Felix Hennings[1], Kai Hoppmann-Baum[2], Janina Zittel[3]

# Optimizing transient gas network control for challenging real-world instances using MIP-based heuristics

[1] 0000-0001-6742-1983
[2] 0000-0001-9184-8215
[3] 0000-0002-0731-0314

ZIB Report 22-08 (May 2022)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30 84185-0
Telefax: +49 30 84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

# Optimizing transient gas network control for challenging real-world instances using MIP-based heuristics

Felix Hennings[1] Kai Hoppmann-Baum[2] Janina Zittel[3]

May 24, 2022

**Abstract**

Optimizing the transient control of gas networks is a highly challenging task. The corresponding model incorporates the combinatorial complexity of determining the settings for the many active elements as well as the non-linear and non-convex nature of the physical and technical principles of gas transport. In this paper, we present the latest improvements of our ongoing work to solve this problem for real-world, large-scale problem instances: By adjusting our mixed-integer non-linear programming model regarding the gas compression capabilities in the network, we reflect the technical limits of the underlying units more accurately while maintaining a similar overall model size. In addition, we introduce a new algorithmic approach that is based on splitting the complexity of the problem by first finding assignments for discrete variables and then determining the continuous variables as locally optimal solution of the corresponding non-linear program. For the first task, we design multiple different heuristics based on concepts for general time-expanded optimization problems that find solutions by solving a sequence of sub-problems defined on reduced time horizons. To demonstrate the competitiveness of our approach, we test our algorithm on particularly challenging historic demand scenarios. The results show that high-quality solutions are obtained reliably within short solving times, making the algorithm well-suited to be applied at the core of time-critical industrial applications.

## 1 Introduction

The natural gas supply of Europe and especially Germany is subject to a disruptive transformation. With the possibility of cutting the supply from Russia and the recently solidified plans to build LNG terminals in Germany as published by the government [Federal Ministry for Economic Affairs and Climate Action, 2022], new flow patterns are likely to emerge in the corresponding transport networks. Moreover, the usage of fossil-based natural gas will soon be terminated in favor of more sustainable alternatives as part of the energy transformation towards massive reductions in $CO_2$ emissions. In particular, the natural gas networks may be used to store synthesized methane generated via power-to-gas from excess renewable energy or could (partially) be repurposed for hydrogen transport, see, for example, the scenario framework for the German *Gas Network Development Plan* as published by the German gas network operators [FNB Gas – Association of German transmission system operators, 2021]. Especially when applying power-to-gas strategies, the intraday supply patterns are likely to become more volatile, complicating the control of the networks even further.

Mathematical optimization can assist the network operators in preparing for these new challenges and ensuring a secure and efficient network operation. However, solving the corresponding optimization problems is hard. On the one hand, the physics of gas transport in pipelines describing the interplay of gas pressure and flow is captured by non-convex equations. On the other hand, the network contains many active elements to control the gas flow, like valves to dynamically change the network topology and compressors to increase the gas pressure. Apart from the non-convex description of the maximum power available to compressors, these control options also introduce an immense combinatorial complexity to the problem. Since the problem is both practically relevant and theoretically challenging, it has been intensively studied in the past, and we will give an overview split into two categories in the following.

First, there is the *stationary* variant of the problem. It defines an equilibrium state of the network by assuming a single infinitely long time step and can, for example, be used for planning purposes. For a

[1] 0000-0001-6742-1983
[2] 0000-0001-9184-8215
[3] 0000-0002-0731-0314

profound introduction to stationary gas network optimization, we recommend the book of [Koch et al., 2015] or the paper of [Pfetsch et al., 2015] for a summary of the corresponding findings. They model the problem as a mixed-integer non-linear program (MINLP) and deal with the two sources of complexity separately by first finding solutions for the discrete decision variables and afterwards completing these to highly precise solutions by solving a corresponding non-linear programming (NLP) model. A more recent study on the stationary problem is presented in [González Rueda et al., 2019]. They use a model based on simplified structures representing the network areas containing the compressors. Algorithmically, the authors mainly focus on solving an NLP-model variant of the problem with a two-step sequential linear programming (SLP) approach, which is inspired by an early attempt at solving the problem reported in [Pratt and Wilson, 1984]. However, by replacing the first step with sequential mixed-integer programming, they can convert their algorithm into a heuristic for MINLP problems. Test on real-world instances, which only have a limited number of binary variables, reveal a performance similar to state-of-the-art NLP and MINLP solvers.

Second, the *transient* variant of the problem aims at finding a short-term control for the network over discrete future time steps. An early publication on the topic was the thesis of [Moritz, 2007]. Here, the problem was formulated as a pure mixed-integer program (MIP) by approximating the non-linear pipe equations and compression power constraints using piece-wise linear functions. To aid the corresponding solver, the author proposed a special branching scheme for the piece-wise linear functions and used a simulated annealing heuristic, which was also described in [Mahlke et al., 2007]. Shortly after, [Domschke et al., 2011] presented their approach for a similar problem formulation, which again used piece-wise linear functions to approximate the non-linear parts of the network. They iteratively improved the linearization by first solving the corresponding MIP model, then using the binary solution values to form a non-linear programming model using the original not approximated equations, and afterwards refining the linearization by adding mesh-points corresponding to the found solutions. The procedure is stopped when a predefined approximation threshold is satisfied. In their computational results, they report problems in solving the single MIP models to optimality even after multiple hours. In a more recent study, [Hahn et al., 2017] directly solve an MINLP formulation of the problem. Although the problem is non-convex, they use a convex solver and apply a customized branching rule for the problem. In their experiments, they design scenarios for small test networks of less than 10 nodes that include the reversal of the gas flow direction on single pipelines, which was not possible in the networks used in [Moritz, 2007] and [Domschke et al., 2011]. Despite the size of the networks, the different variants of their algorithm need, even in the best cases, hours to finish. Another contribution was made by [Burlacu et al., 2019]. By introducing a new discretization for the differential equations describing the gas flow in pipes, they were able to reformulate the transient problem as a stationary one on a time-expanded graph and use a specialized algorithm designed for the stationary case to solve the problem. It is based on iteratively solving mixed-integer programming relaxations of the problem, which they again obtained by applying a piece-wise linearization. By refining the linearization in each iteration, their approach converges to the globally optimal solution. To speed up the solving process, the intermediate MIP solutions were completed to valid primal solutions of the overall problem using an NLP solver. For the one evaluated instance, the algorithm finds a solution with a gap of less than 10% in less than 60 seconds and less than 8% in half an hour. However, after this point, even single iterations of the algorithm reach run times of over an hour and may not even improve the gap.

Finally, we mention the work of [Hoppmann-Baum et al., 2021], who presented a decision support system for network operators to ensure a secure and efficient network operation. The system is split into two separate stages: First, an MINLP problem formulation covering the complete network is solved, which approximates the capabilities of the technical elements located at large pipeline intersections, the *network stations*. In particular, each of these stations is replaced with a simplified graph model using artificial arcs. Non-technical control measures available to the network operators are realized as slack variables to adjust the future supplies and demands in terms of pressure and inflow. Their usage as well as changes in the technical control are penalized in the objective functions of a tri-level optimization model. To derive a feasible solution, a series of linearized MIP models are solved, which use two types of primal heuristics to determine good initial solutions. Afterwards, a linear programming (LP) model is solved to smooth the solution, followed by an SLP approach that transforms the solution into one respecting the original non-linear constraints of the MINLP problem. In a computational study on subsequent real-world instances spanning multiple days, the algorithm's run time and the differences in the solutions are evaluated. In the second stage of the decision support system, the actual control measures recommended to the network operators are determined by solving highly detailed models for each network station individually, as described in [Hennings et al., 2021a]. Note that since the overall system runs in regular intervals to provide up-to-date control recommendations, it must reliably satisfy strict time limit restrictions.

In this paper, we present several improvements to the simplified network-wide model of the first stage of the decision support system described in [Hoppmann-Baum et al., 2021]: We suggest an alternative model for the artificial network station arcs representing the gas compressors, which is closer to the compression capabilities of the highly detailed network station model used in the second stage without increasing the model's size. Furthermore, we formulate a new algorithm to solve the MINLP problem of the first stage, which finds better solutions faster than the original approach. In particular, we first determine sets of values for all the binary variables of the model, which we call *binary assignments*, and afterwards complete these to full solutions of the MINLP by solving the remaining non-linear programming (NLP) model. The binary assignments are obtained by combining a sequential mixed-integer programming (SMIP) routine with heuristics concepts for time-expanded problems: The rolling horizon and the aggregate horizon heuristic. We additionally apply a dynamic branch-and-bound node limit to the single model solves to further reduce the maximum run times. Finally, we thoroughly test the proposed algorithm's run time and its solution quality against valid lower bounds. As a test set, we identify especially hard-to-solve real-world instances of the problem that feature many supply and demand changes. Thereby we verify that our algorithm is suitable for time-critical industry applications and more volatile future supply patterns.

Our proposed approach focuses on finding good primal solutions fast and does not provide a valid lower bound on the value of the problem's optimal solution. This enables us to tackle larger and more challenging problem instances than those found in the literature while respecting strict time limits. We follow the idea of dealing with the combinatorial and non-linear aspects of the problem separately and also use sequential mixed-integer programming for finding good binary assignments. However, since we do not aim at finding lower bounds, we do not need to use an iteratively refined piece-wise linear formulation but only an updatable linearization, similar to what [González Rueda et al., 2019] propose for the first level of their algorithm. As a consequence, the general size of the MIP problems is smaller and does not increase during the solving process, which is one of the reasons for the reported slow lower bound convergence of the transient approaches listed above.

We note that the heuristic concepts we use for time-expanded problems are well known in the literature. The rolling horizon approach was, for example, used for the optimization of product manufacturing in [Tiacci and Saetta, 2012], distributed power generation in [Capitanescu, 2017], or as one of the primal heuristics in [Hoppmann-Baum et al., 2021]. A theoretical analysis of rolling horizon approaches, as well as a more extensive list of applications, is given in [Glomb et al., 2022]. Similarly, heuristics using an aggregation of a time horizon were often described, as in the example of scheduling the iron ore production in [Newman and Kuchta, 2007] or the extension of freight transport networks in [Boland et al., 2013].

The rest of the paper is structured as follows: Section 2 introduces the MINLP problem formulation, where we particularly highlight the differences with respect to the model used in [Hoppmann-Baum et al., 2021]. In the following Section 3, we present our heuristic algorithmic approach for solving this problem. The computational results are the content of Section 4, where we describe the creation of the instance sets, compare a base version of our algorithm against a global MINLP solver in terms of solution quality on smaller instances, and then compare the actually proposed algorithm against the base algorithm version on a more challenging instance set. We close with concluding remarks in Section 5.

## 2 Model formulation

Our problem is based on the gas flow model used for the first stage of the decision support system presented in [Hoppmann-Baum et al., 2021]. As in there, we assume that the gas network contains a set $\mathcal{I}$ of *network stations*. Those are sub-networks that include the majority of all actively controllable network elements, among them all the network's compressor stations, and are mainly located at large pipeline intersections. For the model, the inner network topology of each station is replaced by a set of artificial arcs representing an approximation of the station's gas routing, compression, and regulation capabilities. This artificial model was created manually by industry experts who work for the network operator and have an excellent knowledge of the individual stations and their properties.

In the following, we summarize the differences between our problem formulation and the one of Hoppmann-Baum et al.. Note that we refer to their model as the *base model* for the remainder of the paper.

- As the most crucial change, we introduce in Section 2.5.4 a new model for the artificial compressor arcs in the network stations. It is based on the usage of configurations for each artificial arc, which we construct based on the properties of the compressor stations of the original network station topology. We show that the model is much closer to the highly detailed compressor station model used in [Hennings et al., 2021a] while maintaining a similar model size with respect to the number of variables and constraints.

- The second major difference is the introduction of artificial valves at the boundaries of the network stations in Section 2.7. With these, it is possible to decouple the pressure inside and outside the station at these points. Note that this was initially suggested in [Hoppmann-Baum, 2022].
- In this paper, we do not use a tri-level model to reflect strict objective function priorities for using certain gas network control measures. Instead, we use a single objective function representing the weighted sum of different terms to penalize them, see Section 2.10. This objective also includes *smoothing* terms, which minimize pressure and inflow changes at the network station boundaries over time. As a final change in the objective, we include a newly created incentive to reduce the usage of compressor units based on the new artificial compressor configurations to avoid unnecessary energy consumption.
- We use the regulator model of [Hennings et al., 2021a], which is slightly different from the base regulator model.
- In the artificial arc models introduced in Section 2.5, we do not include *combined arcs*, which have gas compression as well as regulation capabilities, as they do not appear in our real-world-based test instances, see Table 2 in Section 4.1. For the same reason, we did not include artificial *bi-directional compressor arcs* or *bi-directional combined arcs*.
- Finally, we also do not introduce *flow direction conditions* as they are not used in the network station description of our test instances.

When formulating our model below, we often use *indicator constraints*, which have two parts: The first one is a logical condition in the form of a binary variable $y$ attaining a value $b \in \{0,1\}$ and the second one an implied linear constraint $a^T x \leq a_0$ of variables $x = \{x_1, \dots, x_n\}$ and coefficients $a \in \mathbb{R}^n$, $a_0 \in \mathbb{R}$, which only holds if the logical condition is true. We denote such an indicator constraint by

$$y = b \quad \Longrightarrow \quad a^T x \leq a_0.$$

Note that we also use

$$y = b \quad \Longrightarrow \quad a^T x = a_0$$

as an abbreviation of the two indicator constraints

$$y = b \quad \Longrightarrow \quad a^T x \leq a_0 \qquad \text{and} \qquad y = b \quad \Longrightarrow \quad -a^T x \leq -a_0.$$

Indicator constraints can be reformulated as linear constraints via the *big-M* approach in case all the variables $x = \{x_1, \dots, x_n\}$ are bounded, see [Bonami et al., 2015] for a general introduction to the topic. A list of all the variables used in our model, together with their domains, meanings, and units, is given in Table A.1 in the Appendix.

