Freie Universität Berlin

Fachbereich Mathematik und Informatik
Institut für Mathematik

Combinatorial Models of Compressor Stations in Gas Networks

Bachelorarbeit im Studiengang Mathematik

René Saitenmacher

Berlin, den 09.05.2016

Betreut von
Prof. Dr. Ralf Borndörfer
Dr. Benjamin Hiller

Ich erkläre hiermit, dass ich die vorliegende Bachelorarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die Arbeit is frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlich Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Berlin, den 09. Mai 2016

Danksagungen

An dieser Stelle möchte ich mich bei all denen bedanken, die mir diese Bachelorarbeit ermöglicht und mich dabei unterstützt haben.

Mein besonderer Dank gilt Prof. Dr. Ralf Borndörfer und Dr. Benjamin Hiller, die es mir ermöglicht haben, mich in ein interessantes Thema einzuarbeiten und meine Bachelorarbeit am Zuse Institute Berlin zu verfassen. Ebenso möchte ich mich besonders bei Tom Walther für die viele Unterstützung bedanken. Desweiteren geht mein Dank natürlich an alle anderen studentischen Hilfskräfte, die mir oft bei kleinen Problemen weitergeholfen haben, darunter insbesondere Tobias Buchwald, Gerwin Gamrath und Fabian Mett.

Das wichtigste und größte Dankeschön geht aber an meine Eltern und meine Großeltern, die während des gesamten Studiums eine große Unterstützung für mich waren und auf die ich mich jederzeit verlassen konnte. Nicht zuletzt möchte ich mich bei ihnen dafür bedanken, dass sie mir dieses Studium erst ermöglicht haben.

Abschließend möchte ich mich noch bei meiner Freundin France-Audrey bedanken, die während der ganzen Zeit stets an meiner Seite war und mich sehr unterstützt hat.

Contents

List of Algorithms

List of Frameworks

List of Figures

List of Tables

1 Introduction

Natural gas is one of the most important energy sources in Germany. In 2015, its share of 21% of Germany's primary energy consumption was second only to petroleum. Of particular significance is its use as a heat source. In 2015, 40% of German private households used natural gas heating systems, making it the most important energy source on this market. Aside from that, it is a widely used industrial heat and energy source and is predicted to play an increasingly important role as a component of motor fuel. In comparison to other natural resources, such as petroleum or coal, natural gas is valued as a relatively climate friendly and inexpensive energy source with a high degree of efficiency. It is also worth to be noted that despite natural gas being a limited, natural resource the importance of gas as an energy source is not expected to decrease anytime soon. On the one hand, the access to natural gas resources is expected to increase due to technological advancement. On the other hand, gas in general synergizes well with renewable energy sources for many reasons. First, it serves as a backup when there is a shortage due to some other energy source being temporarily unavailable, e.g. solar power at night. Second, biogas is a renewable energy source. And last but not least, gas might be used as a form of storage for energy that has been produced from other, only temporarily available sources, such as solar or wind power. For all these reasons, there exists an extensive gas network in Germany which currently spans more than 500.000 km. It is essential for the import of natural gas of which Germany receives more than 90% from foreign countries through gas pipelines. Furthermore, it is needed for the transport of gas within and also through Germany which due to its geographic location in Central Europe serves as an important transit country for natural gas.[1]

Overview: Our main interest lies in the analysis and simplification of combinatorial models of compressor stations. In Chapter 1, we shall give an overview of the mathematical background of our work which includes flow networks, linear programming and a brief examination of related work. Chapter 2 will be dedicated to a short description of compressor stations and an explanation of how they are modelled in the language of mathematics. Following this, we shall introduce the main problem that we will investigate in Chapter 3 where we will also address our motivation to work on it as well as a brief overview of our approach to solve it. Then, in Chapter 4 and Chapter 5, we shall explain and discuss in detail the approach we took and the methods we used. Chapter 6 will be devoted to the computational implementation of our approach and

[1] Statistics and information according to [1]

to presenting the results achieved with it.

1.1 Flow Networks

Due to their use in gas network modelling, flow networks are at the very foundation of this thesis. As definitions may vary throughout literature and since we require some specific properties to hold for our models, we will start by giving our own definitions as follows:

Definition 1.1. Directed Multigraph
A directed multigraph G consists of a set of nodes N, a set of arcs A and mappings $t : A \to N$ and $h : A \to N$ which assign a tail and head node, respectively, to an arc. No self-loops are allowed, meaning $\forall a \in A : t(a) \neq h(a)$. Often, we shall write that $G = (N, A)$, omitting the direct reference to t and h if that reference is clear from the context.

It shall be noted that whenever we use the term graph we shall refer to a directed multigraph as defined above. Additionally, we shall use the following notation for directed multigraphs:

$$\forall n \in N : \text{set of in-arcs } A^{in}(n) = \{a \in A : h(a) = n\} \tag{1.1}$$

$$\forall n \in N : \text{set of out-arcs } A^{out}(n) = \{a \in A : t(a) = n\} \tag{1.2}$$

$$\forall n \in N : \text{degree } \delta(n) = |A^{in}(n)| + |A^{out}(n)| \tag{1.3}$$

Definition 1.2. Flow Network
A flow network $\Gamma = (G, \underline{f}, \overline{f})$ consists of a directed multigraph $G = (N, A)$ together with functions $\underline{f} : N \cup A \to \mathbb{R} \cup [-\infty, +\infty]$ and $\overline{f} : N \cup A \to \mathbb{R} \cup [-\infty, +\infty]$ which we refer to as lower and upper flow bounds, respectively. Its set of admissible flows $\mathcal{F}(\Gamma)$ is the set of all functions $f : N \cup A \to \mathbb{R}$ such that:

$$\forall e \in (N \cup A) : \underline{f}(e) \leq f(e) \leq \overline{f}(e) \tag{1.4}$$

$$\forall n \in N : f(n) = \sum_{a \in A^{out}(n)} f(a) - \sum_{a \in A^{in}(n)} f(a) \tag{1.5}$$

Note that we allow arc flow to be negative. For the purpose of our modelling, the signum of arc flow describes its direction with respect to the orientation of the arc. This means that for an arc $a \in A$ and $x \in \mathbb{R}^+$, a flow f with $f(a) = x$ effectively models a flow of x from $t(a)$ to $h(a)$ on a while a flow f' with $f'(a) = -x$ effectively models a flow of x from $h(a)$ to $t(a)$ on a.

Similarly, this applies to the nodes. We do not include any notion of sinks or sources in our definition as nodes may function as either of them, depending on the signum of their flow. For the purpose of our modelling, for a node $n \in N$ and $x \in \mathbb{R}^+$, a flow f with $f(n) = x$ effectively models a flow of x at n into the network while a flow f' with $f'(n) = -x$ effectively models a flow of x at n out of the network. However, in our models of compressor stations we will distinguish between two types of nodes, boundary nodes and innodes, where the first admit non-zero flow on themselves while the latter do not. This will be detailed further in the second chapter.

Definition 1.3. Pressure
Consider a flow network Γ together with functions $\underline{p} : N \to \mathbb{R}_0^+$ and $\overline{p} : N \to \mathbb{R}_0^+$ which we refer to as lower and upper pressure bound, respectively. Then, its set of admissible pressures $\mathcal{P}(\Gamma)$ is the set of all functions $p : N \to \mathbb{R}_0^+$ such that:

$$\forall n \in N : \underline{p}(n) \leq p(n) \leq \overline{p}(n) \tag{1.6}$$

$$\forall a \in A : c_a(p(t(a)), p(h(a))) \leq 0 \tag{1.7}$$

where $c_a : \mathbb{R}^2 \to \mathbb{R}^k$ and (1.7) denotes some arc-dependent pressure constraints.

This slightly imprecise seeming definition will become a lot clearer in the next chapter where we will discuss in detail the exact constraints on pressure which arise in gas network modelling.

1.2 Linear Programming

In the course of our investigation, we will employ methods of linear optimization and state linear programs. Therefore, we shall now give a brief introduction to this topic. For further reading, we recommend [11] and [4].

Definition 1.4. Linear Program
A linear program, or short LP, is an optimization problem consisting of the maximization or minimization of a linear function, called objective function, subject to linear constraints. In its most general form it is often stated as:

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{1.8}$$

where $x \in \mathbb{R}^n$ is to be determined, and $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$ are given.

Definition 1.5. Feasibility
Following the notation in the previous definition, $x \in \mathbb{R}^n$ is called a feasible solution of

the linear program if it satisfies the linear program's constraints. It is called optimal, if it is a maximal, feasible solution. If a linear program does not have any feasible solutions, it is called infeasible. If there exist feasible solutions, but no optimal solution, it is called unbounded.

We note that (1.8) has been historically established as a standard way to denote a generic, linear program, and all linear programs may be transformed into this representation. However, at times it can be preferable to choose a different, more meaningful or intuitive representation which relates better to the context within which a linear program arises. This may include, for example, (linear) function notation in place of matrix notation, explicitely stating multiple constraints, minimization of the objective function, the use of equality constraints or the lack of non-negativity constraints.

Linear programs have been extensively studied in the field of linear optimization for decades. Moreover, efficient solution techniques for linear programs have been developed and refined over the years. Most prominently, this includes the simplex algorithm (see, e.g., [8] and [7]) which traverses on the edges of the convex polytope defined by the constraints of a linear program to find extremal vertices.

As a result, there exists a range of computational solvers that can solve various optimization problems including linear programs. Among these are academic, free available solvers such as SCIP [20] as well as commercial solver such as CPLEX [17] or Gurobi [12]. For the purpose of our investigation and subsequent implementation of our approach, we shall make use of them to solve linear programs.

2 Compressor Stations

2.1 Role within Gas Networks

Compressor stations are at the heart of every gas network. While a gas network consists of numerous pipes that transport gas over great lengths, these pipes are mostly static elements and don't allow for regulation of the gas flow. Besides that, friction causes the rate of gas flow through pipes to decrease over distance and thus compressor stations are needed to keep up the gas flow. This is achieved by compressor machines which increase the pressure on the gas. As this compression consumes energy, it is in the interest of network infrastructure providers to optimize the performance of compressor stations. In addition to that, operable elements within compressor stations allow for routing of gas flow. Thus, they are essential for managing the gas flow and adjusting it to a changing demand.

In accordance with their function, a compressor station might lie between only two pipelines where one of them provides an inflow of gas into the compressor station and the other one receives an outflow of compressed gas from the compressor station. However, it might also have more than two boundary points with the surrounding gas network and operable elements in the compressor station might allow to adjust the direction of the flow through the compressor station. Therefore, the role of boundary points might change depending on the routing within the compressor station as well as on the situation of the surrounding gas network.

In general and from a high level of abstraction, compressor stations in gas networks can thus be viewed as (operable) points of intersection between pipeline systems. But, compressor stations themselves represent small gas networks with an often complex layout. Therefore, they are more accurately described as subnetworks of the large gas network that they are a part of. However, from now on we will consider compressor stations as self-contained entities and seperated from the rest of the gas network in order to analyze the network structure that is found inside of them. We shall start by giving an detailed overview of the layout modelling of compressor stations.

Figure 2.1: Aerial view of the compressor station Rothenstadt-Weiherhammer (Wikimedia Commons, 08.05.2016, [19])

2.2 Modelling of Compressor Stations

Definition 2.1. Compressor Station Graph
A compressor station graph is a graph $G = (N, A)$ with

$$N = N_b \sqcup N_i \tag{2.1}$$

$$A = A_{sc} \sqcup A_v \sqcup A_{cv} \sqcup A_{cg} \tag{2.2}$$

where N_b denotes the set of boundary nodes, N_i denotes the set of innodes, A_{sc} denotes the set of short cuts, A_v denotes the set of valves, A_{cv} denotes the set of control valves and A_{cg} denotes the set of compressor groups.

