capacities. This construction can be used to prove the reduction. In accordance with Definition 2.19 and the definition of threshold languages, we prove for all $l \in \mathbb{N}$: $\text{opt}_{\text{ROB}}(N) \geq l \iff \text{opt}_{\text{ORF}}(N') \geq l$.

For the forward implication, assume there exists a robust flow x in N with value at least l. Then the flow x' constructed in the natural way by extending x over all doubled arcs in N' is feasible, since all arcs have copied capacities. Because of the doubled arcs construction we cannot reroute any flow, since the interdictor will always choose to destroy the right copy e_r of an arc e and we cannot reroute starting at node v_e if e_r fails. Therefore, the optional reroutable flow behaves like an ordinary robust flow and we get $\text{val}_{\text{ORF}}(x') = \text{val}_r(x') = \text{val}_r(x) \geq l$.

The reverse implication is very similar. Let x' be a flow in N' with $\text{val}_{\text{ORF}}(x') \geq l$. Then x constructed on N by shortcutting the doubled arcs is again feasible with the same value. There was no rerouting possible in x' because of the double arcs construction, so x' behaves exactly like a robust flow. Altogether $\text{val}_r(x) = \text{val}_r(x') = \text{val}_{\text{ORF}}(x') \geq l$.

The reduction from N to N' is polynomial, since the new number of nodes is $|V'| = |V| + |E|$ and the new number of arcs is $|E'| = 2|E|$. □

It is interesting to note that, since OSRF is polynomially solvable, the above theorem in the version for OSRF yields another proof for the polynomiality of ROB with $k = 1$.

Again, the result can be extended to multiple arc failures as defined in [MMO17, Chapter 5.3]. The proof is almost completely identical, because there is still no rerouting possible, even when k arcs fail. Therefore there exists a polynomial reduction from ROB with k failures to O(S)RF with k failures.

5 Conclusion

Aim of this Master's thesis was to introduce and evaluate variations of two typical problems in the framework of robust and reroutable flows. For that, we examined a variation of Maximum Robust Flow (ROB) called Maximum Scaled Robust Flow (ROB_c), where the amount of interdicted flow was scaled linearly before being subtracted from the objective function.

By Lemma 3.6, ROB_c can be solved in polynomial time via a shortest path computation if a single arc fails. It is NP-hard if the number of failing arcs is unknown.

We have proven in Theorem 3.7 and Lemma 3.11 that all solutions of ROB_c are ordinary maximum network flows if the scaling factor has a small absolute value. Theorem 3.14 establishes optimality of the zero-flow if c is larger than a factor depending on the number of interdicted arcs k and the s-t-path-connectivity $\pi(G)$. In section 3.4.5, the results on solution characterizations are summarized.

Tightness examples could be found for almost all bounds, but in the case of Theorem 3.7, there was a gap between the bound and the given counter-example value.

Lastly, we have proven polynomial solvability for affine linearly scaled robust flows if $k = 1$ and for arc-based scaled robust flows if $k \in \mathbb{N}$ is fixed.

We also defined a variation of Maximum Reroutable Flow (RF) called Maximum Optional Reroutable Flow (ORF) including its strict version Maximal Optional Strictly Reroutable Flow (OSRF). Contrary to the original problem, rerouting was non-mandatory in these variations. In Theorem 4.5, we have proven polynomiality for the strict version, again based on a shortest path computation. Corollary 4.7 presents a 2-approximation for ORF.

We compared O(S)RF and (S)RF to ROB with respect to objective values and hardness. It turns out that while ROB and (S)RF are incomparable, the optimal value of ROB is always smaller than that of O(S)RF, a direct consequence of Fact 4.8. We also showed that making rerouting optional can only increase the objective value. A summary of the results on the objective values of all problems can be found in Figure 10.

For unit capacity networks, Corollary 4.12 concludes that an optimal solution is always given by a maximum network flow and has a fixed value depending on the size of a smallest cut in the network.

Finally, in Theorem 4.13, we have proven a polynomial reduction from ROB with $k = 1$ to O(S)RF. The reduction illustrates the similarity of the two problems.

Overall, the variations introduced in this thesis present simple and valid alterations of pre-existing problems to encompass more difficult or general situations. Very interesting in particular is the notion of scaled robust flows for small values of c as

defined in Theorem 3.7, since the solution is guaranteed to be an ordinary network flow while still retaining most properties of robust flows.