## 2.1 Gas flow in networks

We model a gas network as a directed graph $G = (\mathcal{V}, \mathcal{A})$ with $\mathcal{V}$ being the set of nodes or vertices and $\mathcal{A} = \mathcal{A}^{\text{pi}} \,\dot{\cup}\, \mathcal{A}^{\text{va}} \,\dot{\cup}\, \mathcal{A}^{\text{rg}} \,\dot{\cup}\, \mathcal{A}^{\text{ar}}$ being the set of arcs representing the single element sets present in the network: Pipes $\mathcal{A}^{\text{pi}}$, valves $\mathcal{A}^{\text{va}}$, regulators $\mathcal{A}^{\text{rg}}$, and artificial network station arcs $\mathcal{A}^{\text{ar}}$. The latter is again composed of the different types of artificial arcs found in the network stations: Shortcuts $\mathcal{A}^{\text{arSc}}$, regulating arcs $\mathcal{A}^{\text{arRg}}$, compressor arcs $\mathcal{A}^{\text{arCp}}$, and bi-regulating arcs $\mathcal{A}^{\text{arBiRg}}$. For station $i \in \mathcal{I}$, we denote by $\mathcal{A}_i^{\text{ar}} = \mathcal{A}_i^{\text{arSc}} \,\dot{\cup}\, \mathcal{A}_i^{\text{arRg}} \,\dot{\cup}\, \mathcal{A}_i^{\text{arBiRg}} \,\dot{\cup}\, \mathcal{A}_i^{\text{arCp}}$ we denote its set of artificial arcs and by $\mathcal{V}_i^{\text{fn}} \subseteq \mathcal{V}$ the nodes forming the station's boundary, which we call *fence nodes*. Formally, the fence nodes are those nodes having incident arcs from within the station as well as from outside of the station, i.e.,

$$\forall v \in \mathcal{V}_i^{\text{fn}} : (\exists a^{\text{in}} \in \mathcal{A}_i^{\text{ar}} : a^{\text{in}} = (v,r) \vee a^{\text{in}} = (\ell, v)) \wedge (\exists a^{\text{out}} \in \mathcal{A} \setminus \mathcal{A}_i^{\text{ar}} : a^{\text{out}} = (v,r) \vee a^{\text{out}} = (\ell, v)).$$

Note that, in theory, a node can be the fence node of different stations. However, this does not happen in practice. Furthermore, there can be artificial non-fence nodes inside the station, which are needed for the manual station modeling. Finally, we assume for each station $i \in \mathcal{I}$ that the graph induced by the station's arcs $\mathcal{A}_i^{\mathrm{ar}}$ is connected.

Furthermore, we consider a discrete time horizon $\mathcal{T}_0 := \{0, \ldots, t^{\max}\}$, where $\mathcal{T} := \mathcal{T}_0 \setminus \{0\}$ denotes the set of all future time steps. We denote by $\tau(t)$ the time difference in seconds from timestep $t \in \mathcal{T}_0$ to the initial state time step 0.

The gas flow on each arc $a = (\ell, r)$ from node $\ell$ to node $r$ at some time $t \in \mathcal{T}_0$ is characterized by the pressure $p$ and the mass flow $q$ at each end of the arc, i.e., the incoming pressure $p_{\ell,a,t}$, the outgoing pressure $p_{r,a,t}$, the incoming mass flow $q_{\ell,a,t}$, and the outgoing mass flow $q_{r,a,t}$. Here, negative flow values represent flow against the arc's orientation, while the pressure values are always positive. Since we assume that only pipes have a nonzero volume, the inflow and outflow of all non-pipe arcs is identical. Thus, we define the mass flow of an arc $a \in \mathcal{A} \setminus \mathcal{A}^{\mathrm{pi}}$ as $q_{a,t} := q_{\ell,a,t} = q_{r,a,t}$.

The quantities of the arcs are connected at the nodes of the network. For all arcs incident to a node, the corresponding pressure values at that end node are equal. Hence, we can reference this value as pressure $p_{v,t}$ of node $v \in \mathcal{V}$ for time $t \in \mathcal{T}_0$ and it holds:

$$p_{v,t} = p_{v,a,t} \quad \forall a \in \mathcal{A} : a = (v, r) \vee a = (\ell, v).$$

For the mass flow holds the conservation of mass, a Kirchhoff-type law which demands that the flow into and out of a node $v \in \mathcal{V}$ must be balanced for all time steps $t \in \mathcal{T}$, i.e.,

$$\sum_{a \in \mathcal{A}: a=(v,r)} q_{v,a,t} - \sum_{a \in \mathcal{A}: a=(\ell,v)} q_{v,a,t} = d_{v,t}. \tag{1}$$

This equation is also known as *flow balance equation*. Note that the inflow $d_{v,t}$ denotes gas injection into node $v$ at time $t$ from outside of the network for positive values and gas withdrawal for negative values. Furthermore, the inflow value at time $t$ represents the average gas inflow rate into the node in the continuous time interval between $t$-1 and $t$. We partition the nodes into three sets: The set of *entries* $\mathcal{V}^+$ with $d_{v,t} \geq 0$ for $v \in \mathcal{V}^+$, the set of *exits* $\mathcal{V}^-$ with $d_{v,t} \leq 0$ for $v \in \mathcal{V}^-$, and the set of inner nodes $\mathcal{V}^0$ with $d_{v,t} = 0$ for $v \in \mathcal{V}^0$ for all $t \in \mathcal{T}$. We call the union of all entries and exits the *boundary nodes* $\mathcal{V}^{\mathrm{b}}$ of the network.

For each pressure and flow variable, we are given lower and upper bounds for each $t \in \mathcal{T}$, which are denoted by bars below and above the corresponding variable.

Apart from the network topology, we are also given the initial state of the network, which specifies for $t = 0$ the node pressure $p_{v,0}$ for nodes $v \in \mathcal{V}$ and the arc flow values $q_{l,a,0}$ and $q_{r,a,0}$ for arcs $a = (l, r) \in \mathcal{A}$. Furthermore, the initial states of the active network elements outside the network stations, i.e., valves and regulators, are given.

## 2.2 Pipes

For pipes, we use the *friction-dominated* model given as (ISO3) in [Domschke et al., 2021]. It is based on the *Euler equations* modeling the one-dimensional gas flow in straight cylindric pipes and can be derived by assuming a time-constant temperature (isothermal conditions), gas mixture, and compressibility factor, as well as clearly subsonic gas velocities and a negligible inertia term, which both are usually given in real-world conditions, see [Osiadacz, 1996; Domschke et al., 2021] and [Hennings, 2021]. The usage of the *implicit box* scheme of [Domschke et al., 2011; Kolb et al., 2010] as a discretization yields the model for a pipe $a = (\ell, r) \in \mathcal{A}^{\mathrm{pi}}$ and two adjacent time points $t_1 \in \mathcal{T}$ and $t_0 := t_1 - 1$ given as

$$p_{\ell,t_1} + p_{r,t_1} - p_{\ell,t_0} - p_{r,t_0} + \frac{2R_{\mathrm{s}} T_a z_a (\tau(t_1) - \tau(t_0))}{L_a A_a} (q_{r,a,t_1} - q_{\ell,a,t_1}) = 0 \tag{2a}$$

$$p_{r,t_1} - p_{\ell,t_1} + \underbrace{\frac{\lambda_a R_{\mathrm{s}} T_a z_a L_a}{4A_a^2 D_a} \left( \frac{|q_{\ell,a,t_1}| q_{\ell,a,t_1}}{p_{\ell,t_1}} + \frac{|q_{r,a,t_1}| q_{r,a,t_1}}{p_{r,t_1}} \right)}_{\text{friction term}} + \frac{g s_a L_a}{2R_{\mathrm{s}} T_a z_a} (p_{\ell,t_1} + p_{r,t_1}) = 0. \tag{2b}$$

The first equation (2a) is called the *Continuity* equation, while the second equation (2b) is known as the *Momentum* equation and contains the non-linear *friction term*. In the equations, the specific gas constant is given by $R_{\mathrm{s}}$, the gravitational acceleration by $g$, the gas temperature of the pipe by $T_a$, its compressibility factor by $z_a$, its length by $L_a$, its diameter by $D_a$, its area by $A_a$ defined as $A_a = D_a^2 \pi/4$, its Darcy friction coefficient by $\lambda_a$, and finally its slope by $s_a$ defined as $s_a = (h_r - h_\ell)/L_a$ using the height or altitude $h$ of the pipes' end nodes. The gas temperature depends on the individual pipe, as we determine it based on the given gas temperatures $T_{v,0}$ in the initial state for each node $v$ as

$$T_a = \frac{T_{\ell,0} + T_{r,0}}{2}.$$

In a similar fashion, we determine the pipes' compressibility factor using the formula of Pápay [Pápay, 1968; Takacs, 1989] based on the initial pressure and temperature of each node as

$$z_a = 0.5(z(p_{\ell,0}, T_{\ell,0}) + z(p_{r,0}, T_{r,0})).$$

Finally, the Darcy friction factor is determined using the formula of Nikuradse [Nikuradse, 1950], which is an explicit approximation depending only on the pipe's diameter and its integral roughness $k_a$.

## 2.3 Valves

Valves are active elements that are able to dynamically connect or disconnect two network nodes. Their state is represented by a binary mode variable $m^{\text{op}}_{a,t}$, which specifies if the valve $a$ is open ($m^{\text{op}}_{a,t} = 1$) at time $t$, connecting both end nodes and forcing their pressures variables to attain identical values, or closed ($m^{\text{op}}_{a,t} = 0$), disconnecting the nodes, decoupling the respective pressure values, and prohibiting gas flow exchange. Thus, the corresponding model for a valve $a = (\ell, r) \in \mathcal{A}^{\text{va}}$ reads

$$\begin{aligned}
m^{\text{op}}_{a,t} = 1 \quad &\Longrightarrow \quad p_{\ell,t} = p_{r,t} && \forall t \in \mathcal{T} && \text{(3a)}\\
m^{\text{op}}_{a,t} = 0 \quad &\Longrightarrow \quad q_{a,t} = 0 && \forall t \in \mathcal{T} && \text{(3b)}\\
m^{\text{op}}_{a,t} &\in \{0,1\} && \forall t \in \mathcal{T}_0.
\end{aligned}$$

## 2.4 Regulators

Regulators are active elements that can be seen as an extension of valves. In addition to changing the network's connectivity, they can reduce the pressure of the gas in the direction of the flow. However, this is only possible along the arc's orientation. We use the regulator model of [Hennings et al., 2021a] here, in which the state of a regulator $a$ at time $t$ is modeled by a set of three different binary mode variables: The two valve state variables of an open regulator $m^{\text{op}}_{a,t}$ and a closed regulator $m^{\text{cl}}_{a,t}$ as well as the active mode $m^{\text{ac}}_{a,t}$, which allows for pressure reduction. The model for all $a = (\ell, r) \in \mathcal{A}^{\text{rg}}$ can be stated as

$$\begin{aligned}
1 = m^{\text{ac}}_{a,t} + m^{\text{op}}_{a,t} + m^{\text{cl}}_{a,t} & && \forall t \in \mathcal{T} && \text{(4a)}\\
m^{\text{op}}_{a,t} = 1 \quad &\Longrightarrow \quad p_{\ell,t} \le p_{r,t} && \forall t \in \mathcal{T} && \text{(4b)}\\
m^{\text{ac}}_{a,t} + m^{\text{op}}_{a,t} = 1 \quad &\Longrightarrow \quad p_{\ell,t} \ge p_{r,t} && \forall t \in \mathcal{T} && \text{(4c)}\\
m^{\text{cl}}_{a,t} = 1 \quad &\Longrightarrow \quad q_{a,t} \le 0 && \forall t \in \mathcal{T} && \text{(4d)}\\
q_{a,t} &\ge 0 && \forall t \in \mathcal{T} && \text{(4e)}\\
m^{\text{op}}_{a,t}, m^{\text{cl}}_{a,t}, m^{\text{ac}}_{a,t} &\in \{0,1\} && \forall t \in \mathcal{T}_0.
\end{aligned}$$

Note that in the indicator condition of Equation (4c), the term $m^{\text{ac}}_a + m^{\text{op}}_a$ acts as a binary variable, as Equation (4a) forces the choice of exactly one of the three modes. Furthermore, we assume an internal check valve in the element, which prevents flow against the orientation even in the case of an open regulator. For a more detailed description of the internal regulator behavior, we refer to [Hennings et al., 2021b].

## 2.5 Artificial station arcs

As stated above, the original network topology inside the stations is replaced by artificial arcs of different types, which represent an approximation of the station's gas transport capabilities. Each arc can be active or inactive, which we model by a binary variable

$$x^{\text{arc}}_{a,t} \in \{0,1\} \qquad \forall a \in \mathcal{A}^{\text{ar}} \quad \forall t \in \mathcal{T}_0,$$

where $x^{\text{arc}}_{a,t} = 1$ represents the active state. An inactive arc behaves like a closed valve, i.e., the flow is zero and the end nodes' pressures are decoupled, while the active behavior is specific to each artificial arc type, which we discuss in the following.

### 2.5.1 Shortcuts

Shortcuts connect its end nodes if active. Hence, their model is equivalent to that of a valve and can be stated for a shortcut $a = (\ell, r) \in \mathcal{A}^{\mathrm{arSc}}$ and time $t \in \mathcal{T}$ as:

$$x^{\mathrm{arc}}_{a,t} = 1 \quad \Longrightarrow \quad p_{\ell,t} = p_{r,t} \tag{5a}$$
$$x^{\mathrm{arc}}_{a,t} = 0 \quad \Longrightarrow \quad q_{a,t} = 0. \tag{5b}$$

### 2.5.2 Regulating arcs

Artificial regulating arcs capture the general behavior of regulators. Hence, they only allow flow along their orientation and offer the possibility of pressure reduction if they are active. The corresponding model for a regulating arc $a = (\ell, r) \in \mathcal{A}^{\mathrm{arRg}}$ and time $t \in \mathcal{T}$ can be stated as

$$x^{\mathrm{arc}}_{a,t} = 1 \quad \Longrightarrow \quad p_{\ell,t} \geq p_{r,t} \tag{6a}$$
$$x^{\mathrm{arc}}_{a,t} = 0 \quad \Longrightarrow \quad q_{a,t} \leq 0 \tag{6b}$$
$$q_{a,t} \geq 0. \tag{6c}$$

Note that the Equations (4) do not apply here.

### 2.5.3 Bi-regulating arcs

To represent the behavior of a pair of anti-parallel regulators in the original network topology, we use a bi-regulating arc. We can model its behavior by choosing an orientation of the element in active mode and applying the corresponding constraints of a regulating arc for each direction separately. The model for a bi-regulating arc $a = (\ell, r) \in \mathcal{A}^{\mathrm{arBiRg}}$ and time $t \in \mathcal{T}$ is given as

$$x^{\mathrm{arc}}_{a,t} = x^{\mathrm{fwd}}_{a,t} + x^{\mathrm{bwd}}_{a,t} \tag{7a}$$
$$q_{a,t} = q^{\mathrm{fwd}}_{a,t} - q^{\mathrm{bwd}}_{a,t} \tag{7b}$$
$$x^{\mathrm{fwd}}_{a,t} = 1 \quad \Longrightarrow \quad p_{\ell,t} \geq p_{r,t} \tag{7c}$$
$$x^{\mathrm{fwd}}_{a,t} = 0 \quad \Longrightarrow \quad q^{\mathrm{fwd}}_{a,t} \leq 0 \tag{7d}$$
$$q^{\mathrm{fwd}}_{a,t} \geq 0 \tag{7e}$$
$$x^{\mathrm{bwd}}_{a,t} = 1 \quad \Longrightarrow \quad p_{r,t} \geq p_{\ell,t} \tag{7f}$$
$$x^{\mathrm{bwd}}_{a,t} = 0 \quad \Longrightarrow \quad q^{\mathrm{bwd}}_{a,t} \leq 0 \tag{7g}$$
$$q^{\mathrm{bwd}}_{a,t} \geq 0 \tag{7h}$$
$$x^{\mathrm{fwd}}_{a,t}, x^{\mathrm{bwd}}_{a,t} \in \{0, 1\}.$$

Here, $x^{\mathrm{fwd}}_{a,t}$ represents the choice for the *forward orientation* for arc $a$ and time $t$, in which flow and regulation happen along the arc's orientation, while $x^{\mathrm{bwd}}_{a,t}$ represents the choice of the *backward orientation*, in which the orientation is flipped. In the same fashion, $q^{\mathrm{fwd}}_{a,t}$ represents the forward flow while $q^{\mathrm{bwd}}_{a,t}$ represents the flow in the backward direction.