The starting point of our mathematical investigation are compressor station graphs as defined above which model the topology of compressor stations. Note, that these graphs are models and generally do not resemble the actual topology of compressor stations. These compressor station graphs have been provided to us by Open Grid Europe GmbH. In addition, we have been provided with all the necessary, technical data to define flow/pressure networks on them which align with the different operation modes of the respective compressor stations. A formal definition of this will be given in section 3.1. The key point is that the operation mode of a whole compressor station is given indirectly by a joint specification of operation modes for different elements that lie within the compressor station. This is the reason why our definition of a compressor station graph distinguishes between different types of nodes and arcs. They represent the different elements within an actual compressor station which all put certain constraints on flow and pressure. Some of these constraints that individual elements put on, depend on the respective elements' operation modes. Therefore, we

shall give a brief overview of the different arc and node types in order to prepare the mathematical definitions which will follow in section 3.1. For further reading on the technical and mathematical aspects of gas network modelling and optimization, we recommend [18]. For the most part, we will follow the terminology that is given there.

Boundary Node: Boundary nodes are the nodes at which the compressor station would be connected to the rest of the gas network, thus possibly allowing flow into or out of the compressor station. However, we view compressor stations as self-contained and without further knowledge of their surrounding network. Therefore, the flow on boundary nodes is generally assumed to possibly be unbounded:

$$\forall n \in N_b : -\infty = \underline{f}(n) < \overline{f}(n) = \infty \tag{2.3}$$

Innode: Innodes are all nodes which are not boundary nodes. In contrast to boundary nodes, they do not admit any flow into or out of the compressor station:

$$\forall n \in N_i : \underline{f}(n) = 0 = \overline{f}(n) \tag{2.4}$$

Short Cut: Short cuts are the most basic type of arcs. They represent very short pipes inside of a compressor station. Some of them might not exist in the actual compressor station, because they have been artifically introduced during the modelling process. As short cuts represent just plain, short pipes, we do not distinguish different operation modes for them. They generally admit flow in both directions, and they require the flow on both their endpoints to be identical:

$$\forall a \in A_{sc} : \underline{f}(a) \leq 0 \leq \overline{f}(a) \tag{2.5}$$

$$\forall a \in A_{sc} : p(t(a)) = p(h(a)) \tag{2.6}$$

Pipe: Pipes are usually used to transport gas over greater distances and are thus mostly found outside of compressor stations. Due to physical effects such as friction playing a bigger role over long distances, modelling of flow and pressure on pipes is a topic of research on its own (see, e.g., [16]). However, some pipes might also exist within compressor stations. This can, for example, stem from the fact that some compressor stations are divided into multiple buildings which are connected by pipes. Even though, most of these pipes are of negligble length, because compressor stations are somewhat geographically bounded. For our purposes, all pipes in the models have been replaced by short cuts.

Valve: Valves are operable network elements, which are used to route gas flow through compressor stations. There are two different operation modes for them: open and closed. If a valve is closed, it is physically blocked, preventing gas from passing through it. This also causes the pressure on both ends of the valve to be decoupled. Therefore,

closed valves may be considered as non-existent in flow network models of compressor stations. Open valves, in contrast, do admit flow. However, open valves are just plain, short pipes. As such, they may be replaced by short cuts in flow network models of compressor stations.

Control Valve and Compressor Group: Control valves and compressor groups are operable network elements, which are used to both route gas flow through compressor stations and adjust gas pressure. Control valves may decrease pressure, while compressor groups may increase pressure. For both of them, there exist three different operation modes: closed, bypass and active. Closed control valves and closed compressor groups, just like closed valves, are physically blocked, preventing gas from passing through them. This also causes the pressure on both their endpoints to be decoupled. Therefore, closed control valves as well as closed compressor groups may be considered as non-existent in flow network models of compressor stations. In contrast, control valves and compressor groups in bypass mode do admit gas flow, but they do not affect pressure.as bypass literally means that any gas flow going through them merely passes by. Therefore, control valves and compressor groups in bypass mode just function as plain, short pipes. As such, they may be replaced by short cuts in flow network models of compressor stations. The real difference comes in with active control valves and compressor groups. They have a fixed working direction, which aligns with their orientation as arcs in our models, and they create a pressure difference between their inlet and outlet. All this is described by the following constraint on active control valves and compressor groups in flow networks models of compressor stations:

$$\forall a \in (A_{cv} \cup A_{cg}) : 0 \leq \underline{f}(a) \leq \overline{f}(a) \tag{2.7}$$

$$\forall a \in (A_{cv} \cup A_{cg}) : \underline{p}_{in}(a) \leq p_{in}(a) \leq \overline{p}_{in}(a) \tag{2.8}$$

$$\forall a \in (A_{cv} \cup A_{cg}) : \underline{p}_{out}(a) \leq p_{out}(a) \leq \overline{p}_{out}(a) \tag{2.9}$$

$$\forall a \in (A_{cv} \cup A_{cg}) : \underline{\Delta}(a) \leq \Delta(a) \leq \overline{\Delta}(a) \tag{2.10}$$

where $\Delta : (A_{cv} \cup A_{cg}) \to \mathbb{R}$ denotes the pressure difference that they create, and $p_{in}(A_{cv} \cup A_{cg}) \to \mathbb{R}$ and $p_{out}(A_{cv} \cup A_{cg}) \to \mathbb{R}$ denote the pressure at their inlet and outlet, respectively. They are defined as follows:

$$\forall a \in (A_{cv} \cup A_{cg}) : p_{in}(a) := p(t(a)) - p_{in}^{loss}(a) \tag{2.11}$$

$$\forall a \in (A_{cv} \cup A_{cg}) : p_{out}(a) := p(h(a)) + p_{out}^{loss}(a) \tag{2.12}$$

$$\forall a \in (A_{cv} \cup A_{cg}) : \Delta(a) := p_{in}(a) - p_{out}(a) \tag{2.13}$$

where $p_{in}^{loss} : (A_{cv} \cup A_{cg}) \to \mathbb{R}$ and $p_{out}^{loss} : (A_{cv} \cup A_{cg}) \to \mathbb{R}$ denote some pressure loss that occurs at the inlet and outlet, respectively, of an active control valve or compressor group. As we can see, the only thing, that really distinguishes active control valves and active compressor groups in our model, is that the pressure difference is positive for the former, but negative for the latter. Clearly there exist more differences between actual control valves and compressor stations, but the given description is sufficient for our investigation.

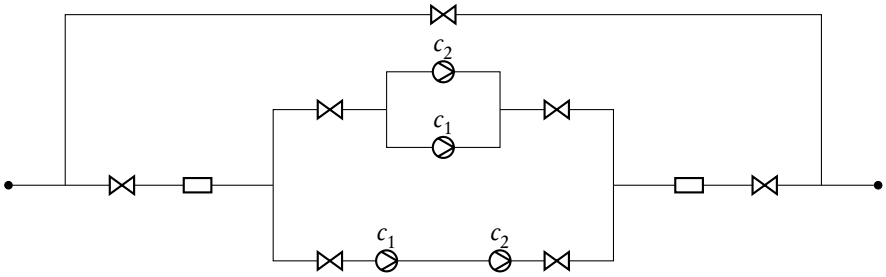
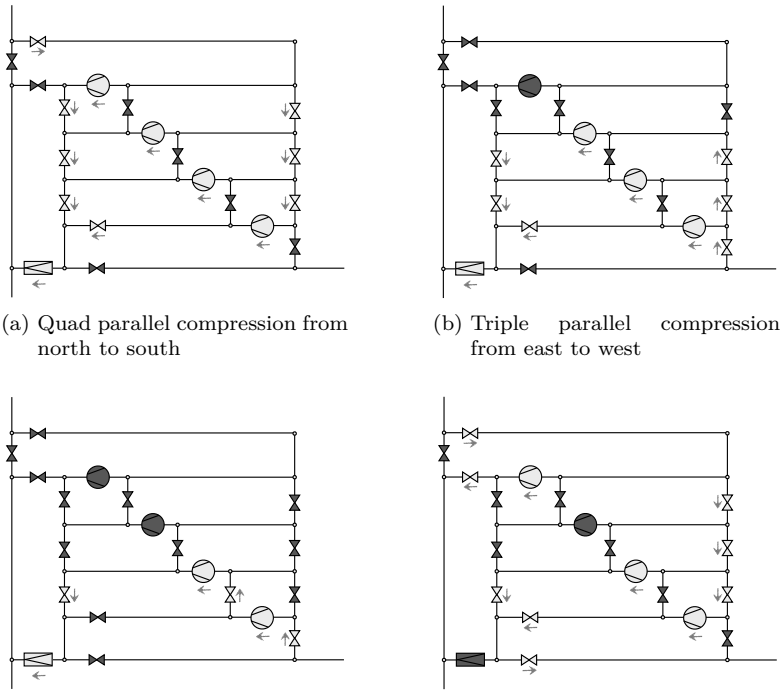


Figure 2.2: Our notion of an arc of type compressor group already contains a significant level of abstraction. This graph shows the underlying structure of a compressor group which corresponds to a single arc in our graph. Here, the gas flows from left to right and the different paths it can take correspond to the possibilites of routing gas flow through the compressor group. From top to bottom, these are bypass, double parallel compression by compressors c_1 and c_2 and double serial compression by compressors c_1 and c_2.

3 Analyzing Operation Modes of Compressor Stations



(a) Quad parallel compression from north to south

(b) Triple parallel compression from east to west

(c) Double serial compression from east to west

(d) Single compression from north to south and double parallel compression from north to east

Figure 3.1: The above technical drawings show four different operation modes of some compressor station. Dark symbols on arcs represent closed elements. Light symbols correspond to open valves, active control valves or active compressor stations. Arrows annotate the flow direction.

3.1 Decisions/Terminology

In section 2.2, we have introduced the notion of a compressor station graph which provides a topological model of a compressor station. We have also given a description of the elements inside of a compressor station, their operation modes and respective constraints on flow and pressure. This allows us to introduce flow network models that represent different operation modes of a compressor station. These operation modes are given as decisions where decisions are joint specifications of operation modes for elements inside of a compressor station. In the following, we shall define all the relevant terminology.

Definition 3.1. Decision Group
Let $G = (N, A)$ be a compressor station graph. Then, $D \subseteq (A \setminus A_{sc})$ is the decision group of any compressor station (sub-)graph $G' = (N', A')$, $G' \subseteq G$ with

$$N'_b = \{n \in N' | n \in N_b \text{ or } \delta_G(n) > \delta_{G'}(n)\} \tag{3.1}$$

$$N'_i = N' \setminus N'_b \tag{3.2}$$

$$\forall S \in \{A_{sc}, A_v, A_{cv}, A_{cg}\} : S' = S \cap A' \tag{3.3}$$

$$D = A' \setminus A'_{sc} \tag{3.4}$$

Corollary 3.2. *$D = A \setminus A_{sc}$ is the decision group of G.*

Corollary 3.3. *For any partition $D_1 \sqcup ... \sqcup D_k = D$, $k \in \mathbb{N}$, there exists an edge-disjoint partition $G_1 \sqcup ... \sqcup G_k = G$, where D_i is the decision group of G_i for all $i \in [k]$.*

Note, that $A \setminus A_{sc}$ exactly denotes the set of all operable elements inside of a compressor station which consists of valves, control valves and compressor groups. We also remark that a compressor station graph must not necessarily represent an actual compressor station as a whole. It may as well just model a part of it. The idea behind this is to partition large compressor stations into smaller parts that are easier to handle. These smaller parts bear the same characteristics as any compressor station and may as well be viewed as compressor stations on their own.