All hardness results from [DM17] could be adapted to the scaled version of robust flows. Unfortunately, in the case of optional reroutable flows, some hardness results could not be reproduced from [MMO17], since the problem was different enough to prevent the given approaches.

Some further work is required in closing the gap between values for which Theorem 3.7 holds and values for which it does not hold. Adapting the approximation algorithm given by [BNO16] for robust flows to the scaled version was not successful because the scaling prevents one fundamental argument in the proof from working. It is possible some kind of similar bound can be accomplished with a slightly different approach. Additionally, the question of hardness for both ROB and ROB_c remains open for values of $k \geq 2$.

As mentioned above, NP-hardness of ORF could not be established based on the ideas which worked for the original problem RF. Similarly, a combinatorial bound using some variation of network cuts as in [MMO17] could not be found. Some more work can also be invested into analyzing variations of (S)RF where more than a single arc can fail and must be rerouted.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, 2009.

[ACN01] Yash P. Aneja, R. Chandrasekaran, and K. P. K. Nair. Maximizing residual flow under an arc destruction. *Networks*, 38(4):194–198, 2001.

[AMO89] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. Chapter iv network flows. In *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, pages 211 – 369. Elsevier, 1989.

[BNO16] Dimitris Bertsimas, Ebrahim Nasrabadi, and James B. Orlin. On the power of randomization in network interdiction. *Operations Research Letters*, 44(1):114–120, 2016.

[BNS13] Dimitris Bertsimas, Ebrahim Nasrabadi, and Sebastian Stiller. Robust and adaptive network flows. *Operations Research*, 61(5):1218–1242, 2013.

[BOS01] G. Brightwell, G. Oriolo, and F. B. Shepherd. Reserving resilient capacity in a network. *SIAM Journal on Discrete Mathematics*, 14(4):524–539, 2001.

[BS03] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1):49–71, Sep 2003.

[CGK+02] Chandra Chekuri, Anupam Gupta, Amit Kumar, Joseph (Seffi) Naor, and Danny Raz. Building edge-failure resilient networks. In William J. Cook and Andreas S. Schulz, editors, *Integer Programming and Combinatorial Optimization*, pages 439–456, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 3rd edition, 2009.

[CZ17] Stephen R. Chestnut and Rico Zenklusen. Hardness and approximation for network flow interdiction. *Networks*, 69(4):378–387, 2017.

[Dan55] George B. Dantzig. Linear programming under uncertainty. In *Management Science*, volume 1, pages 197–206. The Institute of Management Sciences, 1955.

[DC07] Donglei Du and R. Chandrasekaran. The maximum residual flow problem: NP-hardness with two-arc destruction. *Networks*, 50(3):181–182, 2007.

[Die16] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, 5th edition, 2016.

[Dij59] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[DM17] Yann Disser and Jannik Matuschke. The complexity of computing a robust flow. *CoRR*, abs/1704.08241, 2017.

[EK72] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[FF56] Lester R. Ford and Delbert R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[GKL+17] Corinna Gottschalk, Arie M. C. A. Koster, Frauke Liers, Britta Peis, Daniel Schmand, and Andreas Wierz. Robust flows over time: models and complexity results. *Mathematical Programming*, Jun 2017.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, 1988.

[GMT14] Virginie Gabrel, Cécile Murat, and Aurélie Thiele. Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3):471 – 483, 2014.

[GTT89] Andrew V. Goldberg, Éva Tardos, and Roberte Tarjan. Network flow algorithms. Technical report, Cornell University Operations Research and Industrial Engineering, 1989.

[Kel14] Lucia Keller. *Complexity of Optimization Problems: Advice and Approximation*. PhD thesis, ETH Zurich, 2014.

[Kha79] Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244(5):1093–1096, 1979.

[McC96] S. Thomas McCormick. A polynomial algorithm for abstract maximum flow. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 490–497, 1996.

[MMO17] Jannik Matuschke, S. Thomas McCormick, and Gianpaolo Oriolo. Rerouting flows when links fail. In *44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *Leibniz International Proceedings in Informatics*, pages 89:1–89:13, 2017.

[Tre11] Luca Trevisan. Combinatorial optimization: exact and approximate algorithms. *Stanford University*, 2011.

[UU12] Michael Ulbrich and Stefan Ulbrich. *Nichtlineare Optimierung*. Springer-Verlag, 2012.