### 2.5.4 Compressor arcs

Finally, there are the compressor arcs, which are the only elements able to increase the gas pressure in the direction of the flow. They approximate the capabilities of compressor station arcs in the original network station topology. For this reason, we shortly introduce these in the following.

Each compressor station, which we assume to be modeled as an arc $a = (\ell, r)$ as well, represents a combination of single compressor units, which again are a combination of a compressor machine and a corresponding drive that provides the necessary power. The compressor machine of a compressor unit $u$ can operate in its feasible operating range, defined as a polytope in the space $(Q, \psi)$ of the volumetric flow $Q$ and the compression ratio $\psi$ given as

$$p = \rho R_{\mathrm{s}} T_a z_a \tag{8a}$$
$$Q = \frac{q}{\rho_\ell} \quad \Longrightarrow \quad Q = q \frac{R_{\mathrm{s}} T_a z_a}{p_\ell} \tag{8b}$$
$$\psi = \frac{p_r}{p_\ell}. \tag{8c}$$

Here, $\rho$ denotes the density of the gas, $\rho_\ell$ the incoming gas density, while $z_a$ and $T_a$ are defined in the same way as they were for pipes in Section 2.2. Equation (8a) is called the equation of state for real gases [Osiadacz, 1996; Fügenschuh et al., 2015].

Furthermore, the compression is restricted by the maximal amount of power $\bar{P}_u$ the corresponding drive can provide. The power needed for compressing a mass flow value of $q$ from $p_\ell$ to $p_r$, i.e., with $\psi = p_r/p_\ell$, is given as

$$P(q, p_\ell, p_r) = \frac{q}{\eta_{\mathrm{ad}}} R_{\mathrm{s}} T_a z_a \frac{\kappa}{\kappa - 1} \left[ \psi^{\frac{\kappa-1}{\kappa}} - 1 \right]. \tag{9}$$

Here, $\eta_{\mathrm{ad}}$ is the adiabatic efficiency and is assumed to be a constant given per compressor unit. Further, $\kappa$ is the isentropic exponent, which we approximate by the constant value of 1.296, see, for example, [Fügenschuh et al., 2015] for more details. The single units can then be combined to predefined configurations $\mathcal{C}_a$ of the compressor station $a$, defined as serial connections of parallel unit arrangements. For a more in-depth description, we refer to [Fügenschuh et al., 2015; Walther and Hiller, 2017; Hennings et al., 2021a].

In the base model of [Hoppmann-Baum et al., 2021], the artificial pressure increasing arcs were modeled as a single compressor representing the combination of single units, which could dynamically be assigned to pressure-increasing arcs using binary assignment variables. As a consequence of this dynamic combination of different units, some of the constraints describing the resulting compression capabilities were approximated quite generously to keep the overall model linear.

To avoid this problem, we develop a novel formulation, which is closer to the original compressor station model while the model size with respect to the number of variables and constraints is not increased. Like for the compressor stations, we create for each artificial compressor arc $a = (\ell, r) \in \mathcal{A}^{\mathrm{arCp}}$ a set of configurations $\mathcal{C}_a^{\mathrm{ar}}$ to choose from. These form different compression settings based on how the compressor units can be composed in the underlying compressor stations. The feasible operating range of a chosen configuration $c \in \mathcal{C}_a^{\mathrm{ar}}$ of an artificial compressor arc $a$ is restricted in terms of three parameters: The maximum compression ratio $\psi_c$, the maximum volumetric flow $Q_c$, and the maximum compression power $P_c$. Furthermore, each configuration has an associated number $u_c$ of used compressor units. From the parameters restricting the feasible operating range, we further find parameters $\alpha_c^{\mathrm{p}_\ell}$, $\alpha_c^{\mathrm{p_r}}$, and $\alpha_c^{\mathrm{q}}$, which represent a linearization of the compression power of the form

$$P \approx \alpha_c^{\mathrm{p}_\ell} p_\ell + \alpha_c^{\mathrm{p_r}} p_r + \alpha_c^{\mathrm{q}} q,$$

and are determined by a linear regression based on triples $(p_\ell^s, p_r^s, q^s)_{s \in \mathbb{N}}$ sampled from within the intersection of the corresponding variable bound intervals over time, which satisfy the constraints

$$q^s \frac{R_{\mathrm{s}} T_a z_a}{p_\ell^s} \le Q_c \qquad \frac{p_r^s}{p_\ell^s} \le \psi_c \qquad P(q^s, p_\ell^s, p_r^s) = P_c.$$

Note that usually, $\alpha_c^{\mathrm{p}_\ell} < 0$, $\alpha_c^{\mathrm{p_r}} > 0$, and $\alpha_c^{\mathrm{q}} > 0$ hold.

The final component of the configuration-based model is the set of artificial compressor configuration conflicts $\mathcal{Z}$. Each conflict $z$ is defined as tuple $z = (c_1, c_2)$ of configurations of different artificial compressors from the same network station that cannot be used simultaneously. Conflicts are needed to model, for example, compressor units that can be used in different compressor stations, but not at the same time. As a consequence, configurations $c_1$ and $c_2$ of two corresponding artificial compressors cannot be used simultaneously if both can use the same unit. Since conflicts are always defined for configurations of artificial compressors in the same network station, we can partition the conflicts per network station as $\mathcal{Z} = \dot{\bigcup}_{i \in \mathcal{I}} \mathcal{Z}_i$.

Given this description, we state the configuration-based model for an artificial compressor $a = (\ell, r) \in \mathcal{A}^{\mathrm{arCp}}$ and time $t \in \mathcal{T}$ as

$$x_{a,t}^{\mathrm{arc}} = \sum_{c \in \mathcal{C}_a^{\mathrm{ar}}} x_{c,t}^{\mathrm{cfg}} \tag{10a}$$

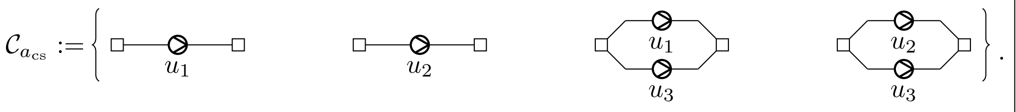
$$x_{a,t}^{\mathrm{arc}} = 1 \implies p_{\ell,t} \le p_{r,t} \tag{10b}$$

$$x_{a,t}^{\mathrm{arc}} = 0 \implies q_{a,t} \le 0 \tag{10c}$$

$$q_{a,t} \ge 0, \tag{10d}$$

where we additionally demand for each configuration $c \in \mathcal{C}_a^{\mathrm{ar}}$ of an artificial compressor $a = (\ell, r) \in \mathcal{A}^{\mathrm{arCp}}$

Assume a artificial compressor $a_{\mathrm{ac}}$ with a single associated compressor station $a_{\mathrm{cs}}$. Let the configurations of $a_{\mathrm{cs}}$ be defined as



Then the corresponding artificial configurations for $a_{\mathrm{ac}}$ are

$$\mathcal{C}^{\mathrm{ar}}_{a_{\mathrm{ac}}} := \left\{ \begin{array}{c} u^p \; u^s \quad U \\ (1, 1, \{u_1, u_2\}), \\ (2, 1, \{u_1, u_3\}), \\ (2, 1, \{u_2, u_3\}) \end{array} \right\}.$$

Figure 1: Example of the construction of the configurations for an artificial compressor $a_{\mathrm{ac}}$. The associated compressor station $a_{\mathrm{cs}}$ has four different configurations using three different compressor units: Two configurations using only a single unit, either unit $u_1$ or unit $u_2$, and two configurations in which each of the two units is used in parallel to unit $u_3$. There are three resulting artificial configurations for $a_{\mathrm{ac}}$, which all allow only one serial stage: One configuration with only one allowed parallel unit, in which $u_1$ or $u_2$ can be chosen, and two configurations allowing for two parallel units, one with the units set $\{u_1, u_3\}$ and one with the set $\{u_2, u_3\}$. While we can combine the first two original configurations into one artificial configuration, this is not possible for the parallel configurations. The corresponding unit set would have to be $\{u_1, u_2, u_3\}$, which is not feasible as there is no original configuration combining $u_1$ and $u_2$ in parallel.

that

$$x^{\mathrm{cfg}}_{c,t} = 1 \implies p_{r,t} \le p_{\ell,t}\psi_c \qquad \forall t \in \mathcal{T} \tag{11a}$$

$$x^{\mathrm{cfg}}_{c,t} = 1 \implies q_{a,t} \le p_{\ell,t}\frac{Q_c}{R_{\mathrm{s}}T_a z_a} \qquad \forall t \in \mathcal{T} \tag{11b}$$

$$x^{\mathrm{cfg}}_{c,t} = 1 \implies \alpha^{\mathrm{p}\ell}_c p_{\ell,t} + \alpha^{\mathrm{pr}}_c p_{r,t} + \alpha^{\mathrm{q}}_c q_{a,t} \le P_c \qquad \forall t \in \mathcal{T} \tag{11c}$$

$$x^{\mathrm{cfg}}_{c,t} \in \{0,1\} \qquad \forall t \in \mathcal{T}_0.$$

Here, the binary variable $x^{\mathrm{cfg}}_{c,t}$ represents the decision to use (value=1) or not to use (value=0) the corresponding configuration $c \in \mathcal{C}^{\mathrm{ar}}_a$ at time $t$. In addition, we need the following constraint for each conflict $z = (c_1, c_2) \in \mathcal{Z}$ and time $t \in \mathcal{T}$:

$$x^{\mathrm{cfg}}_{c_1,t} + x^{\mathrm{cfg}}_{c_2,t} \le 1. \tag{12}$$

Note that the artificial compressor arcs are directed and therefore can only have positive flow as well as compress gas from the end node $\ell$ towards $r$, which is ensured via Equations (10d) and (10b).

**Artificial configuration set construction** Since the artificial compressor configurations are not given as part of the input, we create them for each artificial compressor $a \in \mathcal{A}^{\mathrm{arCp}}$ based on the original compressor stations represented by it. Here, we associate each artificial configuration $c$ with a tuple $(u^p_c, u^s_c, U_c)$, where $u^p_c$ is the number of parallel compressor units to use, $u^s_c$ is the number of serial units to use, and $U_c$ is the set of usable units. We manually create these tuples based on the configurations of the original compressor stations as well as their topological position in the network. If multiple original compressor compositions use similar units in the same layout, we represent them by the same artificial configuration if possible. An example of the construction is given in Figure 1.

Given $(u^p_c, u^s_c, U_c)$ for each artificial configuration $c$, we now derive the corresponding parameters for the number of simultaneously used compressor units $u_c$, the maximum compression ratio $\psi_c$, the maximum volumetric flow $Q_c$, and the maximum power $P_c$. First, we present a procedure for a configuration $c$ for the special case of either $u^p_c$ or $u^s_c$ being equal to 1, i.e., for the case of a pure serial or pure parallel compression. For each compressor unit $u \in U_c$, we are given the maximum power $\bar{P}_u$ its associated drive can provide. Furthermore, we can determine its maximum compression ratio $\bar{\psi}_u$ and maximum volumetric flow $\bar{Q}_u$ by iterating the extreme points of its feasible operating range polytope. Let then $(\tilde{\psi}_n)_{n=1}^{|U_c|}$ be the

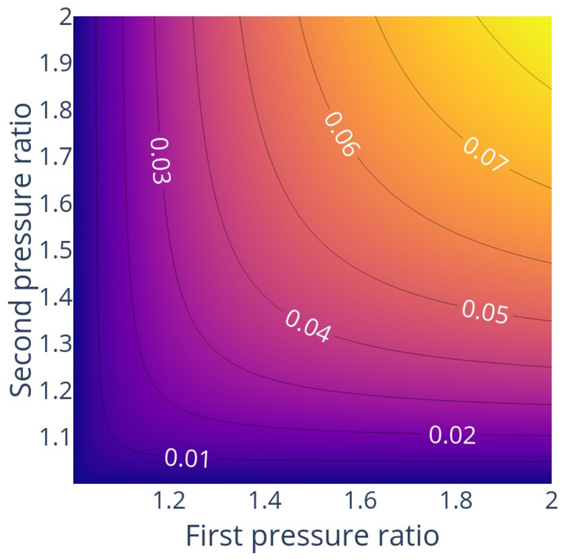


Figure 2: Relative error in compression power. We compare the power $P_{\text{sum}}$ needed for a serial compression using two separate compression processes with ratios $\psi_1$ and $\psi_2$ to the power $P_{\text{one}}$ needed by our approximation of using a single compression process with ratio $\psi_1 \cdot \psi_2$. We plotted the relative error $(P_{\text{one}} - P_{\text{sum}})/P_{\text{sum}}$ for compression ratios $\psi_1$ and $\psi_2$ up to 2.0. Figure created with [Plotly Technologies Inc., 2015].

sequence of maximum compression ratios $\bar{\psi}_u$ for all $u \in U_c$ sorted in descending order, and analogously let $(\tilde{Q}_n)_{n=1}^{|U_c|}$ and $(\tilde{P}_n)_{n=1}^{|U_c|}$ be the corresponding descending sequences of maximum volumetric flows $\bar{Q}_u$ and power values $\bar{P}_u$. We then define the parameters $u_c$, $\psi_c$, $Q_c$, and $P_c$ for $c \in \mathcal{C}_a^{\text{ar}}$ of $a \in \mathcal{A}^{\text{arCp}}$ as

$$u_c = u_c^p \cdot u_c^s \tag{13a}$$

$$\psi_c = \begin{cases} \prod_{i=1}^{u_c^s} \tilde{\psi}_i & \text{if } u_c^s > 1 \\ \tilde{\psi}_{u_c^p} & \text{if } u_c^s = 1, \end{cases} \tag{13b}$$

$$Q_c = \begin{cases} \sum_{i=1}^{u_c^p} \tilde{Q}_i & \text{if } u_c^p > 1 \\ \tilde{Q}_{u_c^s} & \text{if } u_c^p = 1, \end{cases} \tag{13c}$$

$$P_c = \sum_{i=1}^{u_c} \tilde{P}_i. \tag{13d}$$

Note that for the serial compression, summing up the power values is not quite accurate and slightly underestimates the necessary compression power when comparing the original two serial compression processes with one combined compression process using the product of the single compression ratios, see Figure 2. As a consequence, $P_c$ is potentially too small to allow for all compression processes possible in the original network. However, in the data used for our computational experiments in Section 4, the maximum serial compression number $u_c^s$ for a configuration $c$ is 2. Furthermore, all involved units have compressor ratios smaller than 2.0. Thus, the approximation error is guaranteed to be smaller than 9%.