Definition 3.4. Decision
For a decision group $D = A \setminus A_{sc}$, we say that $d : D \to \{open, closed, bypass, active\}$ is a decision if

$$\forall a \in A_v : d(a) \in \{open, closed\} \tag{3.5}$$

$$\forall a \in (A_{cv} \cup A_{cg}) : d(a) \in \{closed, bypass, active\} \tag{3.6}$$

Definition 3.5. Decision's Network
Consider a compressor station graph G with decision group $D = A \setminus A_{sc}$ and a decision $d : D \to \{open, closed, bypass, active\}$. Let $G' = (N', A')$, $G' \subseteq G$ be as follows:

$$N'_b = N_b \tag{3.7}$$
$$N'_i = N_i \tag{3.8}$$
$$A'_{sc} = \{a \in D : d(a) \in \{open, bypass\}\} \cup A_{sc} \tag{3.9}$$
$$A'_{cv} = \{a \in D : d(a) = active\} \tag{3.10}$$
$$A'_{cg} = \{a \in D : d(a) = active\} \tag{3.11}$$
$$A'_v = \emptyset \tag{3.12}$$

There exists a mapping, provided by the network infrastructure provider, that maps G' onto a flow network $\Gamma = (G', \underline{f}, \overline{f})$ with pressure bounds $\underline{p}, \overline{p}$ and pressure constraints as detailed in (2.3), (2.4), (2.5), (2.6), (2.7), (2.8), (2.9) and (2.10). Γ is called the network of d, and we shall use Γ to represent d.

In other words, a decision is a joint specification of operation modes for all operable elements within a compressor station. Its network is the flow network that is given by it in accordance with the description of the elements in section 2.2.

Definition 3.6. Validity of a Decision
Let d be a decision with network Γ.

$$\text{d is valid} :\iff \mathcal{F}(\Gamma) \neq \emptyset \text{ and } \mathcal{P}(\Gamma) \neq \emptyset \tag{3.13}$$

A decision specifies an operation mode of a compressor station by a joint specification of operation modes for all operable elements in the station which all put individual constraints on flow and pressure. Clearly, it might happen that some of these constraints are conflicting. Therefore, the notion of the validity of a decision captures whether the decision is physically feasible.

Definition 3.7. Partial Decision
For a decision group $D = A \setminus A_{sc}$, we say that $d : D \to \{open, closed, bypass, active\}$ is a partial decision if

$$\forall a \in A_v : d(a) \in \{open, closed\} \tag{3.14}$$
$$\exists a \in (A_{cv} \cup A_{cg}) : d(a) = open \tag{3.15}$$

Let $O := \{a \in (A_{cv} \cup A_{cg}) : d(a) = open\}$. Then, d corresponds to the set of all decisions $d' : D \to \{open, closed, bypass, active\}$ such that

$$d'|_{D \setminus O} = d|_{D \setminus O} \tag{3.16}$$
$$\forall a \in O : d'(a) \in \{bypass, active\} \tag{3.17}$$

A partial decision is, so to say, a decision which is not fully specified, because it does not distinguish between bypass and active mode for control valves and compressor groups. The decisions it corresponds to are all possibilities to fully specify it. Note, that some of the decisions it corresponds to might be valid, while others might not. Needless to say, partial decisions are generally not desirable.

What is also important to note, is that the number of different decisions for a decision group is exponential in the size of the decision group. This leads to a very large number of ways a compressor station can be operated in. It is in the best interest of a network operator to identify the best possible among these. For this purpose, there exist predefined decision groups with associated graphs and lists of corresponding decisions. Depending on the complexity of a compressor station, some compressor stations might only comprise one decision group while others might consist of several. So far, all the existing decisions for these decision groups have been generated manually.

3.2 Problem Statement and Motivation

Apart from their use by network operators, decisions are valuable data for mathematical optimization of gas networks. In specific, the existing decision groups and decisions for some compressor station describe that station's operating range, i.e. what flow and pressure it admits. This knowledge permits the construction of more abstract models of compressor stations which then in turn allow the optimization of larger parts of the network. In this context, we shall be concerned with the preprocessing of decision data for other optimization purposes. This specifically means spotting and eliminating errors and redundancy in the data. It is of great importance, because redundancy in the decision data leads to more complex models for subsequent optimization problems which are then harder to solve. Based on this, we shall see that several problems arise when working with the existing, manually generated decisions:

- Manual generation is an error-prone process and small errors in complex data are hard to spot. Thus, some automated check is needed to ensure that the manually generated decisions are valid.

- There might exist redundancy in the decisions. The complexity of compressor stations often allows multiple ways to, say, route gas flow north to south through a compressor station without affecting its pressure. Thus, it is likely that manual generation of the decision data leads to some redundancy. However, it is in our interest to clean the decision data and not have multiple decisions with the same result.

- The current data format only specifies partial decisions which causes ambiguity. We want to factor them out and only retain valid, fully specified decisions.

In order to address these problems, we shall aim to accomplish the following:

- Given a decision, determine whether it is valid.
- Given multiple decisions, determine whether there exists redundancy between them.

3.3 Overview of Approach

In the problem statement we have already identified the two main challenges that we face in the preprocessing of the decision data: validity and redundancy. In the following we shall give an overview of how we will approach them.

In order to determine the validity of a network, we must know what effect the flow and pressure bounds, known for each element individually, have on other elements and the network as a whole. By propagating these bounds, we will receive tight bounds for all elements. Also, we will find contradictions between the bounds, if existent, and thus be able to determine the decision's validity. As means of propagation, we shall investigate algorithmic propagation as well as linear programming. We will treat bound propagation in detail in chapter 4.

In order to determine redundancy between networks, we shall employ network reduction through edge contraction and removal. Concretely, we will identify some operations on the network, which will produce equivalent, but simpler networks. These simplified networks then permit better comparison, allowing us to detect redundancy more easily. We will treat network reduction in detail in chapter 5.

It is to note that there exists some overlap between the above methods as the process of network reduction itself contains some, even though simple propagation. Furthermore, it is to add that network reduction shall happen before propagation. However, since the concept of propagation is so essential, we shall deal with it first before coming to network reduction.

Given multiple decisions, we can now apply the above outlined process of network reduction and propagation to the decisions' respective networks. This will leave us with the simplified and propagated networks of the valid decisions, which can then be tested for isomorphies between them to identify redundancy. We will briefly cover this last aspect in section 5.2.

3.4 Related Work/NETCAST

As far as we are aware, the specific problem that we are concerned with has not been widely considered before within the field of gas network optimization. Our approach to this problem draws inspiration from a network aggregation method that Borndörfer et al. [3] developed for the timetabling problem which arises in the field of railway optimization.

The timetabling problem consists of creating a conflict-free timetable which optimally utilizes railway infrastructure. A major challenge in this is that railway network models are designed to allow exact traffic simulation and therefore contain a high level of detail. This makes them simply too complex for efficient infrastructure planning. In order to address this problem, Borndörfer et al. developed a bottom-up approach that has been implemented in the NETCAST software tool [9] and which consists of an automatic network aggregation with regard to a given, fixed set of train routes and train types. The result of this transformation from a microscopic to a macroscopic model is a simplified network which is then suitable for planning and optimization [2]. Most important, during this transformation the relevant properties of the network are retained in such a way that schedules computed on the macroscopic network can be re-transformed to the microscopic level and allow a conflict-free simulation.

In general, the success of NETCAST has inspired our approach to employ network reduction, or in other words aggregation, in order to obtain simplified flow networks which retain the relevant flow and pressure characteristics. Yet, we carefully remark that even though the transfer of this approach implies some similarity between the problems, the timetabling problem and our problem may not be confused to be identical. This becomes especially evident when considering the actual aggregation steps that are taken and which cannot simply be transfered one to one as they are based on considerations specific to the respective subject matter. Also, an important step in the network aggregation by NETCAST is the discretization of time which influences the accuracy of the resulting, macrotized network. In contrast, time does not play a role in our model which permits an exact network aggregation.

4 Bound Propagation

In this chapter we shall investigate different ways to propagate flow and pressure bounds on a decision's network. Nonetheless, the setting throughout this chapter is as follows: Given a decision represented by a network, we aim to tighten the bounds on flow and pressure in order to find a network which is equivalent to the given network but better suited to represent the decision. Additionally, we aim to determine the decision's validity in that process.

Framework 1 Bound Propagation

Input:

A decision's flow network $\Gamma = (G, \underline{f}, \overline{f})$, $G = (N, A)$ with pressure bounds $\underline{p}$ and $\overline{p}$.

Output:

Generally, a decision's flow network $\Gamma' = (G, \underline{f}', \overline{f}')$ with pressure bounds $\underline{p}'$ and $\overline{p}'$ such that

$$\mathcal{F}(\Gamma') = \mathcal{F}(\Gamma) \tag{4.1}$$

$$\mathcal{P}(\Gamma') = \mathcal{P}(\Gamma) \tag{4.2}$$

$$\forall e \in (N \cup A) : \underline{f}(e) \leq \underline{f}'(e) \tag{4.3}$$

$$\forall e \in (N \cup A) : \overline{f}(e) \geq \overline{f}'(e) \tag{4.4}$$

$$\forall n \in N : \underline{p}(n) \leq \underline{p}'(n) \tag{4.5}$$

$$\forall n \in N : \overline{p}(n) \geq \overline{p}'(n) \tag{4.6}$$

However, in the case that $\mathcal{F}(\Gamma) = \emptyset$ or $\mathcal{P}(\Gamma) = \emptyset$, which would mean that the decision represented by the given network is invalid, we would ideally want our method to report invalidity instead.

In the case that $\mathcal{F}(\Gamma) \neq \emptyset$ and $\mathcal{P}(\Gamma) \neq \emptyset$, which would mean that the decision represented by the given network is valid, we would ideally want the new bounds to also be tight:

$$\forall e \in (N \cup A) : \exists f \in \mathcal{F}(\Gamma') : f(e) = \underline{f}'(e) \tag{4.7}$$

$$\forall e \in (N \cup A) : \exists f \in \mathcal{F}(\Gamma') : f(e) = \overline{f}'(e) \tag{4.8}$$

$$\forall n \in N : \exists p \in \mathcal{P}(\Gamma') : p(n) = \underline{p}'(n) \tag{4.9}$$

$$\forall n \in N : \exists p \in \mathcal{P}(\Gamma') : p(n) = \overline{p}'(n) \tag{4.10}$$

4.1 Locally Enforcing Flow Conservation

Following definition 1.2, there exist two types of constraint on the flow in our network: the minimum and maximum bound on the flow for every element, nodes and arcs alike, and the flow conservation constraint on every node, given by (1.5). Thus, it at first seems like a viable idea to tighten these constraints locally at every node. This idea is best explained by the following example below. We use square nodes to represent boundary nodes, circular nodes to represent innodes and plain arcs to represent short cuts.

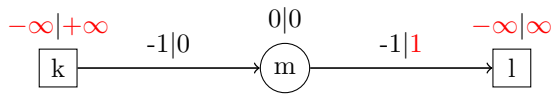


Figure 4.1: A decision's network with flow bounds annotated above arcs and nodes, where the red flow bounds are not tight.

In the above network, the flow bounds denoted in red are not tight meaning that there does not exist a valid flow which takes on these values. Formally, this follows from the flow conservation constraint on the nodes. However, it becomes even more immediately obvious when considering what is intuitively described by the flow conservation constraint: *The amount of flow that enters a node must equal the amount of flow which leaves that node.* Applying this understanding to node m, we see that the maximum amount of flow which can leave m towards l is 0, because the maximum amount of flow which can enter m any other way, namely from k, is 0. This gives the following, first improvement of the flow bounds:

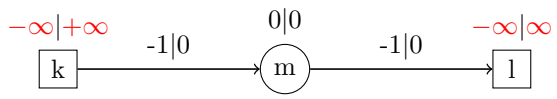


Figure 4.2: Now with an improved upper flow bound on the arc from m to l.

Also, it gives us an intuitive understanding of how the flow conservation constraint on a node puts constraints on arcs incident to the node: *The flow on an arc is constrained by the maximum amounts of flow that can possibly reach and leave the arc.* In other words, the amount of flow from m to l on the respective arc is limited by the amount of flow that can maximally be fed into the arc at m.

Applying similar reasoning based on the flow conservation constraint on k and l, respectively, we can tighten the remaining flow bounds in the network:

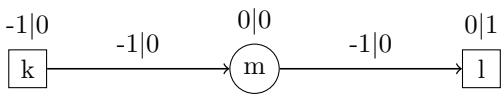


Figure 4.3: Now, the flow bounds on all elements are tight.

We may now formally denote our newly gained, intuitive understanding, starting with two alternative, arc-centered ways to rewrite the flow conservation constraint (1.5):

$$\forall n \in N : \forall a \in A^{in}(n) : f(a) = -f(n) + \sum_{a' \in A^{out}(n)} f(a') - \sum_{\substack{a' \in A^{in}(n) \\ a' \neq a}} f(a') \tag{4.11}$$

$$\forall n \in N : \forall a \in A^{out}(n) : f(a) = f(n) - \sum_{\substack{a' \in A^{out}(n) \\ a' \neq a}} f(a') + \sum_{a' \in A^{in}(n)} f(a') \tag{4.12}$$

This leads us to the following, new bounds on the flow on nodes and arcs, some of which might be tighter than the initial flow bounds $\underline{f}$ and $\overline{f}$:

$$\forall n \in N : \begin{cases} f(n) \leq \sum_{a \in A^{out}(n)} \overline{f}(a) - \sum_{a \in A^{in}(n)} \underline{f}(a) \\ f(n) \geq \sum_{a \in A^{out}(n)} \underline{f}(a) - \sum_{a \in A^{in}(n)} \overline{f}(a) \end{cases} \tag{4.13}$$

$$\forall n \in N : \forall a \in A^{in}(n) : \begin{cases} f(a) \leq -\underline{f}(n) + \sum_{a' \in A^{out}(n)} \overline{f}(a') - \sum_{\substack{a' \in A^{in}(n) \\ a' \neq a}} \underline{f}(a') \\ f(a) \geq -\overline{f}(n) + \sum_{a' \in A^{out}(n)} \underline{f}(a') - \sum_{\substack{a' \in A^{in}(n) \\ a' \neq a}} \overline{f}(a') \end{cases} \tag{4.14}$$

$$\forall n \in N : \forall a \in A^{out}(n) : \begin{cases} f(a) \leq \overline{f}(n) - \sum_{\substack{a' \in A^{out}(n) \\ a' \neq a}} \underline{f}(a') + \sum_{a' \in A^{in}(n)} \overline{f}(a') \\ f(a) \geq \underline{f}(n) - \sum_{\substack{a' \in A^{out}(n) \\ a' \neq a}} \overline{f}(a') + \sum_{a' \in A^{in}(n)} \underline{f}(a') \end{cases} \tag{4.15}$$

Algorithm 1 formally denotes the bound tightening procedure that we have employed in our example. First, we set $\underline{f}'$ equal to $\underline{f}$ and $\overline{f}'$ equal to $\overline{f}$. Then, at every node n we check whether we get tighter flow bounds on n or any arc incident to it by enforcing the flow conservation constraint and update the values of $\underline{f}'$ and $\overline{f}'$ accordingly. In addition, we shall maintain a queue of nodes to enforce the flow conservation constraint on. Since any arc a has two endpoints, it affects the constraint on two nodes, for $t(a)$ as an out-edge and for $h(a)$ as an in-edge. Therefore, successfully enforcing the flow conservation constraint on one of them might allow us reinforcing it on the other and we shall use the queue to keep track of this.

Algorithm 1 Locally Enforcing Flow Conservation

Input: $\Gamma = (G, \underline{f}, \overline{f})$

Output: $\Gamma' = (G, \underline{f}', \overline{f}')$ which satisfies (4.1), (4.3), (4.4) as stated in framework 1

$\overline{f}', \underline{f}' \leftarrow \overline{f}, \underline{f}$
Queue $\leftarrow$ enqueue all $n \in N$
while *Queue* not empty **do**
  $n \leftarrow$ dequeue from *Queue*
  if $\underline{f}'(n) < \sum_{a \in A^{out}(n)} \underline{f}'(a) - \sum_{a \in A^{in}(n)} \overline{f}'(a)$ **then** ▷ Tighten bounds on node
    $\underline{f}'(n) \leftarrow \sum_{a \in A^{out}(n)} \underline{f}'(a) - \sum_{a \in A^{in}(n)} \overline{f}'(a)$
  if $\overline{f}'(n) > \sum_{a \in A^{out}(n)} \overline{f}'(a) - \sum_{a \in A^{in}(n)} \underline{f}'(a)$ **then**
    $\overline{f}'(n) \leftarrow \sum_{a \in A^{out}(n)} \overline{f}'(a) - \sum_{a \in A^{in}(n)} \underline{f}'(a)$
  for $a \in A^{in}(n)$ **do** ▷ Tighten bounds on in-arcs
    if $\underline{f}'(a) < -\overline{f}'(n) + \sum_{a' \in A^{out}(n)} \underline{f}'(a') - \sum_{\substack{a' \in A^{in}(n) \\ a' \neq a}} \overline{f}'(a')$ **then**
      $\underline{f}'(a) \leftarrow -\overline{f}'(n) + \sum_{a' \in A^{out}(n)} \underline{f}'(a') - \sum_{\substack{a' \in A^{in}(n) \\ a' \neq a}} \overline{f}'(a')$
      Queue $\leftarrow$ enqueue $t(a)$
    if $\overline{f}'(a) > -\underline{f}'(n) + \sum_{a' \in A^{out}(n)} \overline{f}'(a') - \sum_{\substack{a' \in A^{in}(n) \\ a' \neq a}} \underline{f}'(a')$ **then**
      $\overline{f}'(a) \leftarrow -\underline{f}'(n) + \sum_{a' \in A^{out}(n)} \overline{f}'(a') - \sum_{\substack{a' \in A^{in}(n) \\ a' \neq a}} \underline{f}'(a')$
      Queue $\leftarrow$ enqueue $t(a)$
  for $a \in A^{out}(n)$ **do** ▷ Tighten bounds on out-arcs
    if $\underline{f}'(a) < \underline{f}'(n) - \sum_{\substack{a' \in A^{out}(n) \\ a' \neq a}} \overline{f}'(a') + \sum_{a' \in A^{in}(n)} \underline{f}'(a')$ **then**
      $\underline{f}'(a) \leftarrow \underline{f}'(n) - \sum_{\substack{a' \in A^{out}(n) \\ a' \neq a}} \overline{f}'(a') + \sum_{a' \in A^{in}(n)} \underline{f}'(a')$
      Queue $\leftarrow$ enqueue $h(a)$
    if $\overline{f}'(a) > \overline{f}'(n) - \sum_{\substack{a' \in A^{out}(n) \\ a' \neq a}} \underline{f}'(a') + \sum_{a' \in A^{in}(n)} \overline{f}'(a')$ **then**
      $\overline{f}'(a) \leftarrow \overline{f}'(n) - \sum_{\substack{a' \in A^{out}(n) \\ a' \neq a}} \underline{f}'(a') + \sum_{a' \in A^{in}(n)} \overline{f}'(a')$
      Queue $\leftarrow$ enqueue $h(a)$
return $\Gamma' := (G, \underline{f}', \overline{f}')$

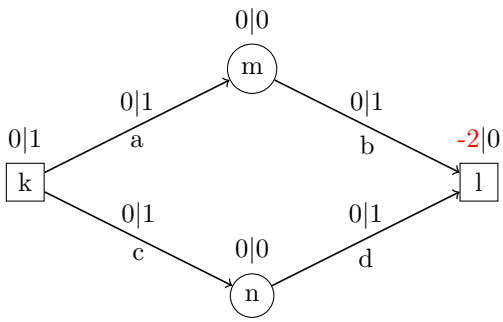


Figure 4.4: For this network, locally enforcing flow conservation fails to produce a tight lower flow bound on node l.

There are several observations that we can make about this algorithmic approach. First, it does not affect the set of admissible flows in any way. It does not render any flow inadmissible that was admissible before as we do not introduce new bounds, but merely rewrite bounds more explicitely that were already contained in our model. Also, it does not render any flow admissible that was inadmissible before as we are strictly tightening bounds in our algorithm. Second, it does not report invalidity for invalid decisions' networks, because it is not designed to report any invalidity at all. This can be overcome by always assuring that $\underline{f}(e) \leq \overline{f}(e)$ after tightening bounds for some $e \in (N \cup A)$ since otherwise there cannot exist an admissible flow due to (1.4). Third, algorithm 1 is unfortunately not guaranteed to produce tight flow bounds as we can see in figure 4.4.

In the network in figure 4.4, the lower flow bound on l is not tight. When looking at the network, it is intuitively clear that the amount of flow leaving the network at l cannot exceed the amount of flow going into the network at k. Correct substitution of flow conservation constraints yields

$$f(l) \geq -\overline{f}(k) = -1 \tag{4.16}$$

However, we fail to tighten this bound by locally enforcing the flow conservation constraint at l which merely gives us the following:

$$f(l) \geq -\overline{f}(b) - \overline{f}(d) = -2 \tag{4.17}$$

As we see, the flow bounds on b and d are tight. There exist admissible flows f_b and f_d for which $f_b(b) = 1$ and $f_d(d) = 1$. However, we fail to get a tight bound on the flow on l based on the tight bounds on b and d, because there does not exist an admissible flow f for which both $f(b) = 1$ and $f(d) = 1$. This means that by locally enforcing flow constraints at l we fail to see that the maximum amount of flow on b and d together is limited by the common source k and strictly less than the sum of the individual

maxima. As this cannot possibly be locally recognized at l, this example shows us the limitations of our attempt to tighten flow bounds locally.

In order to improve our approach, different methods seem possible. One way could be to propagate further constraints on the edges which are meant to model the dependencies of flow between different edges. Another way could be to recognize structures in the graph for which we know additional constraints, such as the diamond shape in figure 4.4 and the resulting constraint that $f(l) \geq -\overline{f}(k)$. However, there exists a much better and easier method for propagation flow bounds which we shall discuss in the next section.

4.2 Optimization-Based Flow Bound Tightening

In this section, we shall leverage our graph based flow bound propagation problem onto an algebraic level by modelling it as a linear program. This way, we do not have to be concerned with developing a propagation algorithm around structures found in the network. Instead, we will be able to make use of already existing and powerful methods from the field of linear optimization.

More precisely, we resort to Optimization Based Bound Tightening (OBBT), a technique which is used in the preprocessing of mixed-integer nonlinear programs (MINLPs), optimization problems which may include nonlinear or integer constraints. OBBT consists of solving LPs to obtain bounds on variables in a MINLP, and is known to be efficient, but computationally expensive due to the creation of additional optimization problems. For further reading on this topic, we recommend [10].

Using this approach, we will minimize and maximize the admissible flow on every element in the gas network seperately by solving the following, two LPs for every $e \in (N \cup A)$ where $\delta_{ee'}$ denotes the Kronecker delta:

$$\begin{aligned}
\text{maximize} \quad & \sum_{e' \in (N \cup A)} \delta_{ee'} f(e) \\
\text{subject to} \quad & f(e') \leq \overline{f}(e'), && \forall e' \in (N \cup A) \\
& f(e') \geq \underline{f}(e'), && \forall e' \in (N \cup A) \\
& f(n) = \sum_{a \in A^{out}(n)} f(a) - \sum_{a \in A^{in}(n)} f(a), && \forall n \in N
\end{aligned} \tag{4.18}$$

$$
\begin{aligned}
\text{minimize} \quad & \sum_{e' \in (N \cup A)} \delta_{ee'} f(e) \\
\text{subject to} \quad & f(e') \leq \overline{f}(e'), && \forall e' \in (N \cup A) \\
& f(e') \geq \underline{f}(e'), && \forall e' \in (N \cup A) \\
& f(n) = \sum_{a \in A^{out}(n)} f(a) - \sum_{a \in A^{in}(n)} f(a), && \forall n \in N
\end{aligned}
\tag{4.19}
$$

We note that an optimal solution of (4.18) yields a tight, upper bound on $f(e)$, which we shall denote by $\overline{f}_{opt}(e)$ and an optimal solution of (4.19) yields a tight, lower bound on $f(e)$, which we shall denote by $\underline{f}_{opt}(e)$. Moreover, we shall remark that these LPs cannot be unbounded. For all elements except boundary nodes, there exist finite flow bounds. For boundary nodes, the admissible flow is constrained to be finite by the flow conservation constraint (1.5) and the fact that any incident arcs have finite flow bounds. In addition, infeasibility of the LPs yields $\mathcal{F}(\Gamma) = \emptyset$, allowing us to identify decisions which are invalid due to the nonexistence of admissible flows.