If a configuration features multiple serial and multiple parallel stages, the above-presented formulas for creating the configuration parameters cannot be applied directly. Instead, we create for each such arrangement and each concrete assignment of a compressor unit to a position in the arrangement a separate artificial configuration for the corresponding artificial compressor arc. The total number of used compressor units $u_c$ is then equal to the number of active units in each specific arrangement, and the maximum power $P_c$ is determined as the sum of maximum power values of these units. To determine the parameters $\psi_c$ and $Q_c$, we recursively apply the formulas used in Equations (13b) and (13c) for each of the concrete serial and parallel sub-compositions of units. The approach of creating one configuration for each possible unit arrangement can, in theory, result in a lot of artificial configurations. However, in our experience, this is not the case in practice, as such configurations do not occur very often.

**Model comparison** The main benefit of the new artificial compressor model in Equations (10), (11), and (12) compared to the base model is that each compressor unit contributes either in serial or in parallel to the compression capabilities and does not use the benefits both at the same time.

Furthermore, we improved the description of the maximum compression ratio $\psi$ and the maximum volumetric flow $Q$. For both, we do not the need to replace the incoming pressure of the arc $p_\ell$ by a constant value, which was done explicitly in Equation (40) in [Hoppmann-Baum et al., 2021] and implicitly by assuming a maximum mass flow bound per compressor unit, which can only be derived from the inherent volumetric flow bound by assuming a constant incoming pressure. Furthermore, we can correctly employ the product of compressor ratios in the computation of the maximum usable compression ratio of serial compression processes in Equation (13b) instead of the linear approximation formula used in (35) in [Hoppmann-Baum et al., 2021].

As an example, we show the model improvements in comparison to [Hoppmann-Baum et al., 2021] for a single artificial compressor arc in Figure 3, where we compare the feasible region polytopes produced by the base model, our new formulation, and the actual feasible region of the configuration polytopes of the compressor station in the original station topology. To have a clean comparison, we choose an artificial compressor arc representing exactly one compressor station. This compressor station features three compressor units, from which two units can be combined in parallel, but has no serial unit combination capabilities. Note that we excluded the power bound from the feasible region polytopes since it is modeled for both models in the same way, i.e., as a linear approximation using a linear regression approach.

We clearly see the increased accuracy of the new model already for an artificial compressor arc, which is only able to combine two compressor units. When combining more compressor units, the differences would further increase.

## 2.6 Station simple states

The stations' artificial arcs do not operate independently from each other. Instead, each station $i \in \mathcal{I}$ is always in exactly one *simple state*, where the set of possible simple states of the station is given as $\mathcal{S}_i$, and the union of simple states over all stations is given as $\mathcal{S} = \bigcup_{i \in \mathcal{I}} \mathcal{S}_i$. Each simple state $s \in \mathcal{S}_i$ restricts the usage of the artificial arcs by defining a set of arcs $\mathcal{A}^{\text{on}}_s \subseteq \mathcal{A}^{\text{ar}}_i$, which need to be active if the station uses state $s$, and a set of arcs $\mathcal{A}^{\text{off}}_s \subseteq \mathcal{A}^{\text{ar}}_i$, which cannot be active if the station uses state $s$. For all other artificial arcs $\mathcal{A}^{\text{ar}}_i \setminus (\mathcal{A}^{\text{on}}_s \cup \mathcal{A}^{\text{off}}_s)$, the activity is optional. We represent the choice for a simple state $s$ at time $t$ by the binary variable $x^{\text{st}}_{s,t}$, where a value of 1 represents an active simple state. Using this, we formulate the simple state model for each network station $i \in \mathcal{I}$ as

$$\sum_{s \in \mathcal{S}_i} x^{\text{st}}_{s,t} = 1 \qquad \forall t \in \mathcal{T} \tag{14}$$

$$x^{\text{st}}_{s,t} \in \{0,1\} \qquad \forall s \in \mathcal{S}_i \quad \forall t \in \mathcal{T}_0$$

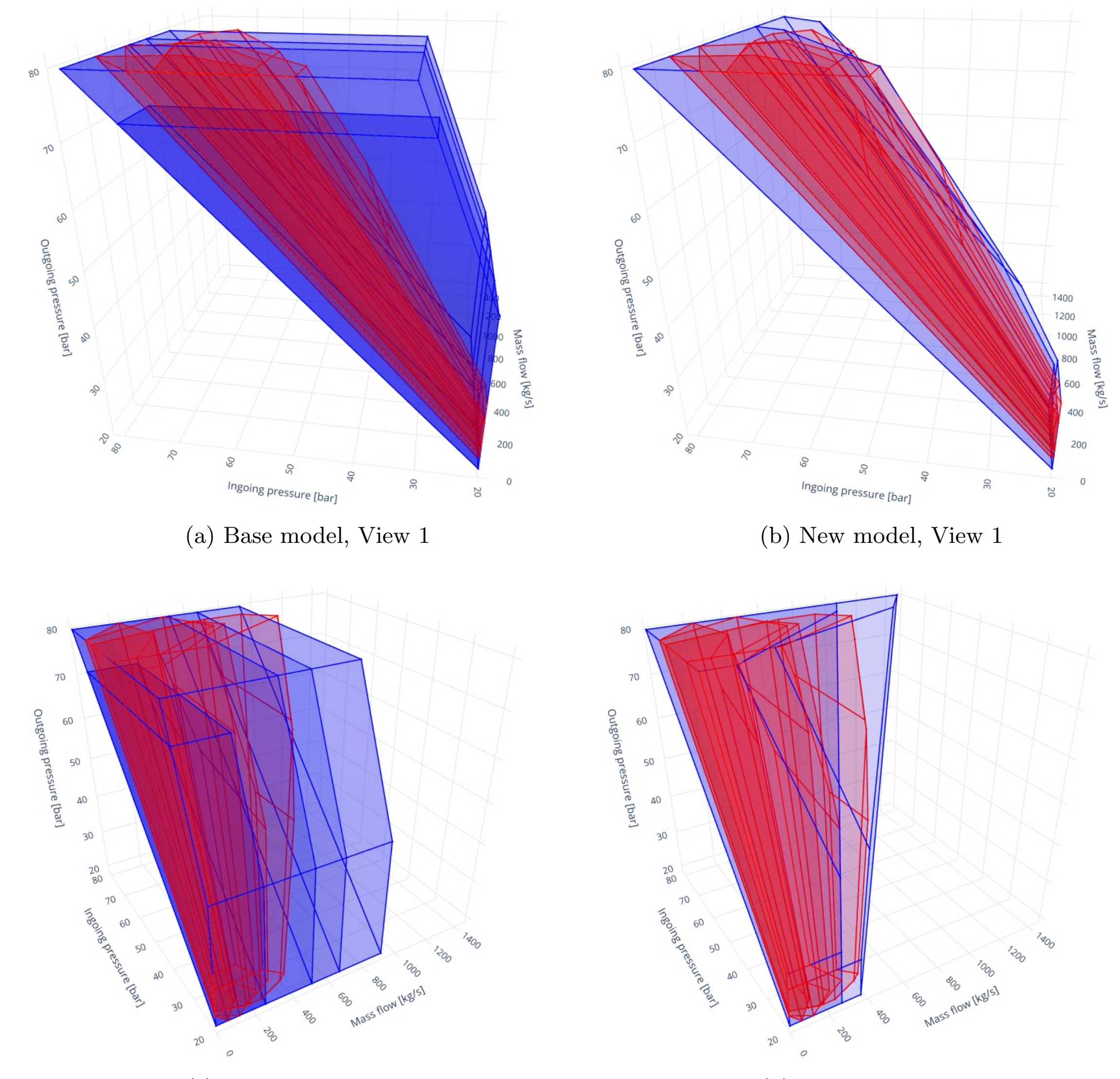
and for all artificial arcs $a \in \mathcal{A}^{\text{ar}}_i$ of each network station $i \in \mathcal{I}$ and time $t \in \mathcal{T}$ as

$$\sum_{s \in \mathcal{S}_i : a \in \mathcal{A}^{\text{on}}_s} x^{\text{st}}_{s,t} \le x^{\text{arc}}_{a,t} \qquad 1 - \sum_{s \in \mathcal{S}_i : a \in \mathcal{A}^{\text{off}}_s} x^{\text{st}}_{s,t} \ge x^{\text{arc}}_{a,t}. \tag{15}$$

## 2.7 Station flow directions and fence node valves

Apart from the simple state, each station is also assigned exactly one *flow direction* $f$ at each point in time $t \in \mathcal{T}_0$, captured by the binary variable $x^{\text{fd}}_{f,t}$, where the value 1 again stands for an active flow direction. The set of possible flow directions for station $i$ is given as $\mathcal{F}_i$. A flow direction $f \in \mathcal{F}_i$ determines the connection between the interior and exterior of each station by defining two sets of fence nodes: Those fence nodes $\mathcal{V}^{\text{in}}_f \subseteq \mathcal{V}^{\text{fn}}_i$ allowing for flow into the station and those fence nodes $\mathcal{V}^{\text{out}}_f \subseteq \mathcal{V}^{\text{fn}}_i$ allowing for flow out of the station. All other fence nodes $\mathcal{V}^{\text{fn}}_i \setminus (\mathcal{V}^{\text{in}}_f \cup \mathcal{V}^{\text{out}}_f)$ do not allow any exchange of flow between the station's interior and exterior.

As an extension to the model given in [Hoppmann-Baum et al., 2021], we also decouple the pressure at those fence nodes without flow exchange. To do that, we adopt an idea initially mentioned in [Hoppmann-Baum, 2022]: We create a copy $v'$ of each fence node $v$, create a valve $a'$ in between the two nodes, and reconnect each artificial arc that was connected to the fence node $v$ to its copy $v'$ instead. The construction is also visualized in Figure 4. Note that we add the copy $v'$ to $\mathcal{V}$ and the new valve $a$ to $\mathcal{A}^{\text{va}}$. Thereby, we create all the corresponding variables and constraints. We further denote by $a^{\text{vaFn}}_v$ the *fence node*



(a) Base model, View 1

(b) New model, View 1

(c) Base model, View 2

(d) New model, View 2

Figure 3: Feasible region comparison of the artificial compressor base model against our new one at the example of one artificial compressor arc. As a reference solution, we plotted the feasible region polytopes of the configurations of the associated compressor station from the original network topology in red. The polytopes for the artificial compressor arc are plotted in blue, where Figures (a) and (c) show the base model polytopes and Figures (b) and (d) show the new model polytopes. For the new model, each polytope represents one artificial configuration to choose from. Since the base model does not feature configurations, we plotted one polytope for each feasible compressor unit combination on the artificial arc. The pressure bound interval used for both end nodes is [20 bar, 80 bar]. For the constant incoming pressure needed for the base model, we use the median value of 50 bar. For a better impression of the three-dimensional polytopes, we show them from two different camera positions in View 1, visible in Figures (a) and (b), and View 2, visible in Figures (c) and (d). Figures created with [Plotly Technologies Inc., 2015].

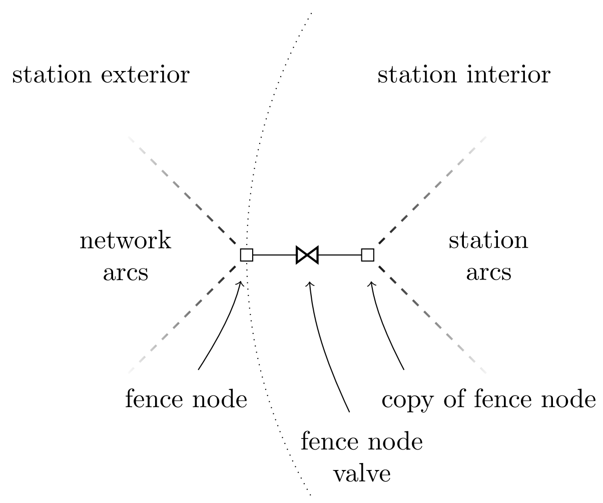


Figure 4: Schematic fence node valve construction. We create a copy of the fence node and add a new valve between the fence node and its copy, the fence node valve of the node. The artificial station arcs that have been connected to the fence node are instead now connected to the copied node.

*valve* of fence node $v$, by $\mathcal{A}^{\text{vaFn}} \subseteq \mathcal{A}^{\text{va}}$ the set of all fence node valves contained in the network, and by $\mathcal{A}^{\text{vaOr}} = \mathcal{A}^{\text{va}} \setminus \mathcal{A}^{\text{vaFn}}$ the set of original non-fence-node valves of the network.

In addition to the flow restrictions, some fence nodes $v$ have an upper pressure bound $\bar{p}_v^{\text{exit}}$ in case flow leaves the station over $v$ for some flow direction, i.e. the fence node serves as an exit of the station. For all other fence nodes $v$, we set $\bar{p}_v^{\text{exit}} = \bar{p}_v$.

Finally, not all flow directions fit to all simple states. Hence, we are given the set $\mathcal{Q} \subset \mathcal{F} \times \mathcal{S}$ of suitable combinations of flow directions and simple states.

For the model regarding flow directions and fence node valves, we demand for all network stations $i \in \mathcal{I}$ and all times $t \in \mathcal{T}$ that

$$\sum_{f \in \mathcal{F}_i} x_{f,t}^{\text{fd}} = 1 \tag{16}$$
$$x_{f,t}^{\text{fd}} \in \{0,1\} \qquad \forall f \in \mathcal{F}_i,$$

for all simple states $s \in \mathcal{S}_i$ of all stations $i \in \mathcal{I}$ and all times $t \in \mathcal{T}$ that

$$\sum_{(f,s) \in \mathcal{Q}} x_{f,t}^{\text{fd}} \geq x_{s,t}^{\text{st}}, \tag{17}$$

and finally, for all fence nodes $v \in \mathcal{V}_i^{\text{fn}}$ of all stations $i \in \mathcal{I}$ and all times $t \in \mathcal{T}$ that

$$\sum_{f \in \mathcal{F}_i : v \in (\mathcal{V}_f^{\text{in}} \cup \mathcal{V}_f^{\text{out}})} x_{f,t}^{\text{fd}} = m_{a_v^{\text{vaFn}},t}^{\text{op}} \tag{18a}$$
$$\sum_{f \in \mathcal{F}_i : v \in \mathcal{V}_f^{\text{in}}} x_{f,t}^{\text{fd}} = 1 \implies q_{a_v^{\text{vaFn}},t} \geq 0 \tag{18b}$$
$$\sum_{f \in \mathcal{F}_i : v \in \mathcal{V}_f^{\text{out}}} x_{f,t}^{\text{fd}} = 1 \implies q_{a_v^{\text{vaFn}},t} \leq 0 \tag{18c}$$
$$\sum_{f \in \mathcal{F}_i : v \in \mathcal{V}_f^{\text{out}}} x_{f,t}^{\text{fd}} = 1, \implies p_{v,t} \leq \bar{p}_v^{\text{exit}}. \tag{18d}$$

## 2.8 Demand scenario

At each boundary node, we are given future requirements in terms of inflow and pressure, which we call the *demands*. A distinct set of demand values for all boundary nodes of the network is called a *demand scenario* of the problem. While for each future time step $t \in \mathcal{T}_0$, there is a required inflow value $\hat{d}_{v,t}$ for each boundary node $v \in \mathcal{V}^{\mathrm{b}}$, the prescribed pressure $\hat{p}_{v,t}$ only exists for entries $v \in \mathcal{V}^{+}$ with non-zero inflow values at time $t$. Furthermore, we do not have to meet them exactly but within a tolerance of $\varepsilon = 1\,\mathrm{bar}$ for each value.