This leads us to the following algorithm:

Algorithm 2 Optimization-Based Flow Bound Tightening

Input: A decision's flow network $\Gamma = (G, \underline{f}, \overline{f})$

Output: If $\mathcal{F}(\Gamma) \neq \emptyset$ is valid, a decision's flow network $\Gamma' = (G, \underline{f}', \overline{f}')$ that satisfies (4.1), (4.3), (4.4), (4.7), (4.8) as specified in framework 1. Otherwise, algorithm reports invalidity of the decision.

```
for e ∈ (N_b ∪ A) do                          ▷ We skip N_i due to (1.4) and (2.4)
    if (4.18) is feasible then
        f̄'(e) ← f̄_opt(e)                      ▷ LP gives tight bound
    else
        return error                          ▷ Decision infeasible
    if (4.19) is feasible then
        f̲'(e) ← f̲_opt(e)                      ▷ LP gives tight bound
    else
        return error                          ▷ Decision infeasible
return Γ' := (G, f̲', f̄')
```

We remark, that algorithm 2 has all the desirable characteristics that we specified in the bound propagation framework. It does not increase the set of admissible flows since we tighten existing bounds as specified in the first two constraints of (4.18) and (4.19), respectively. Also, it does not decrease the set of admissible flows which is guaranteed by the construction of our LPs that take minimum and maximum values over all admissible flows. In addition, algorithm 2 produces tight flow bounds on all

network elements, and is able to identify invalid decisions. This of course only for decisions which are invalid due to their set of admissible flows being empty. A decision with a non-empty set of admissible flows might still be invalid if it does not admit any pressure. This case will be dealt with during propagation of pressure bounds.

The tradeoff that we make with algorithm 2 is of course the computational expense that comes from solving $2*(|N_b|+|A|)$ linear programs. However, this can be well justified. First, algorithm 2 solves the flow bound propagation problem for real, existing compressor stations in reasonable time. Further details about our implementation can be found in chapter 6. Second, it is safe to assume that this problem does not scale infinitely due to geographical or operational constraints on the size of compressor stations. Third, our goal is to solve a one-time, data preprocessing problem. Therefore, runtime is not our biggest concern.

4.3 Optimization-Based Pressure Bound Tightening

Analogously, we may use the optimization-based approach from the previous section to tighten pressure bounds on our network which means minimizing and maximizing the admissible pressure on every node in the gas network seperately. This will be done subject to the pressure constraints on the various gas network elements as given in definition 1.3 and detailed in section 2.2. Thus, we shall formulate the following two LPs for every $n \in N$ where $\delta_{nn'}$ denotes the Kronecker delta:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{n' \in (N)} \delta_{nn'} p(n) \\
\text{subject to} \quad & p(n') \leq \overline{p}(n'), && \forall n' \in N \\
& p(n') \geq \underline{p}(n'), && \forall n' \in N \\
& p(t(a)) = p(h(a)), && \forall a \in A_{sc} \\
& p_{in}(a) \leq \overline{p}_{in}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& p_{in}(a) \geq \underline{p}_{in}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& p_{out}(a) \leq \overline{p}_{in}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& p_{out}(a) \geq \underline{p}_{in}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& \Delta(a) \leq \overline{\Delta}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& \Delta(a) \geq \underline{\Delta}(a), && \forall a \in (A_{cv} \cup A_{cg})
\end{aligned}
\tag{4.20}
$$

$$
\begin{aligned}
\text{minimize} \quad & \sum_{n' \in (N)} \delta_{nn'} p(n) \\
\text{subject to} \quad & p(n') \leq \overline{p}(n'), && \forall n' \in N \\
& p(n') \geq \underline{p}(n'), && \forall n' \in N \\
& p(t(a)) = p(h(a)), && \forall a \in A_{sc} \\
& p_{in}(a) \leq \overline{p}_{in}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& p_{in}(a) \geq \underline{p}_{in}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& p_{out}(a) \leq \overline{p}_{in}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& p_{out}(a) \geq \underline{p}_{in}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& \Delta(a) \leq \overline{\Delta}(a), && \forall a \in (A_{cv} \cup A_{cg}) \\
& \Delta(a) \geq \underline{\Delta}(a), && \forall a \in (A_{cv} \cup A_{cg})
\end{aligned}
\tag{4.21}
$$

We note that an optimal soultion of (4.20) yields a tight, upper bound on $p(n)$, which we shall denote by $\overline{p}_{opt}(n)$, and an optimal solution of (4.21) yields a tight, lower bound on $p(n)$, which we shall denote by $\underline{p}_{opt}$. Again, these LPs cannot be unbounded due to the existing pressure bounds on every node. Also, infeasibility of the LPs implies $\mathcal{P}(\Gamma) = \emptyset$, allowing us to identify decisions which are invalid due to the nonexistence of admissible flows.

This leads us to the following algorithm:

Algorithm 3 Optimization-Based Pressure Bound Tightening

Input: A decision's flow network Γ with pressure bounds $\underline{p}, \overline{p}$

Output: If $\mathcal{P}(\Gamma) \neq \emptyset$, a decision's flow network $\Gamma' = \Gamma$ with pressure bounds $\underline{p}'$, $\overline{p}'$ that satisfy (4.2), (4.5), (4.6), (4.9), (4.10) as specified in framework 1. Otherwise, algorithm reports invalidity of the decision.

```
for n ∈ (N) do
    if (4.20) is feasible then
        p̄'(n) ← p̄_opt(n)                    ▷ LP gives tight bound
    else
        return error                          ▷ Decision infeasible
    if (4.21) is feasible then
        p'(n) ← p_opt(n)                      ▷ LP gives tight bound
    else
        return error                          ▷ Decision infeasible
return p', p̄', Γ' := Γ
```

In accordance with our evaluation of algorithm 2 in the previous chapter, we shall

remark that algorithm 3 has all the desirable characteristics that we described in the bound propagation framework, specifically it produces tight bounds and identifies invalid decisions. Also, it comes at a tolerable, computational cost.

In addition, it is worthwile mentioning that it might be possible to also produce tight pressure bounds through local enforcement of pressure constraints. This did not work for flow bounds, but it might work for pressure bounds due to the different nature of the constraints. The use of such an approach could decrease the computational cost of pressure propagation significantly. However, there exists a reason why investigating this is not of great interest to us. It is that there actually exists a relation between flow and pressure in physical reality which has not yet been incorporated in the combinatorial models of compressor stations that our investigation relies on. Therefore, it is not worth focussing our efforts on separate pressure propagation.

5 Network Reduction

In this chapter we will describe the approach that we take with network reduction. The setting is as follows: Given a decision represented by its network, we shall remove and replace elements in the network's graph in a way that retains certain, important characteristics of the given network which are formally denoted in framework 2. The result of this process will be a new and simpler representative network for the decision. We shall also elaborate on the benefits of such a new and simpler representative.

Framework 2 Network Reduction

Input:

A decision's flow network $\Gamma = (G, \underline{f}, \overline{f})$, $G = (N, A)$ with pressure bounds $\underline{p}$ and $\overline{p}$.

Output:

A decision's flow network $\Gamma' = (G', \underline{f}', \overline{f}')$, $G' = (N', A')$ with pressure bounds $\underline{p}'$, $\overline{p}'$ such that:

$$N' \subseteq N \tag{5.1}$$
$$N'_b = N_b \tag{5.2}$$
$$|A'| \leq |A| \tag{5.3}$$
$$A'_{cv} = A_{cv} \tag{5.4}$$
$$A'_{cg} = A_{cg} \tag{5.5}$$
$$\forall f \in \mathcal{F}(\Gamma) : \exists f' \in \mathcal{F}(\Gamma') : f|_{N_b} = f'|_{N_b} \tag{5.6}$$
$$\forall p \in \mathcal{P}(\Gamma) : \exists p' \in \mathcal{P}(\Gamma') : p|_{N_b} = p'|_{N_b} \tag{5.7}$$
$$\forall f' \in \mathcal{F}(\Gamma') : \exists f \in \mathcal{F}(\Gamma) : f'|_{N_b} = f|_{N_b} \tag{5.8}$$
$$\forall p' \in \mathcal{P}(\Gamma') : \ \exists p \in \mathcal{P}(\Gamma) : p'|_{N_b} = p|_{N_b} \tag{5.9}$$

5.1 General Idea

There are two things that are of great importance to us in a decision's network: First, the values at the boundary nodes for each admissible flow and pressure. They provide a description of how the respective compressor station may be operated with regards

to the surrounding network. For example, whether it is possible to route some specified amount of flow with some specified pressure north to south through the compressor station under the given decision. Altogether, the knowledge of these values for all decisions of a decision group means knowledge about all ways gas flow can be routed through the respective compressor station. Second, the arrangement of active control valves and active compressor groups in a decision's network is information that matters to us. Even though different arrangements, such as two active compressor groups lined after one another versus two active compressor groups working parallely, might lead to the same admissible flow and pressure values on the boundary nodes, they cannot be considered redundant. This may partially be due to reasons we are not concerned with such as the operation costs of different arrangements.

However, there also exist things within a decision's network which are of less importance to us, namely short cuts. They are insofar relevant as they impose flow and pressure bounds and thus affect the values of admissible flows and pressures at the boundary nodes. Nonetheless, short cuts themselves are not of interest to us and it is often the case that they establish unnecessarily complex or redundant structures in a network. This is probably illustrated best by the following examples:

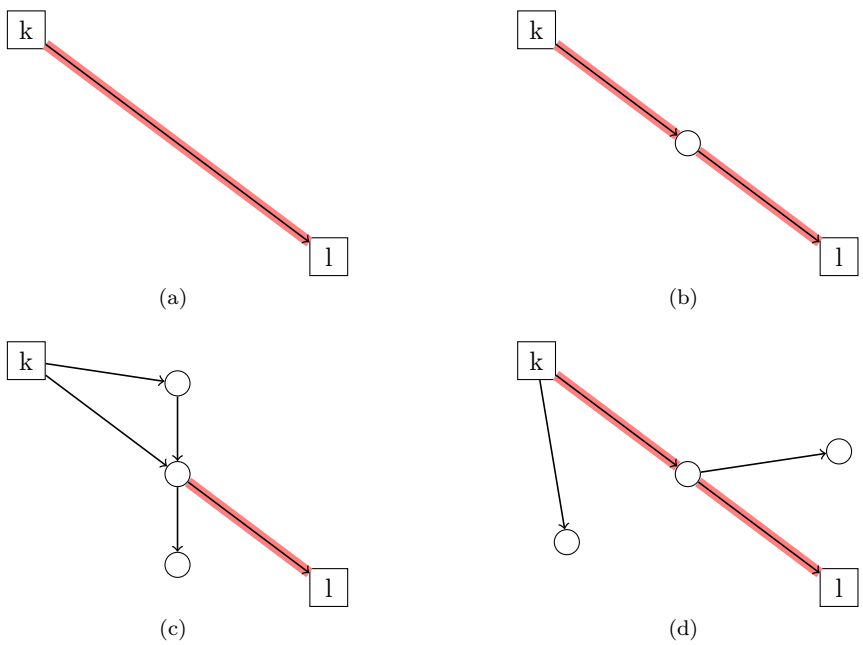


Figure 5.1: All short cuts are assumed to have identical flow bounds. Also, pressure bounds are the same on all nodes, and the respective flow bounds on the boundary nodes k and l are identical for all figures. Marked in red are these short cuts that function as bottlenecks for the flow between k and l.

It is strikingly evident that the networks shown in 5.1a, 5.1b and 5.1c, 5.1d all admit the same flows and pressures when restricted to the boundary nodes k and l. In terms of flow, the edges marked in red function as bottlenecks that force the same limit on flow between k and l for each graph. In terms of pressure, the constraint that

$$p(k) = p(l) \tag{5.10}$$

for each of the graphs easily follows from constraint (2.6) on each of the short cuts. In short, they all model the same scenario.

The idea behind network reduction is to formalize the process of reasoning that we used in the example and specify a set of rules according to which some decision's network can be transformed into a simpler representative. In the case of our example, this would mean reducing all of the networks to the network shown in 5.1a. There are two benefits from this. First of all, we can then tell easily that they're redundant. Second, we favor simpler representations because usually the cost of optimization carried out on some network scales with the network's size. Also, from a human perspective it is often easier to intuitively understand simpler networks.

5.2 Equivalence of Decisions

Before we come to the reduction rules, we ought to make some more notes on comparing different decisions' networks. Let us start with a formal definition of the equivalence of two decisions:

Definition 5.1. Equivalence of Decisions
Two decisions of the same decision group, represented by flow networks Γ_1 and Γ_2, respectively, are considered equivalent if

$$N_{b_1} = N_{b_2} \tag{5.11}$$
$$A_{cv_1} = A_{cv_2} \tag{5.12}$$
$$A_{cg_1} = A_{cg_2} \tag{5.13}$$

and there exist bijective functions $\phi : N_1 \to N_2$ and $\psi : A_1 \to A_2$ such that: ‘

$$\phi(t_1(a)) = t_2(\psi(a)) \tag{5.14}$$
$$\phi(h_1(a)) = h_2(\psi(a)) \tag{5.15}$$
$$\phi|_{N_{b_1}} = id \tag{5.16}$$
$$\psi|_{A_{cv_1} \cup A_{cg_1}} = id \tag{5.17}$$
$$\forall n \in N_1 : (\underline{f}_1(n), \overline{f}_1(n)) = (\underline{f}_2(\phi(n)), \overline{f}_2(\phi(n))) \tag{5.18}$$
$$\forall n \in N_1 : (\underline{p}_1(n), \overline{p}_1(n)) = (\underline{p}_2(\phi(n)), \overline{p}_2(\phi(n))) \tag{5.19}$$
$$\forall a \in A : (\underline{f}_1(a), \overline{f}_1(a)) = (\underline{f}_2(\psi(a)), \overline{f}_2(\psi(a))) \tag{5.20}$$

In other words, we consider two decisions to be equivalent if there exists an isomorphism between their networks which involves flow and pressure bounds and is fixed at boundary nodes and active elements. On the practical side, we have conducted all isomorphy testing with NetworkX [13] which implements the VF2 algorithm for matching graphs (see [6], [5]).

In order to effectively test two decisions for equivalence according to definition 5.1, we must ensure that they are represented by networks in some standardized form which allows for good comparison of graph structure as well as flow and pressure bounds. This has been sketched in figure 5.1 in the previous section. In general, three steps need to be taken to allow effective isomorphy testing. First, we apply network reduction to simplify graph structure. Second, we apply bound propagation to facilitate comparison of flow and pressure bounds. In an additional third step, we deal with comparibility issues resulting from the combination of oriented arcs with negative flows in our networks. Note that this only applies to short cuts as control valves and compressor stations do not admit negative flow by (2.7). To overcome the problem, we ensure that all short cuts have non-negative flow bounds by changing their orientation or splitting them into two oppositely oriented short cuts as appropriate.

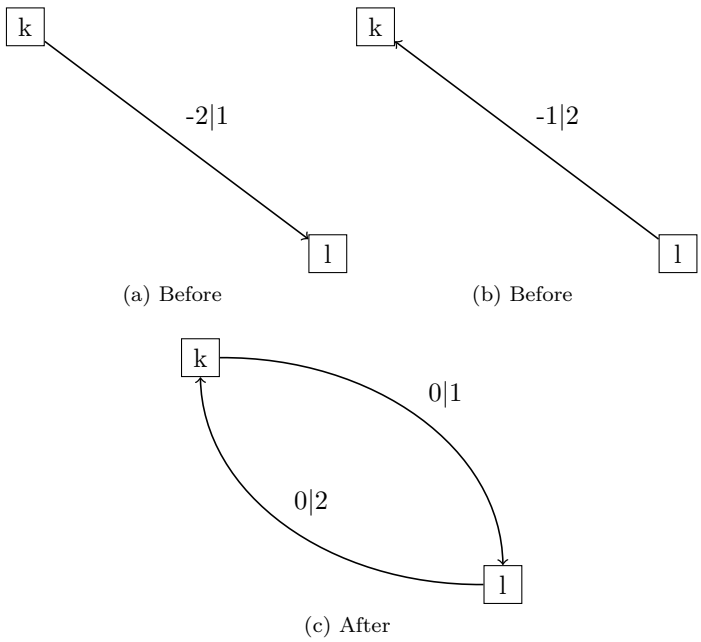


Figure 5.2: (a) and (b) both model the same scenario. However, the variation in the orientation of their short cuts, made possible by the existence of negative flow in our models, makes isomorphy testing generally difficult. Thus, we introduce seperate arcs for the flow from k to l and l to k, respectively.

5.3 Reduction Rules

Preceding the actual reduction process, we remove all weakly connected components in the network which contain neither a boundary node nor any active elements. Such components of isolated short cuts might exist because they have been disconnected from the rest of the network by closed, operable elements.

5.3.1 Rule 1

Rule 1 specifies the removal of an innode $n \in N_i$ of degree one which is incident to a short cut $a \in A_{sc}$. It is formally described in algorithm 4.

Proposition 5.2. *The output of algorithm 4 indeed satisfies all the output constraints given in framework 2.*

Proof. Let the notation be as in algorithm 4 and w.l.o.g. let $t(a) = m$, $h(a) = n$. By the algorithm, $G' = (N \setminus \{n\}, A \setminus \{a\})$ where $a \in A_{sc}$ and $n \in N_i$. Thus, (5.1), (5.2), (5.3), (5.4), (5.5) are trivially satisfied. Now, let $f \in \mathcal{F}(\Gamma)$, $f' := f|_{N' \cup A'}$. We only need to check that f' is admissible for Γ' with respect to the flow conservation constraint at m.

$$f(a) \overset{(1.5)}{=} f(n) \tag{5.21}$$

$$\overset{(2.4)}{\Longrightarrow} f(a) = 0 \tag{5.22}$$

$$\Longrightarrow f(m) = \sum_{\substack{a' \in A^{out}(m) \\ a' \neq a}} f(a') - \sum_{a' \in A^{in}(m)} f(a') \tag{5.23}$$

$$\Longrightarrow f'(m) = \sum_{a' \in A'^{out}(m)} f'(a') - \sum_{a' \in A'^{in}(m)} f'(a') \tag{5.24}$$

Thus, $f' \in \mathcal{F}(\Gamma')$ and (5.6) is satisfied. Analogously, (5.8) is satisfied: Let $f' \in \mathcal{F}(\Gamma')$. Then, $f \in \mathcal{F}(\Gamma)$ for $f|_{N' \cup A'} := f'$, $f(a) := 0$, $f(n) := 0$. Now, let $p \in \mathcal{P}(\Gamma)$, $p' := p|_{N'}$. We only need to check that p' is admissible at m.

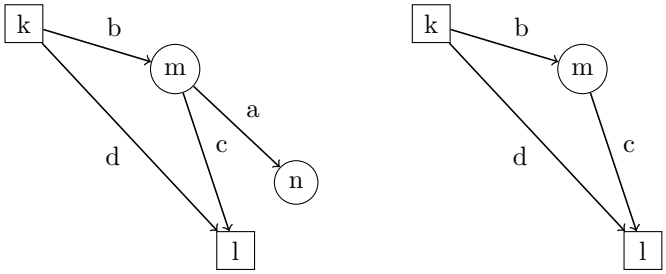
$$p(m) \overset{(2.6)}{=} p(n) \tag{5.25}$$

$$\Longrightarrow \underline{p}(n) \leq p(m) \leq \overline{p}(n) \tag{5.26}$$

$$\Longrightarrow max(\underline{p}(m), \underline{p}(n)) \leq p(m) \leq min(\overline{p}(m), \overline{p}(n)) \tag{5.27}$$

$$\overset{p.d.}{\Longrightarrow} \underline{p}'(m) \leq p'(m) \leq \overline{p}'(m) \tag{5.28}$$

Thus, $p' \in \mathcal{P}(\Gamma')$ and (5.7) is satisfied. Analogously, (5.9) is satisfied: Let $p' \in \mathcal{P}(\Gamma')$. Then, $p \in \mathcal{P}(\Gamma)$ for $p|_{N'} := p'$, $p(n) := p'(m)$. □



(a) Before (b) After

Figure 5.3: Application of rule 1

Algorithm 4 Rule 1

Input: $\Gamma = (G, \underline{f}, \overline{f})$ with pressure bounds $\underline{p}$, $\overline{p}$ and $n \in N_i$, where n has degree one, n adjacent to some $m \in N$ through some $a \in A_{sc}$

Output: $\Gamma' = (G', \underline{f}', \overline{f}')$ with pressure bounds $\underline{p}', \overline{p}'$ according to the output constraints given in framework 2

$N' \leftarrow N \setminus \{n\}$
$A' \leftarrow A \setminus \{a\}$

for $a' \in A'$ **do**
  $\overline{f}'(a') \leftarrow \overline{f}(a')$
  $\underline{f}'(a') \leftarrow \underline{f}(a')$
  $\overline{t}'(a') \leftarrow \overline{t}(a')$
  $h'(a') \leftarrow h(a')$

for $n' \in N' \setminus \{m\}$ **do**
  $\overline{f}'(n') \leftarrow \overline{f}(n')$
  $\underline{f}'(n') \leftarrow \underline{f}(n')$
  $\overline{p}'(n') \leftarrow \overline{p}(n')$
  $\underline{p}'(n') \leftarrow \underline{p}(n')$

$\overline{f}'(m) \leftarrow \overline{f}(m)$
$\underline{f}'(m) \leftarrow \underline{f}(m)$
$\overline{p}'(m) \leftarrow min(\overline{p}(m), \overline{p}(n))$
$\underline{p}'(m) \leftarrow max(\underline{p}(m), \underline{p}(n))$

return $\underline{p}', \overline{p}', \Gamma' := ((N', A'), \underline{f}', \overline{f}')$

5.3.2 Rule 2

Rule 2 specifies a rule for arc contraction that is designed to take on the problem of short cuts which artificially extend paths in our graph. It is formally described in algorithm 5.