However, there is no guarantee that meeting the demands is actually possible. For these cases, we allow a deviation from the demands, which we measure and penalize in the objective function. We call this deviation *slack* and capture it in the variables $\sigma^{d+}_{v,t}$ and $\sigma^{d-}_{v,t}$ for the positive and negative inflow deviation and $\sigma^{p+}_{v,t}$ and $\sigma^{p-}_{v,t}$ for the positive and negative pressure deviation for a boundary node $v \in \mathcal{V}^{\mathrm{b}}$ and time $t \in \mathcal{T}$. When denoting by $\bar{\sigma}^{d}$ the maximum amount of inflow slack per boundary node and time step as well as by $\bar{\sigma}^{p}$ the corresponding maximum amount of pressure slack, we can formulate the model of the inflow demands for all boundary nodes $v \in \mathcal{V}^{\mathrm{b}}$ and a future time $t \in \mathcal{T}$ as

$$d_{v,t} = \hat{d}_{v,t} + \sigma^{d+}_{v,t} - \sigma^{d-}_{v,t} \tag{19a}$$

$$0 \leq \sigma^{d+}_{v,t} \leq \bar{\sigma}^{d} \tag{19b}$$

$$0 \leq \sigma^{d-}_{v,t} \leq \bar{\sigma}^{d}, \tag{19c}$$

and the model of the pressure demands for all entry nodes $v \in \mathcal{V}^{+}$ with non-zero inflow demands $\hat{d}_{v,t}$ and a future time $t \in \mathcal{T}$ as

$$p_{v,t} \leq \hat{p}_{v,t} + \varepsilon + \sigma^{p+}_{v,t} \tag{20a}$$

$$p_{v,t} \geq \hat{p}_{v,t} - \varepsilon - \sigma^{p-}_{v,t} \tag{20b}$$

$$0 \leq \sigma^{p+}_{v,t} \leq \bar{\sigma}^{p} \tag{20c}$$

$$0 \leq \sigma^{p-}_{v,t} \leq \bar{\sigma}^{p}. \tag{20d}$$

## 2.9 Initial state of artificial elements

As stated at the end of Section 2.1, we are given an initial state for the network, specifying the pressures, flows, and modes of all nodes and elements of the original network. However, since the network station model is based on an artificial topology, there are no initial state values for the non-fence nodes inside the station, the artificial arcs, the station's simple state, and its flow direction.

For the inner station nodes, we do not need the initial pressure values as variables in the model. However, the initial pressures and temperatures are used as parameters in the artificial compressor model. For these values, we use the corresponding average values per quantity over the set of fence nodes. As the only exception, we use the initial quantities given for the original fence node for those nodes being fence node copies. Regarding the flow values, we do not use the initial flow values of artificial arcs. However, for the objective function, we need the initial flow on the fence node valve (see Section 2.10), which can be determined by using the flow balance equation (1) of the corresponding fence node.

Regarding the discrete states of the network stations as well as their contained artificial arcs, we aim to choose settings that fit the initial state of the fence nodes, fence node valves, and the compressor stations in the original network station topology. Since there might be multiple suitable settings, we only sort out non-fitting ones and include the final decision into the optimization model.

First, we determine a list of possible initial simple states $\mathcal{S}^{0}$. This is done by checking for each simple state of each network station if there would be a corresponding feasible initial state for the flow direction choice, as well as all the elements in the network station which fits to the initial pressure and flow values of the fence nodes and corresponding fence node valve. As a second step, we determine the set of initially running compressor units $\mathcal{U}^{0}$ based on the active configurations of the compressors stations in the original network station topology. Finally, we determine the set $\mathcal{C}^{\mathrm{ar}0}$ of those configurations of artificial compressors $c$, for which there are enough initially running compressor units contained in their set of usable compressor units $U_c$, such that the configuration might be active.

Having these sets, we can define the model for the initial time step. From the above-defined constraints, we use (10a), (12), (14), and (15) also for $t = 0$. In addition, we add the following new constraints:

$$x^{\mathrm{st}}_{s,0} = 0 \qquad \forall s \in \mathcal{S} \setminus \mathcal{S}^{0} \tag{21}$$

$$x^{\mathrm{cfg}}_{c,0} = 0 \qquad \forall a \in \mathcal{A}^{\mathrm{arCp}} \quad \forall c \in \mathcal{C}^{\mathrm{ar}}_{a} \setminus \mathcal{C}^{\mathrm{ar}0} \tag{22}$$

$$\sum_{a \in \mathcal{A}^{\mathrm{arCp}}} \sum_{c \in \mathcal{C}^{\mathrm{ar}}_{a} : u \in U_c} x^{\mathrm{cfg}}_{c,0} \geq 1 \qquad \forall u \in \mathcal{U}^{0}. \tag{23}$$

While the first two constraints ensure that only initially valid simple states and artificial configurations are chosen for the initial state, the last constraint demands for each initially running compressor unit $u$ that there is at least one initially active artificial configuration $c$ with $u \in U_c$.

## 2.10 Objective function

The general goal of the model is to reduce the amount of changes necessary for fulfilling the given demands while trying to reduce unnecessary compressor unit usage. In contrast to the base model of [Hoppmann-Baum et al., 2021], where a multi-level model was formulated to ensure each solution has minimal slack values, we instead used a single-level objective function representing the sum of differently weighted terms. Another difference from the base model is that we do not perform a solution *smoothing* in a post-processing step but instead include this already in the model, which increases its complexity.

To build our objective function, we need to introduce additional variables to track the corresponding changes of the single entities and connect these with the quantities describing the network state. We start with the changes of valve modes. For each non-fence-node valve $a \in \mathcal{A}^{\text{vaOr}}$ and time $t_1 \in \mathcal{T}$, we capture a change in the binary variable $\delta^{\text{va}}_{a,t_1}$ and specify the corresponding model as

$$m^{\text{op}}_{a,t_1} - m^{\text{op}}_{a,t_0} \leq \delta^{\text{va}}_{a,t_1} \tag{24a}$$
$$m^{\text{op}}_{a,t_0} - m^{\text{op}}_{a,t_1} \leq \delta^{\text{va}}_{a,t_1} \tag{24b}$$
$$\delta^{\text{va}}_{a,t_1} \in \{0,1\},$$

where $t_0 = t_1 - 1$ denotes the time step preceding $t_1$.

For the regulator changes, we use the model introduced in [Hennings et al., 2021a] tracking changes of the regulator's mode and, in case it is active, the regulator's operation point in terms of changes in the incoming pressure, outgoing pressure, and flow. For a regulator $a = (\ell, r) \in \mathcal{A}^{\text{rg}}$ and times $t_1 \in \mathcal{T}$ and $t_0 = t_1 - 1$, the model reads as

$$\delta^{\text{rg}}_{a,t_1} \geq m^{\text{x}}_{a,t_1} - m^{\text{x}}_{a,t_0} \qquad \forall \text{x} \in \{\text{cl}, \text{op}, \text{ac}\} \tag{25a}$$
$$\delta^{\text{rg}}_{a,t_1} \leq 2 - m^{\text{x}}_{a,t_1} - m^{\text{x}}_{a,t_0} \qquad \forall \text{x} \in \{\text{cl}, \text{op}, \text{ac}\} \tag{25b}$$
$$y_{t_1} - y_{t_0} \leq \delta^{\text{rg-y}}_{a,t_1} + (m^{\text{op}}_{a,t_1} + m^{\text{cl}}_{a,t_1} + \delta^{\text{rg}}_{a,t_1})(\bar{y}_{t_1} - \underline{y}_{t_0}) \qquad \forall y \in \{p_\ell, p_r, q_a\} \tag{25c}$$
$$y_{t_0} - y_{t_1} \leq \delta^{\text{rg-y}}_{a,t_1} + (m^{\text{op}}_{a,t_1} + m^{\text{cl}}_{a,t_1} + \delta^{\text{rg}}_{a,t_1})(\bar{y}_{t_0} - \underline{y}_{t_1}) \qquad \forall y \in \{p_\ell, p_r, q_a\} \tag{25d}$$
$$\delta^{\text{rg}}_{a,t_1} \in \{0,1\},$$

where $\delta^{\text{rg}}_{a,t_1}$ denotes the binary variable indicating a mode change at the regulator between times $t_0$ and $t_1$, and $\delta^{\text{rg-pl}}_{a,t_1}$, $\delta^{\text{rg-pr}}_{a,t_1}$, and $\delta^{\text{rg-qa}}_{a,t_1}$ track the changes in the operation point for an active regulator for the incoming pressure, outgoing pressure, and flow through the element.

Next are the changes regarding the network stations and their corresponding elements. First, we introduce binary variables $\delta^{\text{arc}}_{a,t}$, representing a change in the activity of the artificial arc $a$ between time $t$ and the previous time step. The corresponding constraints have the same structure as the valve constraints above and read for an artificial arc $a \in \mathcal{A}^{\text{ar}}$ and times $t_1 \in \mathcal{T}$ and $t_0 = t_1 - 1$ as

$$x^{\text{arc}}_{a,t_1} - x^{\text{arc}}_{a,t_0} \leq \delta^{\text{arc}}_{a,t_1} \tag{26a}$$
$$x^{\text{arc}}_{a,t_0} - x^{\text{arc}}_{a,t_1} \leq \delta^{\text{arc}}_{a,t_1} \tag{26b}$$
$$\delta^{\text{arc}}_{a,t_1} \in \{0,1\}.$$

For artificial compressors, we also count configuration changes as changes in the arc's activity. We only need to track if a configuration is newly activated, as turning it off either turns another configuration on or deactivates the whole compressor arc. Hence, we formulate for each configuration $c \in \mathcal{C}^{\text{ar}}_a$ of each artificial compressor $a \in \mathcal{A}^{\text{arCp}}$ and times $t_1 \in \mathcal{T}$ and $t_0 = t_1 - 1$ the corresponding constraint as

$$x^{\text{cfg}}_{c,t_1} - x^{\text{cfg}}_{c,t_0} \leq \delta^{\text{arc}}_{a,t_1}. \tag{27}$$

As in the base model, we track changes in the network stations' simple states but not in the flow directions. Since each station has exactly one simple state at each point in time, we capture the activation of a simple state $s$ at time $t$ in the binary variable $\delta^{\text{st}}_{s,t}$ and give the model for a simple state $s \in \mathcal{S}_i$ of a station $i \in \mathcal{I}$ and times $t_1 \in \mathcal{T}$ and $t_0 = t_1 - 1$ as

$$x^{\text{st}}_{s,t_1} - x^{\text{st}}_{s,t_0} \leq \delta^{\text{st}}_{s,t_1} \tag{28}$$
$$\delta^{\text{st}}_{s,t_1} \in \{0,1\}.$$

Finally, we capture the pressure changes at each fence node $v$ between time $t$ and the previous time point in the variable $\delta^{\text{fn-p}}_{v,t}$ and the flow changes at the corresponding fence node valve by the variable $\delta^{\text{fn-q}}_{v,t}$. The corresponding model for a fence node $v \in \mathcal{V}^{\text{fn}}_i$ of network station $i \in \mathcal{I}$ and times $t_1 \in \mathcal{T}$ and $t_0 = t_1 - 1$ is given as

$$p_{v,t_1} - p_{v,t_0} \leq \delta^{\text{fn-p}}_{v,t_1} \tag{29a}$$

$$p_{v,t_0} - p_{v,t_1} \leq \delta^{\text{fn-p}}_{v,t_1} \tag{29b}$$

$$q_{a^{\text{vaFn}}_v,t_1} - q_{a^{\text{vaFn}}_v,t_0} \leq \delta^{\text{fn-q}}_{v,t_1} \tag{29c}$$

$$q_{a^{\text{vaFn}}_v,t_0} - q_{a^{\text{vaFn}}_v,t_1} \leq \delta^{\text{fn-q}}_{v,t_1}. \tag{29d}$$

Having defined all the change tracking variables, we are now able to formulate the objective function as a weighted sum of different penalty terms. Apart from the changes and slack values defined above, we also penalize the usage of compressor units by penalizing the usage of artificial configurations $c$ weighted by their number of used compressor units $u_c$. For each of the terms in the sum, we have specific weights defining the penalty costs: The demand slack cost per hour $w^{\sigma\text{-d}}$, the pressure slack cost per hour $w^{\sigma\text{-p}}$, the cost for each compressor unit running in an active configuration per hour $w^{\text{ru}}$, the non-fence-node valve change cost $w^{\text{va}}$, the regulator mode change cost $w^{\text{rg}}$, the operation point change costs of active regulators for the incoming pressure $w^{\text{rg-pl}}$, the outgoing pressure $w^{\text{rg-pr}}$ and the flow $w^{\text{rg-q}}$, the artificial arc change costs $w^{\text{arc}}$, the costs for activating a simple state $w^{\text{st}}_s$, which are specific values manually defined by the industry experts for each simple state $s \in \mathcal{S}$, and the costs for changes of the fence node pressure $w^{\text{fn-p}}$ and flow changes on the corresponding fence node valves $w^{\text{fn-q}}$. The complete objective is then given as

$$\begin{aligned}
\min \sum_{t\in\mathcal{T}} \Bigg( & \sum_{v\in\mathcal{V}^{\text{b}}} \frac{\tau(t)-\tau(t-1)}{1\,\text{h}} \left(w^{\sigma\text{-d}}(\sigma^{d+}_{v,t} + \sigma^{d-}_{v,t}) + w^{\sigma\text{-p}}(\sigma^{p+}_{v,t} + \sigma^{p-}_{v,t})\right) \\
& + \sum_{a\in\mathcal{A}^{\text{arCp}}} \sum_{c\in\mathcal{C}^{\text{ar}}_a} \frac{\tau(t)-\tau(t-1)}{1\,\text{h}} \cdot u_c \cdot w^{\text{ru}} \cdot x^{\text{cfg}}_{c,t} \\
& + \sum_{a\in\mathcal{A}^{\text{vaOr}}} w^{\text{va}} \cdot \delta^{\text{va}}_{a,t} \\
& + \sum_{a\in\mathcal{A}^{\text{ar}}} w^{\text{arc}} \cdot \delta^{\text{arc}}_{a,t} \\
& + \sum_{i\in\mathcal{I}} \sum_{s\in\mathcal{S}_i} w^{\text{st}}_s \cdot \delta^{\text{st}}_{s,t} \\
& + \sum_{a\in\mathcal{A}^{\text{rg}}} \left(w^{\text{rg}} \cdot \delta^{\text{rg}}_{a,t} + w^{\text{rg-pl}} \cdot \delta^{\text{rg-pl}}_{a,t} + w^{\text{rg-pr}} \cdot \delta^{\text{rg-pr}}_{a,t} + w^{\text{rg-q}} \cdot \delta^{\text{rg-qa}}_{a,t}\right) \\
& + \sum_{i\in\mathcal{I}} \sum_{v\in\mathcal{V}^{\text{fn}}_i} \left(w^{\text{fn-p}} \cdot \delta^{\text{fn-p}}_{v,t} + w^{\text{fn-q}} \cdot \delta^{\text{fn-q}}_{v,t}\right) \Bigg).
\end{aligned} \tag{30}$$