Proposition 5.3. *The output of algorithm 5 indeed satisfies all the output constraints given in framework 2.*

Proof. Let the notation be as in algorithm 5 and w.l.o.g. let $t(b) = k$, $h(b) = m$, $t(c) = m$, $h(c) = l$. By the algorithm, $G' = (N \setminus \{m\}, A \setminus \{c\})$ where $c \in A_{sc}$ and $m \in N_i$. Thus, (5.1), (5.2), (5.3), (5.4), (5.5) are trivially satisfied. Now, let $f \in \mathcal{F}(\Gamma)$, $f' := f|_{N' \cup A'}$. We need to check that f' is admissible for Γ' on b as well as with respect to the flow conservation constraint at l.

$$f(m) \overset{(1.5)}{=} f(c) - f(b) \tag{5.29}$$

$$\overset{(2.4)}{\Longrightarrow} f(b) = f(c) \tag{5.30}$$

$$\Longrightarrow max(\underline{f}(b), \underline{f}(c)) \leq f(b) \leq min(\overline{f}(b), \overline{f}(c)) \tag{5.31}$$

$$\overset{p.d.}{\Longrightarrow} \underline{f}'(b) \leq f'(b) \leq \overline{f}'(b) \tag{5.32}$$

$$f(l) = \sum_{a' \in A^{out}(l)} f(a') - \sum_{a' \in A^{in}(l)} f(a') \tag{5.33}$$

$$\overset{(5.30)}{\Longrightarrow} f(l) = \sum_{a' \in A^{out}(l)} f(a') - \sum_{\substack{a' \in A^{in}(l) \\ a' \neq c}} f(a') - f(b) \tag{5.34}$$

$$\overset{p.d.}{\Longrightarrow} f'(l) = \sum_{a' \in A'^{out}(l)} f'(a') - \sum_{a' \in A'in(l)} f'(a') \tag{5.35}$$

Thus, $f' \in \mathcal{F}(\Gamma')$ and (5.6) is satisfied. Analogously, (5.8) is satisfied: Let $f' \in \mathcal{F}(\Gamma')$. Then, $f \in \mathcal{F}(\Gamma)$ for $f|_{N' \cup A'} := f'$, $f(m) := 0$, $f(c) := f'(b)$. Now, let $p \in \mathcal{P}(\Gamma)$, $p' := p|_{N'}$. We check that p' is admissible at l.

$$p(l) \overset{(2.6)}{=} p(m) \tag{5.36}$$

$$\Longrightarrow \underline{p}(m) \leq p(l) \leq \overline{p}(m) \tag{5.37}$$

$$\Longrightarrow max(\underline{p}(l), \underline{p}(m)) \leq p(l) \leq min(\overline{p}(l), \overline{p}(m)) \tag{5.38}$$

$$\overset{p.d.}{\Longrightarrow} \underline{p}'(l) \leq p'(l) \leq \overline{p}'(l) \tag{5.39}$$

By (5.36), p' also fulfills the constraint that b puts on the pressure in Γ'. Thus, $p' \in \mathcal{P}(\Gamma')$ and (5.7) is satisfied. Analogously, (5.9) is satisfied: Let $p' \in \mathcal{P}(\Gamma')$. Then, $p \in \mathcal{P}(\Gamma)$ for $p|_{N'} := p'$, $p(m) := p'(l)$. □

Algorithm 5 Rule 2

Input: $\Gamma = (G, \underline{f}, \overline{f})$ with pressure bounds $\underline{p}, \overline{p}$ and $m \in N_i$, where m has degree two, m adjacent to some $k, l \in N$ through some $c \in A_{sc}$, $b \in A$, respectively

Output: $\Gamma' = (G', \underline{f}', \overline{f}')$ with pressure bounds $\underline{p}', \overline{p}'$ according to the output constraints given in framework 2

$N' \leftarrow N \setminus \{m\}$
$A' \leftarrow A \setminus \{c\}$

for $a' \in A' \setminus \{b\}$ **do**
  $\overline{f}'(a') \leftarrow \overline{f}(a')$
  $\underline{f}'(a') \leftarrow \underline{f}(a')$
  $t'(a') \leftarrow t(a')$
  $h'(a') \leftarrow h(a')$

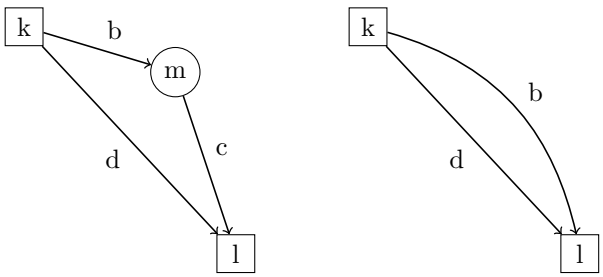
if $t(b) = m$ **then** ▷ Case distinction on the orientation of b
  $t'(b) \leftarrow l$
  $h'(b) \leftarrow h(b)$
else
  $t'(b) \leftarrow t(b)$
  $h'(b) \leftarrow l$

if $t(b) = h(c)$ **or** $t(b) = h(c)$ **then** ▷ Case distinction on the orientation of b, c
  $\overline{f}'(b) \leftarrow min(\overline{f}(b), \overline{f}(c))$
  $\underline{f}'(b) \leftarrow max(\underline{f}(b), \underline{f}(c))$
else
  $\overline{f}'(b) \leftarrow min(\overline{f}(b), -\underline{f}(c))$
  $\underline{f}'(b) \leftarrow max(\underline{f}(b), -\overline{f}(c))$

for $n' \in N' \setminus \{l\}$ **do**
  $\overline{f}'(n') \leftarrow \overline{f}(n')$
  $\underline{f}'(n') \leftarrow \underline{f}(n')$
  $\overline{p}'(n') \leftarrow \overline{p}(n')$
  $\underline{p}'(n') \leftarrow \underline{p}(n')$

$\overline{f}'(l) \leftarrow \overline{f}(l)$
$\underline{f}'(l) \leftarrow \underline{f}(l)$
$\overline{p}'(l) \leftarrow min(\overline{p}(l), \overline{p}(m))$
$\underline{p}'(l) \leftarrow max(\underline{p}(l), \underline{p}(m))$

return $\underline{p}', \overline{p}', \Gamma' := ((N', A'), \underline{f}', \overline{f}')$



(a) Before (b) After

Figure 5.4: Application of rule 2

5.3.3 Rule 3

Rule 3, as a follow-up of rule 2, specifies the merge of two parallel short cuts which can result from the application of rule 2. It is formally described in algorithm 6. Note that our unreduced network models do not feature any parallel arcs.

Proposition 5.4. *The output of algorithm 6 indeed satisfies all the output constraints given in framework 2.*

Proof. Let the notation be as in algorithm 6 and w.l.o.g. let $t(b) = k$, $h(b) = l$, $t(d) = k$, $h(d) = l$. By the algorithm, $G' = (N, A \setminus \{b\})$ where $b \in A_{sc}$. Thus, (5.1), (5.2), (5.3), (5.4), (5.5) are trivially satisfied. Now, let $f \in \mathcal{F}(\Gamma)$, $f' : N' \cup A' \to \mathbb{R}$ with $f'|_{N' \cup (A' \setminus \{d\})} := f|_{N' \cup (A' \setminus \{d\})}$ and $f'(d) := f(d) + f(b)$. We need to check that f' is admissible for Γ' on d as well as with respect to the flow conservation constraint at k and l.

$$\underline{f}(d) + \underline{f}(b) \leq f(d) + f(b) \leq \overline{f}(d) + \overline{f}(b) \tag{5.40}$$

$$\overset{p.d.}{\Longrightarrow} \underline{f}'(d) \leq f'(d) \leq \overline{f}'(d) \tag{5.41}$$

$$f(k) = f(b) + f(d) + \sum_{\substack{a' \in A^{out}(k) \\ a' \in \{b,d\}}} f(a') - \sum_{a' \in A^{in}(k)} f(a') \tag{5.42}$$

$$\overset{p.d.}{\Longrightarrow} f'(k) = f'(d) + \sum_{\substack{a' \in A'^{out}(k) \\ a' \neq d}} f'(a') - \sum_{a' \in A'^{in}(k)} f'(a') \tag{5.43}$$

$$\Longrightarrow f'(k) = \sum_{a' \in A'^{out}(k)} f'(a') - \sum_{a' \in A'^{in}(k)} f'(a') \tag{5.44}$$

Similarly to (5.42), (5.43), (5.44) f' fulfills the flow conservation constraint at l in Γ'. Thus, $f' \in \mathcal{F}(\Gamma')$ and (5.6) is satisfied. Analogously, (5.8) is satisfied: Let $f' \in \mathcal{F}(\Gamma')$. Then, $f \in \mathcal{F}(\Gamma)$ with $f|_{N' \cup (A' \setminus \{d\})} = f'|_{N' \cup (A' \setminus \{d\})}$ and $f(b) + f(d) = f'(d)$. Now, let $p \in \mathcal{P}(\Gamma)$. Then, trivially $p \in \mathcal{P}(\Gamma')$ and (5.7) is satisfied. Analogously, (5.9) is satisfied: Let $p' \in \mathcal{P}(\Gamma)$. Then also $p' \in \mathcal{P}(\Gamma)$, because $d \in A_{sc}$ imposes the constraint of equal pressure at k and l which p' already fulfills for $b \in A_{sc}$. □

Algorithm 6 Rule 3

Input: $\Gamma = (G, \underline{f}, \overline{f})$ with pressure bounds $\underline{p}, \overline{p}$ and $k, l \in N$ where k, l adjacent through some $b, d \in A_{sc}$

Output: $\Gamma' = (G', \underline{f}', \overline{f}')$ with pressure bounds $\underline{p}', \overline{p}'$ according to the output constraints given in framework 2

$N' \leftarrow N$
$A' \leftarrow A \setminus \{b\}$

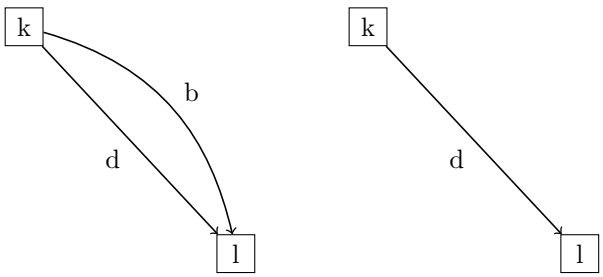
for $a' \in A \setminus \{d\}$ **do**
  $\overline{f}'(a') \leftarrow \overline{f}(a')$
  $\underline{f}'(a') \leftarrow \underline{f}(a')$
  $t'(a') \leftarrow t(a')$
  $h'(a') \leftarrow h(a')$

if $h(b) = h(d)$ **then** ▷ Case distinction on the orientation of b, d
  $\overline{f}'(d) \leftarrow \overline{f}(d) + \overline{f}(b)$
  $\underline{f}'(d) \leftarrow \underline{f}(d) + \underline{f}(b)$
else
  $\overline{f}'(d) \leftarrow \overline{f}(d) - \underline{f}(b)$
  $\underline{f}'(d) \leftarrow \underline{f}(d) - \overline{f}(b)$

$t'(d) \leftarrow t(d)$
$h'(d) \leftarrow h(d)$

for $n' \in N'$ **do**
  $\overline{f}'(n') \leftarrow \overline{f}(n')$
  $\underline{f}'(n') \leftarrow \underline{f}(n')$
  $\overline{p}'(n') \leftarrow \overline{p}(n')$
  $\underline{p}'(n') \leftarrow \underline{p}(n')$

return $\underline{p}', \overline{p}', \Gamma' := ((N', A'), \underline{f}', \overline{f}')$



(a) Before (b) After

Figure 5.5: Application of rule 3

5.3.4 Reduction Process

Algorithm 7 Reduction Process

Input: $\Gamma = (G, \underline{f}, \overline{f})$ with pressure bounds $\underline{p}, \overline{p}$

Output: $\Gamma' = (G', \underline{f}', \overline{f}')$ with pressure bounds $\underline{p}', \overline{p}'$ according to the output constraints given in framework 2

```
Γ', p', p̄' ← Γ, p, p̄
Queue ← enqueue all n ∈ N'

while Queue not empty do
    n ← dequeue from Queue

    if rule 1 applicable for n then
        m ← neighbor of n
        Γ', p', p̄' ← apply rule 1 to Γ', p', p̄' for n
        Queue ← enqueue m

    else if rule 2 applicable for n then
        k, l ← neighbors of n
        Γ', p', p̄' ← apply rule 2 to Γ', p', p̄' for n

        if rule 3 applicable for k, l then
            Γ', p', p̄' ← apply rule 3 to Γ', p', p̄' for k, l
            Queue ← enqueue k, l

return p', p̄', Γ' := ((N', A'), f', f̄')
```

We may now combine rules 1 to 3 to a single procedure. We do so by going through all nodes in the graph and check if one of our rules is applicable for that node. Also, we note that the application of rule 1 or rule 3, respectively, decreases the degree of nodes in our graph. This might allow the application of a rule for these nodes that was not possible before. In order to check these nodes again, we maintain a queue of nodes to go through and check. The complete reduction process is formally described in algorithm 7.