### 2.11 Complete model

By combining all the above-defined variables and constraints with the objective function, we can formulate our gas flow problem $\mathscr{P}$ as the following MINLP model:

$$
\begin{array}{llll}
 & \min & (30) & \\
 & \text{s.t.}\quad \forall t \in \mathcal{T} & & \\
 & & (1)\quad \forall v \in \mathcal{V} & (2)\quad \forall a \in \mathcal{A}^{\mathrm{pi}} \\
 & & (3)\quad \forall a \in \mathcal{A}^{\mathrm{va}} & (4),(25)\quad \forall a \in \mathcal{A}^{\mathrm{rg}} \\
 & & (5)\quad \forall a \in \mathcal{A}^{\mathrm{arSc}} & (6)\quad \forall a \in \mathcal{A}^{\mathrm{arRg}} \\
 & & (7)\quad \forall a \in \mathcal{A}^{\mathrm{arBiRg}} & (10)\quad \forall a \in \mathcal{A}^{\mathrm{arCp}} \\
 & & (11),(27)\quad \forall a \in \mathcal{A}^{\mathrm{arCp}} \quad \forall c \in \mathcal{C}_a^{\mathrm{ar}} & (12)\quad \forall z \in \mathcal{Z} \\
 & & (14),(16)\quad \forall i \in \mathcal{I} & (15),(26)\quad \forall i \in \mathcal{I} \quad \forall a \in \mathcal{A}_i^{\mathrm{ar}} \\
 & & (17),(28)\quad \forall i \in \mathcal{I} \quad \forall s \in \mathcal{S}_i & (18),(29)\quad \forall i \in \mathcal{I} \quad \forall v \in \mathcal{V}_i^{\mathrm{fn}} \\
 & & (19)\quad \forall v \in \mathcal{V}^{\mathrm{b}} & (20)\quad \forall v \in \mathcal{V}^{+} : \hat{d}_{v,t} \neq 0 \\
 & & (24)\quad \forall a \in \mathcal{A}^{\mathrm{vaOr}} & \\
 & t = 0 & & \\
 & & (10\mathrm{a})\quad \forall a \in \mathcal{A}^{\mathrm{arCp}} & (12)\quad \forall z \in \mathcal{Z} \\
\mathscr{P} & & (14)\quad \forall i \in \mathcal{I} & (15)\quad \forall i \in \mathcal{I} \quad \forall a \in \mathcal{A}_i^{\mathrm{ar}} \\
 & & (21)\quad \forall s \in \mathcal{S} \setminus \mathcal{S}^0 & (22)\quad \forall a \in \mathcal{A}^{\mathrm{arCp}} \quad \forall c \in \mathcal{C}_a^{\mathrm{ar}} \setminus \mathcal{C}^{\mathrm{ar0}} \\
 & & (23)\quad \forall u \in \mathcal{U}^0 & \\
 & \forall t \in \mathcal{T}_0 & & \\
 & \{0,1\} \ni & m_{a,t}^{\mathrm{op}} \quad \forall a \in \mathcal{A}^{\mathrm{va}}, & m_{a,t}^{\mathrm{op}}, m_{a,t}^{\mathrm{ac}}, m_{a,t}^{\mathrm{cl}} \quad \forall a \in \mathcal{A}^{\mathrm{rg}}, \\
 & & x_{a,t}^{\mathrm{arc}} \quad \forall a \in \mathcal{A}^{\mathrm{ar}}, & x_{c,t}^{\mathrm{cfg}} \quad \forall a \in \mathcal{A}^{\mathrm{arCp}} \quad \forall c \in \mathcal{C}_a^{\mathrm{ar}}, \\
 & & x_{s,t}^{\mathrm{st}} \quad \forall i \in \mathcal{I} \quad \forall s \in \mathcal{S}_i & \\
 & \forall t \in \mathcal{T} & & \\
 & \{0,1\} \ni & x_{a,t}^{\mathrm{fwd}}, x_{a,t}^{\mathrm{bwd}} \quad \forall a \in \mathcal{A}^{\mathrm{arBiRg}}, & x_{f,t}^{\mathrm{fd}} \quad \forall i \in \mathcal{I} \quad \forall f \in \mathcal{F}_i, \\
 & & \delta_{a,t}^{\mathrm{va}} \quad \forall a \in \mathcal{A}^{\mathrm{vaOr}}, & \delta_{a,t}^{\mathrm{rg}} \quad \forall a \in \mathcal{A}^{\mathrm{rg}}, \\
 & & \delta_{a,t}^{\mathrm{arc}} \quad \forall a \in \mathcal{A}^{\mathrm{ar}}, & \delta_{s,t}^{\mathrm{st}} \quad \forall s \in \mathcal{S}.
\end{array}
$$

## 3 MIP-based heuristic algorithms

We aim at solving the transient gas flow problem presented in the previous section for challenging demand scenarios on large real-world-based instances. Since this is currently out of reach for black-box MINLP solvers, we create a specialized solution approach given as Algorithm 1. It is based on the idea of dealing with the two sources of complexity, the non-continuity of binary variables and the non-convexity of the constraints, separately: In the first step, we determine a promising and feasible set of solution values for all the binary variables, which we call a *binary assignment* of the problem. Afterwards, we transform the problem into an NLP by fixing these values and then solve this remaining problem with a black-box NLP solver. Finally, we return the overall best solution out of all successful NLP runs. If non of the runs found a feasible solution, the heuristic failed. Note that the algorithm could be easily parallelized by solving the NLP model immediately after finding a new binary assignment and in parallel to the ongoing search for binary assignments. Hence, solutions would be obtainable early during the execution, which is especially valuable in environments with strict time limit restrictions.

Since we consider especially challenging and large instances, we purely focus on finding good feasible solutions fast, i.e., we do not aim for valid global lower bounds for the problem and thereby prove optimality of our solutions. However, note that in our approach, the black-box NLP solver usually provides, at least locally, optimal solutions for the corresponding sub-problem for a given binary assignment.

In the remainder of this section, we focus on finding binary assignments that yield high-quality solutions for the overall problem. Note that we sometimes claim that these binary-assignment-producing

**Algorithm 1:** Base algorithm

```
Data: An instance $\Phi$ of problem $\mathscr{P}$
Result: A valid solution for $\Phi$
1 baSet ← findBinaryAssignments(Φ)
2 solSet ← solveNLP(baSet, Φ)
3 return chooseBestSol(solSet)
```

algorithms solve the overall problem, which we use as a short form of explaining that we solve the problem by using them as function `findBinaryAssignments` in Algorithm 1.

## 3.1 Sequential mixed-integer programming

The core of our algorithm for finding binary assignments consists of a sequential algorithm solving linearized versions of the original MINLP problem, which we refer to as `SMIP` and which is given as Algorithm 2. After initializing a linearization for the non-convex Momentum equation (2b) in Line 2, we solve the remaining MIP problem. If successful, this yields a valid binary assignment in the sense that it respects all the constraints involving no continuous variables. We then check if the binary assignment was already found before and if the linearization error is small enough to fulfill the convergence criteria. If both are not the case and we have not reached the maximum number of iterations $n^{\text{it}}$, we update our linearization based on the current solution and repeat the process by solving the updated MIP model. The algorithm finally returns the set of found binary assignments to be completed to full solutions for the original problem $\mathscr{P}$. Note that given a valid binary assignment, the resulting NLP model might be infeasible and hence does not yield a feasible overall solution.

### 3.1.1 Linearization and convergence

As already mentioned above, the only non-linear term in the model is the friction term in the Momentum equation (2b) for pipes given as

$$c_a \cdot \left( \frac{|q_{\ell,a,t}| q_{\ell,a,t}}{p_{\ell,t}} + \frac{|q_{r,a,t}| q_{r,a,t}}{p_{r,t}} \right) \qquad \text{with the constant } c_a := \frac{\lambda_a R_{\text{s}} T_a z_a L_a}{4 A_a^2 D_a}.$$

Hence, we are given for each Momentum equation of pipe $a = (\ell, r) \in \mathcal{A}^{\text{pi}}$ and time $t \in \mathcal{T}$ two non-linear functions of the form $c_a(|q_{x,a,t}| q_{x,a,t}/p_{x,t})$, one for each end node $x \in \{\ell, r\}$. Both functions are approximated by a linear combination of the involved quantities as

$$c_a \frac{|q_{x,a,t}| q_{x,a,t}}{p_{x,t}} \approx \alpha^0_{x,a,t} + \alpha^p_{x,a,t} \cdot p_{x,t} + \alpha^q_{x,a,t} \cdot q_{x,a,t}.$$

To check if a solution of a linearized MIP model has converged against a non-linear solution, we use two tolerance parameters: The absolute convergence tolerance $n^{\text{tolAbs}}$ and the relative convergence tolerance $n^{\text{tolRel}}$. Given these, we define that a solution with pressure values $p'$ and mass flow values $q'$ converged, if for all pipes and future time points the differences between the linearized friction function and the non-linear friction function with respect to $p'$ and $q'$ are smaller than the tolerances. Hence, we check in function `isLinearizationErrorSmall` used in Line 10 of Algorithm 2 if the following constraints are fulfilled:

$$\forall t \in \mathcal{T} \quad \forall a = (\ell, r) \in \mathcal{A}^{\text{pi}} :$$

$$\begin{aligned}
\Lambda^{\text{lin}} &= \alpha^0_{\ell,a,t} + \alpha^p_{\ell,a,t} \cdot p'_{\ell,t} + \alpha^q_{\ell,a,t} \cdot q'_{\ell,a,t} \\
&\quad + \alpha^0_{r,a,t} + \alpha^p_{r,a,t} \cdot p'_{r,t} + \alpha^q_{r,a,t} \cdot q'_{r,a,t} \\
\Lambda^{\text{orig}} &= c_a \cdot \left( \frac{|q'_{\ell,a,t}| q'_{\ell,a,t}}{p'_{\ell,t}} + \frac{|q'_{r,a,t}| q'_{r,a,t}}{p'_{r,t}} \right) \\
n^{\text{tolAbs}} &\geq |\Lambda^{\text{lin}} - \Lambda^{\text{orig}}|
\end{aligned}$$

and, if $|\Lambda^{\text{orig}}| > \varepsilon^{\text{zero}}$,

$$n^{\text{tolRel}} \geq \frac{|\Lambda^{\text{lin}} - \Lambda^{\text{orig}}|}{|\Lambda^{\text{orig}}|}.$$

**Algorithm 2:** General sequential mixed-integer programming algorithm (SMIP)

```
Data: An instance $\Phi$ of problem $\mathscr{P}$
Parameters: Maximum number of iteration $n^{\text{it}}$
            Absolute convergence tolerance $n^{\text{tolAbs}}$
            Relative convergence tolerance $n^{\text{tolRel}}$
Result: A list of valid binary assignments for $\Phi$
1  baSet ← createSet()
2  mip ← applyInitialLinearization($\Phi$)
3  for $i \leftarrow 1$ to $n^{\text{it}}$ do
       // solve the linearized MIP problem
4      foundPrimalSolution, solutionPointer ←solve(mip)
5      if foundPrimalSolution then
           // Found a feasible solution for the linearized MIP
6          ba ← getBinarySolutionValues(solutionPointer)
7          if ba ∈ baSet then
               // Found an already known binary assignment
               // ⇒ abort the iteration
8              break // the loop
           // Found a new binary assignment
9          baSet.add(ba)
10         if isLinearizationErrorSmall(mip, solutionPointer, $n^{\text{tolAbs}}$, $n^{\text{tolRel}}$) then
               // Given solution fulfills the convergence criterion
               // ⇒ abort the iteration
11             break // the loop
           // update linearization
12         mip ← linearizeBasedOnPreviousSolution($\Phi$,solutionPointer)
13     else
           // linearized MIP no successful
           // ⇒ abort the iteration
14         break // the loop
15 return baSet
```

Here, the value of the linearized friction function with respect to the variable values of the given solution is denoted by $\Lambda^{\text{lin}}$ and the corresponding non-linear friction function value by $\Lambda^{\text{orig}}$. Furthermore, we can only check for the relative tolerance if the non-linear friction value is not zero. We ensure this by comparing the term against a corresponding epsilon value $\varepsilon^{\text{zero}}$, which we define to be equal to 0.001 bar.

For the initial linearization determined in function `applyInitialLinearization` of Algorithm 2 in Line 2, we use the constant velocity approximation introduced by [Hennings, 2018], which was also used as initial linearization in [Hoppmann-Baum et al., 2021] and is given as

$$\alpha^0_{x,a,t} = 0 \quad \alpha^p_{x,a,t} = 0 \quad \alpha^q_{x,a,t} = \frac{\lambda_a L_a}{4 A_a D_a} |v_{x,a,0}|$$
$$\text{with} \quad |v_{x,a,0}| := \frac{R_{\text{s}} T_a z_a}{A_a} \frac{q_0}{p_{x,0}}, \quad \text{and} \quad q_0 := \max\{|q_{x,a,0}|, q^{\min}\}.$$

Here, $|v_{x,a,0}|$ denotes the absolute value of the gas velocity based on the initial state quantities using a minimal absolute flow value of $q^{\min}$ to avoid $\alpha^q_{x,a,t}$ being zero. The advantage of this linearization is that it is symmetric in $q$ and also zero for $q = 0$.

For all subsequent iterations of Algorithm 2, we determine the linearization based on the pressure values $p'$ and flow values $q'$ of a previous MIP solution by function `linearizeBasedOnPreviousSolution` in Line 12. As linearization, we use the tangential plane on the non-linear function in the point ($p'$, $q'$), which is the first-order Taylor expansion at this point and is given as

$$\begin{aligned}
\alpha^{p}_{x,a,t} &= c_a \cdot \frac{-|q'_{x,a,t}|q'_{x,a,t}}{(p'_{x,t})^2} \\
\alpha^{q}_{x,a,t} &= c_a \cdot \frac{2|q'_{x,a,t}|}{p'_{x,t}} \\
\alpha^{0}_{x,a,t} &= c_a \cdot \frac{|q'_{x,a,t}|q'_{x,a,t}}{p'_{x,t}} - \alpha^{p}_{x,a,t} \cdot p'_{x,t} - \alpha^{q}_{x,a,t} \cdot q'_{x,a,t}.
\end{aligned}$$

### 3.1.2 Comments on infeasibility

Algorithm 2 struggles with infeasibility in two different ways: First, the linearized MIP problems of each iteration might be infeasible even if the overall MINLP problem $\mathscr{P}$ is not. Second, the algorithm has no functionality to detect the overall infeasibility of the general problem $\mathscr{P}$.

Despite those structural weaknesses, we still expect the algorithm to perform well on real-world-based instances, like those used in our computational experiments in Section 4. On these, infeasibility usually does not occur, as real-world gas transport networks are, in general, very flexible and allow for a broad range of different pressure, flow, and inflow values. In addition, the slack values can adjust problematic inflow and pressure demands.

The computational results we present in Section 4 confirm our expectations, as we found a feasible solution for each instance of the problem $\mathscr{P}$ and rarely encountered infeasibility when solving the linearized MIP models, see Table 5.

## 3.2 Reduced time horizon heuristics

To further accelerate the solving process, we introduce additional heuristics that find binary assignments by solving a series of models on time horizons of reduced size, where the size of a time horizon is defined as the number of future time steps. The heuristics are based on two well-known ideas for time-expanded problems: The rolling horizon and the aggregated horizon. While rolling horizon heuristics rely on iteratively solving the model for a subset of time steps that are gradually shifted to the future, the aggregated horizon combines certain time steps and afterwards completes the aggregated solution to one for the whole time horizon.

Note that we maintain the general sequential algorithm structure of Algorithm 2 for all of the heuristics below.

### 3.2.1 Rolling horizon heuristic

In our rolling horizon heuristic `RH` given in Algorithm 3, we only change a single line in contrast to the `SMIP` Algorithm 2: Instead of solving the linearized MIP for the whole time horizon in Line 4, we find a solution by using a rolling horizon approach of fixed length $n^{\text{rolHor}}$. We start by building a reduced model in Line 8 that only considers the first $n^{\text{rolHor}}$ future time steps by not adding any variables or constraints for the remaining ones. After solving the reduced model, we save all variable values regarding the first of the considered future time steps for our overall solution in Line 12. We call this set of values the *solution state* of this time step and use it as the initially fixed state when building the reduced model in the next iteration. Note that we still use the overall initial time step 0 to determine the model's constant parameters, like, for example, the fixed temperature $T_a$ of a pipe $a \in \mathcal{A}^{\text{pi}}$. Finally, for the next iteration, we also include the earliest not already included future time step into our reduced time horizon, which thereby again has a length of $n^{\text{rolHor}}$. We repeat this procedure until we reach the iteration containing the very last future time step $t^{\max}$. After saving all the solution states of this iteration in Line 14, we can create a solution for the entire time horizon by combining all the collected solution states in Line 18.

As a minor detail, we note that for all variables without a fixed initial state for time $t = 0$, for example, for the simple state variables $x^{\text{st}}$, we use the model as defined in Section 2 for the first iteration and then add the there determined values for $t = 0$ to the corresponding solution state. For all subsequent iterations, we fix the values determined in the previous iteration for the initial state and can therefore ignore all the constraints added to the model for $t = 0$.

**Algorithm 3:** Sequential rolling horizon heuristic algorithm (`RH`)

```
Data: An instance $\Phi$ of problem $\mathscr{P}$
Parameters: Size of the rolling horizon $n^{\text{rolHor}}$
            Maximum number of iteration $n^{\text{it}}$
            Absolute convergence tolerance $n^{\text{tolAbs}}$
            Relative convergence tolerance $n^{\text{tolRel}}$
Result: A list of valid binary assignments for $\Phi$
1  Function solveByRollingHorizonHeuristic(mip) is
2      initialTimeStep ← 0
3      savedSolutionStates ← createList()
4      savedSolutionStates.append(getStartState($\Phi$))
5      while initialTimeStep + $n^{\text{rolHor}} \leq t^{\max}$ do
6          startState ← savedSolutionStates.getLast()
7          lastTimeStep ← initialTimeStep + $n^{\text{rolHor}}$
           // Create MIP model with reduced time horizon
8          reducedMip ← createModelForReducedTimeHorizon($\Phi$, startState, initialTimeStep,
            lastTimeStep)
9          foundPrimalSolution, solutionPointer ← solve(reducedMip)
10         if foundPrimalSolution then
               // Found feasible solution for the reduce time horizon mip
               // check if this is the last iteration
11             if initialTimeStep + $n^{\text{rolHor}} < t^{\max}$ then
                   // This is not the last iteration
                   // ⇒ save state of first future time step
12                 savedSolutionStates.append(getFirstFutureState(solutionPointer))
13             else
                   // This is the last iteration!
                   // ⇒ Save all future solution states
14                 savedSolutionStates.appendAll(getAllFutureState(solutionPointer))
               // Prepare the next iteration
15             initialTimeStep ← initialTimeStep + 1
16         else
               // Reduced time horizon mip was not successful
               // return foundPrimalSolution=False
17             return *False, nullptr*
       // All iterations have been successful!
       // ⇒ Return final solution
18     return *True,* combineStatesToOverallSolution(savedSolutionStates)
   // Use the SMIP algorithm, but replace the 'solve' function
19 return SMIP($\Phi$, $n^{\text{it}}$, $n^{\text{tolAbs}}$, $n^{\text{tolRel}}$, solve = solveByRollingHorizonHeuristic)
```

### 3.2.2 Aggregated horizon heuristic

For the aggregated horizon heuristic `AH` stated as Algorithm 4, we create a time horizon of reduced size $n^{\text{aggHor}}$ by combining certain time steps, resulting in longer but fewer time steps in the overall horizon. We represent this by a function $\vartheta : \{0, 1, \dots, n^{\text{aggHor}}\} \to \mathcal{T}_0 = \{0, 1, \dots, t^{\max}\}$, which maps the time steps of the aggregated time horizon to time steps of the original time horizon and is created as variable `aggToOrigMap` by calling the function `mapAggregatedToOriginalTimeSteps` in Line 1. A time interval between two subsequent time steps $t$ and $t+1$ in the aggregated horizon represents the combination of all original time intervals between the time points $\vartheta(t)$ and $\vartheta(t+1)$. To combine a similar amount of original time steps for each time interval, we determine $\vartheta$ as

$$\begin{aligned}
\bar{c} &:= \lceil \frac{t^{\max}}{n^{\text{aggHor}}} \rceil && \text{the larger number of original time intervals to combine}\\
\underline{c} &:= \lfloor \frac{t^{\max}}{n^{\text{aggHor}}} \rfloor && \text{the smaller number of original time intervals to combine}\\
\bar{n} &:= t^{\max} \mod n^{\text{aggHor}} && \text{the amount of larger aggregated time intervals}\\
\vartheta(t) &:= \begin{cases} t \cdot \bar{c} & \text{if } t \le \bar{n} \\ \bar{c} \cdot \bar{n} + t \cdot \underline{c} & \text{if } \bar{n} < t \end{cases},
\end{aligned}$$

and give an example as

$$t^{\max} = 7 \qquad \& \qquad n^{\text{aggHor}} = 4 \qquad \Longrightarrow \qquad \begin{array}{c|ccccc} t & 0 & 1 & 2 & 3 & 4 \\ \vartheta(t) & 0 & 2 & 4 & 6 & 7 \end{array}.$$

Note that $\vartheta(0) = 0$ and $\vartheta(n^{\text{aggHor}}) = t^{\max}$ hold. If $n^{\text{aggHor}}$ is a divisor of $t^{\max}$, each aggregated time interval combines $\bar{c} = \underline{c}$ original time intervals.

Using the function $\vartheta$, we can now derive a time aggregated problem instance $\Phi^{\text{agg}}$ from the original problem instance $\Phi$ by calling the function `createTimeAggregatedProblemInstance` in Line 2 of Algorithm 4. Since all the elements sets of $\Phi$ and $\Phi^{\text{agg}}$ coincide except of the time horizon, we are only missing a description of the time-dependent instance parameters. In a first step, we determine the inflow demands. As they represent the average inflow rate over a time interval, we determine the time aggregated inflow demands as the average of original inflow demands, weighted by the length of the corresponding time intervals, i.e., for the aggregated time $t \in \{1, \dots, n^{\text{aggHor}}\}$ and a boundary node $v \in \mathcal{V}^{\text{b}}$ we define

$$\hat{d}_{v,t} := \frac{\sum_{i=\vartheta(t-1)+1}^{\vartheta(t)} (\tau(i) - \tau(i-1)) \cdot \hat{d}_{v,i}}{\tau(\vartheta(t)) - \tau(\vartheta(t-1))}.$$

For all the remaining time-dependent parameters, like, for example, the function $\tau(t)$ representing the time difference between $t$ and the initial time step in seconds, the pressure demands at the entries, or the general pressure and flow bounds, we use for each aggregated time point $t$ the parameter value of the corresponding original time point $\vartheta(t)$ in $\Phi$ and thereby complete the aggregated instance $\Phi^{\text{agg}}$.

Using the `SMIP` Algorithm 2, we determine a set of valid binary assignments for the aggregated problem instance. Note that we use the parameter $n^{\text{itAgg}}$ to specify the maximum number of iterations to use in `SMIP` to solve $\Phi^{\text{agg}}$. In the following, we explain the steps to complete these to full binary assignments for the original problem instance $\Phi$. The general idea is to fix certain variables in the model based on the aggregated binary assignment and thereby create the problem instance $\Phi^{\text{fixed}}$, which we then again solve by a variant of the `SMIP` Algorithm 2 to find the binary values for the rest of the original time horizon. Our set of variables to fix for creating $\Phi^{\text{fixed}}$ is given in Line 4 of Algorithm 4.

Since we do not fix all the variables, the aggregated binary assignments created by solving $\Phi^{\text{agg}}$ might be identical with respect to the variable fixations. In order to avoid unnecessary computation, we remove duplicated binary assignments by calling the function `removeDuplicateBAsWithRespectToVarsToFix` in Line 5 of Algorithm 4. Now, we create for each remaining binary assignment a variant $\Phi^{\text{fixed}}$ of the original problem instance $\Phi$ by calling function `createProblemInstanceWithPartiallyFixedBinaries` in Line 8. When denoting the binary assignment by $B$ and the value of a binary variable $x$ in $B$ by $\varsigma(x, B)$, we perform the following fixations:

$$\begin{aligned}
&\forall t^{\text{agg}} \in \{0, 1, \dots, n^{\text{aggHor}}\} \text{ and variables } x \in \text{varTypesToFix}: && x_{\vartheta(t^{\text{agg}})} = \varsigma(x_{t^{\text{agg}}}, B)\\
&\forall t^{\text{agg}} \in \{1, \dots, n^{\text{aggHor}}\} \text{ and variables } x \in \text{varTypesToFix} &&\\
&\quad \text{with } \varsigma(x_{t^{\text{agg}}-1}, B) = \varsigma(x_{t^{\text{agg}}}, B), &&\\
&\quad \forall t \in \{\vartheta(t^{\text{agg}} - 1) + 1, \dots, \vartheta(t^{\text{agg}}) - 1\}: && x_t = \varsigma(x_{t^{\text{agg}}}, B)
\end{aligned}$$

**Algorithm 4:** Sequential aggregated horizon heuristic algorithm (AH)

```
   Data: An instance $\Phi$ of problem $\mathscr{P}$
   Parameters: Size of the aggregated horizon $n^{\mathrm{aggHor}}$
               Maximum number of iteration for the aggregated horizon MIP $n^{\mathrm{itAgg}}$
               Maximum number of iteration for the full horizon MIP $n^{\mathrm{itFull}}$
               Absolute convergence tolerance $n^{\mathrm{tolAbs}}$
               Relative convergence tolerance $n^{\mathrm{tolRel}}$
   Result: A list of valid binary assignments for $\Phi$
 1 aggToOrigMap ← mapAggregatedToOriginalTimeSteps($\Phi$,$n^{\mathrm{aggHor}}$)
   // Create problem on aggregated time horizon
 2 $\Phi^{\mathrm{agg}}$ ← createTimeAggregatedProblemInstance($\Phi$, aggToOrigMap)
   // Solve aggregated problem via SMIP
 3 aggregatedBAs ← SMIP($\Phi^{\mathrm{agg}}$, $n^{\mathrm{itAgg}}$, $n^{\mathrm{tolAbs}}$, $n^{\mathrm{tolRel}}$)
 4 varTypesToFix ← {
     $m_a^{\mathrm{op}}, m_a^{\mathrm{ac}}, m_a^{\mathrm{cl}}\ \forall a \in \mathcal{A}^{\mathrm{arRg}}$,   $m_a^{\mathrm{op}}\ \forall a \in \mathcal{A}^{\mathrm{vaOr}}$,
     $x_a^{\mathrm{arc}}\ \forall a \in \mathcal{A}^{\mathrm{ar}}$,   $x_c^{\mathrm{cfg}}\ \forall c \in \mathcal{C}_a^{\mathrm{ar}}\ \forall a \in \mathcal{A}^{\mathrm{arCp}}$,   $x_s^{\mathrm{st}}\ \forall s \in \mathcal{S}$}
 5 aggregatedBAs ← removeDuplicateBAsWithRespectToVarsToFix(aggregatedBAs,
    varTypesToFix)
 6 baSetFullHorizon ← createList()
   // Complete the time aggregated binary assignments to those for the full horizon
 7 for aggBA ∈ aggregatedBAs do
 8     $\Phi^{\mathrm{fixed}}$ ← createProblemInstanceWithPartiallyFixedBinaries($\Phi$, aggBA, varTypesToFix,
        aggToOrigMap)
       // Solve SMIP with adjusted initial linearization and partial start solution
 9     baSetForAggBA ← SMIP($\Phi^{\mathrm{fixed}}$, $n^{\mathrm{itFull}}$, $n^{\mathrm{tolAbs}}$, $n^{\mathrm{tolRel}}$,
        applyInitialLinearization =
         initializeLinearizationBasedOnAggSolution($\Phi^{\mathrm{fixed}}$, aggBA),
        solve = solveMIPUsingPartialStartSolution(mip, aggBA))
10     baSetFullHorizon.addAll(baSetForAggBA)
   // Return all full horizon binary assignment found be completing aggregated ones
11 return baSetFullHorizon
```

In words, we do two types of fixations: First, we fix all binary variables of the corresponding types whose time steps are part of the aggregated time horizon to their corresponding solution values defined by $B$. Second and for all time steps in between those time steps from the aggregated horizon, we check if a variable has the same solution value with respect to $B$ at the two surrounding time steps from the aggregated horizon. If it does, we also fix them to exactly that solution value. Otherwise, we do not fix its value. This means that we allow a variable value change between two aggregated time points to happen at any point in time in the corresponding original time interval.

After creating the instance $\Phi^{\mathrm{fixed}}$, we again solve it by the SMIP Algorithm 2 using the parameter $n^{\mathrm{itFull}}$ to specify its maximum number of iterations. However, we make two adjustments to the algorithm. First, we replace the `applyInitialLinearization` function with a newly defined function `initializeLinearizationBasedOnAggSolution`. In there, we still use the constant velocity approximation but determine the coefficient $\alpha_{x,a,t}^{q}$ for each time $t$, pipe $a = (\ell, r)$ and end node $x \in \{\ell, r\}$ based on the pressures and flows of the aggregated solution that created the binary assignment. If $t = \vartheta(t^{\mathrm{agg}})$ for some $t^{\mathrm{agg}}$ in the aggregated time horizon, we use the pressures and flows of that aggregated time step. Otherwise, we linearly interpolate the values. As a second adjustment, we provide a partial starting solution to each attempt to solve a linearized MIP, from which the solver tries to build a complete solution for the problem. We represent this adjustment by replacing the original `solve` function with the function `solveMIPUsingPartialStartSolution`. As the partial start solution, we provide the values for all binary variables $x_f^{\mathrm{fd}}$, which determine the activity of a flow direction $f$, and $m_a^{\mathrm{op}}$, which represent the mode for all fence node valves $a$. The values for each entity and time step are created based on the given binary assignment using the same algorithm as in function `createProblemInstanceWithPartiallyFixedBinaries`. Since both variables do not have a meaningful value for the initial time step, we use for all original time steps, which exist before the first time step with a corresponding aggregated time step, the variable value of that aggregated time step.