Corollary 5.5. *The output of algorithm 7 indeed satisfies all the output constraints given in framework 2.*

Proof. Immediately follows from propositions 5.2, 5.3 and 5.4 by induction over the application of rules. □

5.4 Possible Improvements

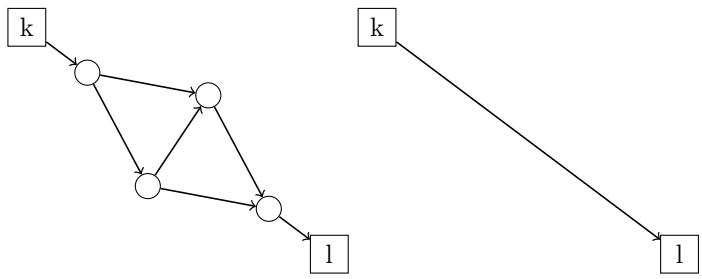
The rules we have specified perform a good amount of reduction for compressor stations that we have tested our method on. However, on some networks our rules were too limited and failed to achieve the amount of reduction that we would ideally like to achieve. An example of this is shown in figure 5.6. We suggest three ways to further improve the effect of the network reduction.

First, it might help to specify more rules or somehow generalize our existing rules in order to identify appropriate structures in the network that we wish to contract. A starting point for this could be the generalization of the type of structure that we fail to contract in figure 5.6. It remains in question, by what means these structures can be efficiently identified.

Second, we were able to improve the process of bound propagation by borrowing tools from linear programming. Perhaps, a similiar improvement can be made for network reduction. More precisely, it is evident that if we view bound tightening on a network as an optmization problem, then what network reduction effectively does is that it performs a preprocessing of the optimization problem through variable and constraint elimination. Therefore, it might be helpful to explore network reduction from an algebraic perspective which could be a way around the problem of having to deal explicitely with different graph structures.

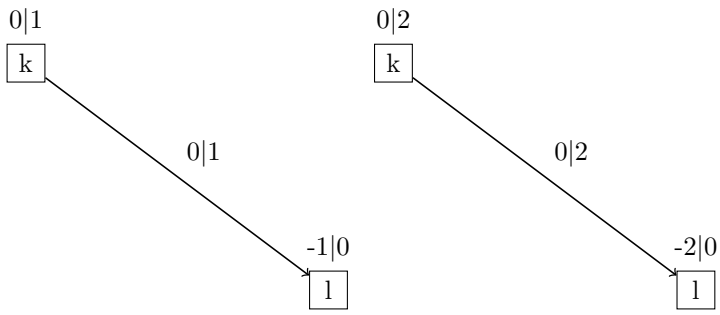
Third, we have so far only considered redundancy in the case that two decisions describe the exact same scenario. However, redundancy also exists when two decisions describe two scenarios where one of the scenarios is completely contained in the other. This is examplarily shown in figure 5.7. Therefore, it might be possible to further

improve redundancy detection by extending the network isomorphism testing to a comparation method which is able to recognize this type of redundancy.



(a) Before (b) After

Figure 5.6: Ideally, we would like to perform this reduction since the complex structure of short cuts merely is a bloated path from k to l. However, none of our rules can be applied in this case because all innodes have degree three.



(a) (b)

Figure 5.7: The scenario described by the left decision, where the amount of admissible flow from k to l is between 0 and 1, is completely contained in the scenario described by the right decision, where the amount of admissible flow from k to l is between 0 and 2.

6 Computational Experience

6.1 Implementational Issues

We used Python 2.7 to implement the bound tightening and network reduction methods described in chapter 4 and 5, respectively. We also implented the creation of decisions from partial decision as well as the application of decisions as described in chapter 3. For general graph representation and isomorphism testing according to section 5.2, we used NetworkX [13] which implements the VF2 algorithm for matching graphs (see [6], [5]). In addition, we used CPLEX [17] to solve linear programs that arise in the bound tightening procedure. We accessed CPLEX through Pyomo [15] [14] which provides a standardized Python interface for a large number of solvers.

6.2 Test Instances

We tested our methods on models of real compressor stations that have been provided by Open Grid Europe GmbH. Besides the compressor stations' graphs, the models include all the values for flow and pressure bounds as detailed in section 2.2. In addition, the models contain a partition of the compressor stations into decision groups, and a list of partial decisions for each decision group as described in section 3.1. We tested our methods on five compressor stations which can be characterized as follows:

- Station A is a small compressor station of low complexity which consist of only one decision group.

- Station B and Station C are medium-sized compressor stations of moderate complexity which consist of two decision groups each.

- Station D and Station E are large compressor station of high complexity. Station E consists of two decision groups. Station D consists of three decision groups.

For all stations (except Station A which only consists of a single decision group), we refer to the decision groups that we were given by Station B1, Station B2 and so on.

Then, Station B refers to the graph and decision group that result from the union of Station B1 and Station B2, and the partial decisions for Station B are all products of partial decisions from Station B1 and Station B2. Similarly, this applies for Station C, Station D and Station E.

6.3 Test Results

All our computations have been conducted on a computer with a 64-bit, 3.70GHz Intel® Xeon® E3-1290 V2 CPU and 16 GB RAM.

Table 6.1 shows the combined effect of determining validity and detecting redundancy through network reduction, optimization-based bound tightening and isomorphism testing. As discussed in chapter 4, optimization-based bound tightening results in tight flow and pressure bounds, and the number of valid decisions is exact for every decision group. With respect to redundancy detection, table 6.1 shows that the effectiveness of network reduction varies between stations. It proves to be rather effective, for example, for Station C and Station D, but shows very limited effect for Station E. Of course, there are multiple properties of the stations' graphs, such as their structure or the number of boundary nodes in them, that influence the effectiveness of network reduction.

However, our results are partially skewed by a questionable property of the models that we rely on. That is, some compressor groups can be operated in active mode such that they exactly compensate the pressure loss occuring at their inlet and outlet. Clearly, there is no point in this, because it is equivalent to bypass mode, but consumes energy. Thus, we also tested our methods with the additional constraint that an active compressor group must increase the pressure in its working direction by some $\epsilon > 0$:

$$\forall a \in A_{cg} : p(h(a)) \geq p(t(a)) + \epsilon \tag{6.1}$$

The results of this are shown in 6.2.

In addition, 6.3 shows that the process of network reduction significantly reduces the size of decisions' networks. Even though there possibly is room for improvement in redundancy detection between different decisions, this means that network reduction allows us to detect a lot of redundancy within a single decision's network. This alone can be considered a great step in preprocessing decision data for further optimization purposes. However, the exact gain in performance for other optimization problems, that is achieved by our decision data preprocessing, is still to be measured.

Decision Group	Partial Decisions	All Decisions	Valid Decisions	Different Networks	Total Runtime
Station A	53	655	204	146	159s
Station B	90	200	185	150	119s
Station C	100	825	726	464	1008s
Station D	450	2088	1710	1338	2918s
Station E	280	2750	2640	2616	4331s

Table 6.1: Results of our process of optimization-based bound tightening, network reduction and isomorphism testing where "Different Networks" refers to the number of valid, non-equivalent decisions.

Decision Group	Partial Decisions	All Decisions	Valid Decisions	Different Networks	Total Runtime
Station A	53	655	83	43	66s
Station B	90	200	170	135	185s
Station C	100	825	627	377	860s
Station D	450	2088	1440	1104	2332s
Station E	280	2750	2640	2616	4120s

Table 6.2: Results of our process of optimization-based bound tightening, network reduction and isomorphism testing using the additional constraint (6.1)

Decision Group	Before Reduction						After Reduction					
	Short Cuts			Innodes			Short Cuts			Innodes		
	min	max	avg	min	max	avg	min	max	avg	min	max	avg
Station A	2	22	17	1	17	15	0	12	6	0	7	4
Station B	17	35	31	16	28	26	1	10	5	0	4	2
Station C	26	35	30	16	21	19	11	16	14	1	4	2
Station D	59	70	66	51	56	55	11	21	15	3	8	5
Station E	43	64	57	38	53	49	6	19	13	1	9	5

Table 6.3: Effect of network reduction on the size of the network where minimum, maximum and average for a decision group are taken over all decisions for that decision group.

Bibliography

[1] BMWi. *Erdgasversorgung in Deutschland*. 2016. URL: http://www.bmwi.de/DE/Themen/Energie/Konventionelle-Energietraeger/gas.html (visited on 05/04/2016).

[2] Ralf Borndörfer, Berkan Erol, and Thomas Schlechte. "Optimization of macroscopic train schedules via TS-OPT". In: *Proceedings of 3rd International Seminar on Railway Operations Modelling and Analysis - Engineering and Optimisation Approaches*. Ed. by I. A. Hansen et al. 2009.

[3] Ralf Borndörfer et al. "Aggregation Methods for Railway Networks". In: *Proceedings of 4th International Seminar on Railway Operations Modelling and Analysis (IAROR)*. Ed. by I. A. Hansen et al. Vol. 4. 2011. URL: http://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/1188.

[4] William J. Cook et al. *Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011. ISBN: 9781118031391. URL: https://books.google.de/books?id=tarLTNwM3gEC.

[5] Luigi P. Cordella et al. "An improved algorithm for matching large graphs". In: *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen*. 2001, pp. 149–159.

[6] Luigi P. Cordella et al. "A (sub)graph isomorphism algorithm for matching large graphs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (10 2004), pp. 1367–1372. DOI: 10.1109/TPAMI.2004.75.

[7] George B. Dantzig. *Linear Programming and Extensions*. Princeton, NJ, USA: Princeton University Press, 1963.

[8] George B. Dantzig. *Origins of the Simplex Method*. Tech. rep. Stanford University, Systems Optimization Laboratory, 1987. URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a182708.pdf.

[9] Berkan Erol. *Models for the train timetabling problem*. Diploma thesis, TU Berlin, 2009.

[10] Ambros Gleixner et al. *Three Enhancements for Optimization-Based Bound Tightening*. ZIB-Report 15-16. Zuse Institute Berlin (ZIB), 2016. URL: https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/5780.

[11] Martin Grötschel. *Diskrete Optimierung (lecture notes)*. 2015. URL: http://www.zib.de/groetschel/teaching/SS2015/Skriptum_ADM_II-2015-07-20.pdf.

[12] Inc. Gurobi Optimization. *Gurobi Optimizer Reference Manual.* URL: `http://www.gurobi.com`.

[13] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring network structure, dynamics, and function using NetworkX". In: *Proceedings of the 7th Python in Science Conference (SciPy2008).* Pasadena, CA USA, Aug. 2008, pp. 11–15.

[14] William E Hart, Jean-Paul Watson, and David L Woodruff. "Pyomo: modeling and solving mathematical programs in Python". In: *Mathematical Programming Computation* 3.3 (2011), pp. 219–260.

[15] William E Hart et al. *Pyomo–optimization modeling in python.* Vol. 67. Springer Science & Business Media, 2012.

[16] Michael Herty, Jan Mohring, and Veronika Schleper. "A new model for gas flow in pipe networks". In: *Mathematical Methods in the Applied Sciences* 33 (7 2010), pp. 845–855.

[17] IBM ILOG. *CPLEX Optimization Studio.* URL: `http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/`.

[18] Thorsten Koch et al., eds. MOS-SIAM Series on Optimization. SIAM, 2015.

[19] Wikimedia Commons (Alois Köppl). *Erdgaskompressorstation Rothenstadt-Weiherhammer.* URL: `https://de.wikipedia.org/wiki/Datei:Rothenstadt_Megal_Verdichterstation_2012_01.jpg`.

[20] Zuse Institute Berlin (ZIB). *SCIP: solving constraint integer programs.* URL: `http://scip.zib.de/`.