As a result of calling the adjusted `SMIP` in Line 9, we obtain a set of binary assignments for the original problem instance $\Phi$. The union of these for all aggregated binary assignments is returned as the result of Algorithm 4. Note that the binary assignments obtained by completing different aggregated binary assignments are also different due to the guaranteed variable fixations on those original time steps with a corresponding aggregated time step.

### 3.2.3 Aggregated horizon heuristic using rolling horizon

As the final heuristic based on reduced time horizons, we propose a combination of the previous two algorithms and is given as Algorithm 5. We refer to it as `ARH`.

**Algorithm 5:** Sequential aggregated and rolling horizon heuristic algorithm (`ARH`)

```
  Data: An instance $\Phi$ of problem $\mathscr{P}$
  Parameters: Size of the aggregated horizon $n^{\text{aggHor}}$
              Size of the rolling horizon $n^{\text{rolHor}}$
              Maximum number of iteration for the aggregated horizon MIP $n^{\text{itAgg}}$
              Maximum number of iteration for the full horizon MIP $n^{\text{itFull}}$
              Absolute convergence tolerance $n^{\text{tolAbs}}$
              Relative convergence tolerance $n^{\text{tolRel}}$
  Result: A list of valid binary assignments for $\Phi$
1 ahUsingRh ← adjustAhByUsingRollingHorizonToSolveAggregatedMIP($n^{\text{rolHor}}$)
2 return ahUsingRh($\Phi$, $n^{\text{aggHor}}$, $n^{\text{itAgg}}$, $n^{\text{itFull}}$, $n^{\text{tolAbs}}$, $n^{\text{tolRel}}$)
```

The heuristic is nearly equivalent to the aggregated horizon heuristic given as Algorithm 4. The only difference is that we use in Line 3 the rolling horizon Algorithm 3 instead of the sequential mixed-integer programming Algorithm 2 to solve the problem instance $\Phi^{\text{agg}}$ on the aggregated time horizon. We represent this in Algorithm 5 by the function `adjustAhByUsingRollingHorizonToSolveAggregatedMIP`, which returns a copy of Algorithm 4 adjusted by the mentioned change. This adjusted algorithm is then used to solve the given problem instance $\Phi$. Note that we are given the parameter $n^{\text{rolHor}}$, which specifies the length of the rolling horizon.

## 3.3 A dynamic node limit

When testing the above-given heuristics, we observed that during the solving process of the linearized MIP models, we find good primal solutions fast and then spend much time proving their optimality. This is typical behavior for branch-and-bound-based MIP solvers according to [Berthold et al., 2018]. However, for our heuristics, we are only interested in the optimal (or at least a good) primal solution for each MIP. The corresponding dual bound is not relevant to us. Therefore, there is the potential to save time if we can end the solving process at the point at which the solver has found a good solution and changed its focus to improving the dual bound.

To implement the described behavior, we define a maximum limit $L$ on the number of processed branch-and-bound nodes, which depends on the so far found feasible solutions and can be applied to the MIP solving processes in the heuristics. The general idea is to increase the node limit as long as we find new best primal solutions that significantly improve the primal bound. If this is no longer the case, we hit the dynamic node limit and stop the run with the current best solution.

At the start of the solving process, we set the limit to $L = \infty$. Whenever we find a new best feasible solution with objective function value $O$ at node $N$, we update the node limit dynamically as follows: Let $O^{\text{last}}$ be the objective function value of the last feasible solution that updated the node limit and let $O^{\text{last}} = \infty$ if no feasible solution has been found so far. Then update can be stated as

$$\text{If } \frac{O^{\text{last}} - O}{O^{\text{last}}} > n^{\text{improve}}: \qquad L = \max(n^{\text{limitMin}}, n^{\text{limitRel}} \cdot N)$$
$$O^{\text{last}} = O.$$

In words, we update the node limit $L$ whenever there is a significant relative improvement of at least $n^{\text{improve}} \in (0, 1)$ of the primal bound compared to its value at the last node limit update. The new limit has a minimal value of $n^{\text{limitMin}}$ and is otherwise set to $n^{\text{limitRel}}$ times the current number of processed branch-and-bound nodes. The minimal node limit avoids a premature termination of the solver and should cover the early stage of the solving process, in which the relative increments in the number of nodes are still relatively high.

## 4 Computational Experiments

The performance of the presented algorithms is tested by conducting a series of computational experiments. The corresponding set of instances is based on a real-world network with corresponding initial states as well as inflow and pressure patterns at the boundary nodes, which are provided by our project partner OGE [OGE]. From this data we choose 40 especially challenging instances, by finding those with the highest amount of demand changes. After giving a description of the corresponding network features and describing the instance generation process, we specify the setup of the experiments as well as the used parameters. Then, we present the results of the two conducted experiments: First, we compare the results of the general sequential mixed-integer programming (`SMIP`) Algorithm 2 used in Algorithm 1 against those of a black-box global MINLP solver. Here, we focus purely on the solution quality to investigate if the algorithm is able to yield solutions close to the proven optimum. In a second experiment, we evaluate the general performance of the reduced time horizon heuristics presented in Section 3.2 while using the general `SMIP` approach as a reference.

### 4.1 Instance generation

The instances for testing our algorithms are based on a section of the real-world network of our project partner OGE [OGE], which was also used in [Hoppmann-Baum et al., 2021]. It has been modified by replacing the original network station topology with the artificial model that was manually created by industry experts. Furthermore, the network outside of network stations is aggregated using the methods presented in [Lenz, 2021].

We choose the demand scenarios defining our instances from a historical time period of 12 consecutive months. During this time, the network topology, the manually created station model, and the parameters of the applied network aggregation changed multiple times. Hence, we are given a slightly different network topology for each instance. An overview of the network characteristics is given in Table 1, in which the minimal and maximal amounts for each element type are given. Most noteworthy, the number of network stations changed from 7 to 10, as three regulators arrangements in the network were replaced by small stations. We note that compared to the instances used in [Hoppmann-Baum et al., 2021], a less aggressive network aggregation was used, resulting in more elements outside of network stations.

| | $\lvert\mathcal{V}\rvert$ | $\lvert\mathcal{V}^+\rvert$ | $\lvert\mathcal{V}^-\rvert$ | $\lvert\mathcal{A}\rvert$ | $\lvert\mathcal{A}^{\text{pi}}\rvert$ | $\lvert\mathcal{A}^{\text{vaOr}}\rvert$ | $\lvert\mathcal{A}^{\text{rg}}\rvert$ | $\lvert\mathcal{A}^{\text{arSc}}\rvert$ | $\lvert\mathcal{A}^{\text{arRg}}\rvert$ | $\lvert\mathcal{A}^{\text{arBiRg}}\rvert$ | $\lvert\mathcal{A}^{\text{arCp}}\rvert$ | $\lvert\mathcal{I}\rvert$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| min | 375 | 12 | 114 | 421 | 303 | 0 | 11 | 29 | 20 | 2 | 16 | 7 |
| max | 477 | 13 | 162 | 531 | 391 | 1 | 13 | 42 | 28 | 3 | 19 | 10 |

Table 1: Network topology statistics. Since the network and manually created station model change over time, we give for each quantity the corresponding minimal and maximal amounts.

In Table 2, we list parameters regarding the network stations. If at least one of these values changed over time, we give two sets of values for a station representing the corresponding minimal and maximal amounts per quantity. The network stations A to G are the same as in [Hoppmann-Baum et al., 2021], while the newly introduced stations replacing the regulators are named H, I, and J.

**Artificial compressor model size** Based on the specific network station characteristics in the given time period, we compare the actual sizes of the artificial compressors arc models used in the base model of [Hoppmann-Baum et al., 2021] and our newly introduced model from Section 2.5.4. We count the number of continuous and binary variables as well as the number of linear inequality constraints needed for each discrete time step. For the variables, we included the activation and flow variable per arc as well as all additional variables needed specifically for artificial compressors. The constraints we count for our model are those defined in Equations (10), (11), and (12), while we count all the constraints (31)-(42) from [Hoppmann-Baum et al., 2021] for the base model. Since we count inequality constraints, equations are counted twice. The result is given in Table 3. We see that our model is smaller regarding all three categories, as it uses a slightly smaller number of binary variables and considerably fewer continuous variables and inequality constraints.

| $i \in \mathcal{I}$ | $\lvert\mathcal{V}_i^{\text{fn}}\rvert$ | $\lvert\mathcal{A}_i^{\text{arSc}}\rvert$ | $\lvert\mathcal{A}_i^{\text{arRg}}\rvert$ | $\lvert\mathcal{A}_i^{\text{arBiRg}}\rvert$ | $\lvert\mathcal{A}_i^{\text{arCp}}\rvert$ | $\lvert\mathcal{S}_i\rvert$ | $\lvert\mathcal{F}_i\rvert$ | $\sum_{a\in\mathcal{A}_i^{\text{arCp}}} \lvert\mathcal{C}_a^{\text{ar}}\rvert$ | $\lvert\mathcal{Z}_i\rvert$ |
|---|---|---|---|---|---|---|---|---|---|
| A | 2 | 1 | 0 | 0 | 2 | 5 | 3 | 7 | 0 |
| $\text{B}_{\text{min}}$ | 2 | 1 | 0 | 0 | 3 | 5 | 2 | 4 | 0 |
| $\text{B}_{\text{max}}$ | 2 | 1 | 0 | 0 | 3 | 6 | 3 | 4 | 0 |
| $\text{C}_{\text{min}}$ | 4 | 2 | 6 | 0 | 1 | 3 | 4 | 1 | 0 |
| $\text{C}_{\text{max}}$ | 6 | 4 | 6 | 0 | 1 | 9 | 6 | 1 | 0 |
| $\text{D}_{\text{min}}$ | 3 | 2 | 1 | 0 | 2 | 5 | 7 | 2 | 0 |
| $\text{D}_{\text{max}}$ | 3 | 6 | 2 | 0 | 3 | 7 | 8 | 3 | 0 |
| $\text{E}_{\text{min}}$ | 5 | 1 | 5 | 0 | 1 | 9 | 11 | 2 | 0 |
| $\text{E}_{\text{max}}$ | 6 | 8 | 6 | 1 | 2 | 19 | 13 | 3 | 0 |
| $\text{F}_{\text{min}}$ | 6 | 6 | 1 | 2 | 2 | 7 | 3 | 6 | 0 |
| $\text{F}_{\text{max}}$ | 6 | 6 | 1 | 2 | 3 | 11 | 6 | 12 | 6 |
| $\text{G}_{\text{min}}$ | 10 | 15 | 6 | 0 | 5 | 25 | 12 | 18 | 17 |
| $\text{G}_{\text{max}}$ | 10 | 16 | 10 | 0 | 5 | 33 | 19 | 18 | 17 |
| H | 2 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| I | 3 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| J | 2 | 1 | 2 | 0 | 0 | 3 | 2 | 0 | 0 |

Table 2: Network station statistics. Since the network and manually created station model may change over time, we give for each station and each quantity the corresponding minimal and maximal counts. If there are no changes in a station, we only give one set of values.

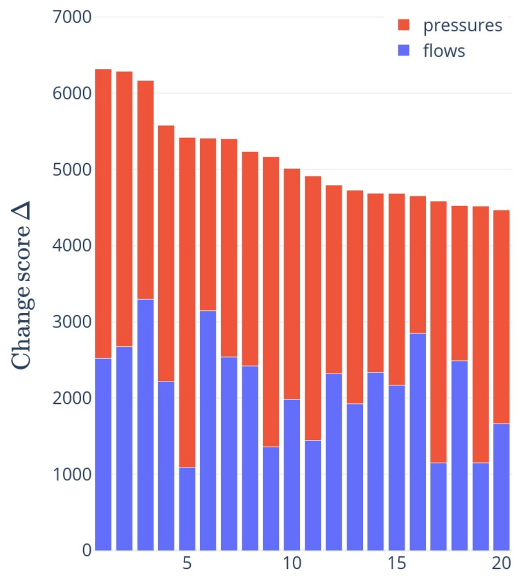
| | # continuous variables | # binary variables | # constraints |
|---|---|---|---|
| base | 56 – 68 | 61 – 72 | 236 – 278 |
| new | 14 – 17 | 54 – 65 | 151 – 184 |

Table 3: Artificial compressor model sizes of the base and our new model based on the used instance set. Numbers vary over time. The corresponding minimal and maximal numbers originate from the same point in the overall time period.

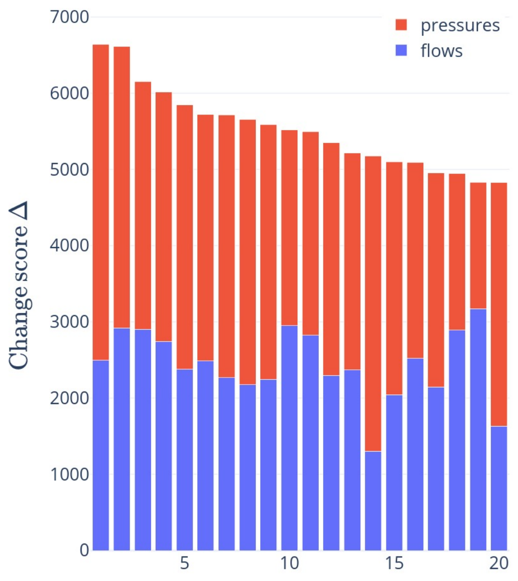
**Challenging demand scenarios** To properly examine if the presented algorithms are suitable for application in time-critical industry environments, we search for particularly challenging instances. In [Hoppmann-Baum et al., 2020], where the gas flow model of [Hoppmann-Baum et al., 2021] was used to conduct a case study on hydrogen transport, the authors suggest that the difficulty of the problem corresponds to the number of element control changes needed to fulfill the given demand scenario. As the number of changes is not known a priori, we instead look for demand scenarios featuring large demand changes for single boundary nodes since we expect these changes to induce inevitable changes in some network elements' control. From a cooperation with our project partner OGE, we attain historical network demands in hourly resolution over a time period of 12 months, covering the majority of the year 2020. The time period is nearly consecutive, having only three periods of 2, 6, and 11 missing days due to problems during the data creation.

To find demand scenarios with large demand changes from within this time period, we first specify the desired time horizon length for our instances to be 12 and 24 hours. For both, we then enumerate all time horizons of the corresponding length and sum the differences between the initial and end time point of the two quantities used as future demands in Section 2.8: The boundary node inflow values, which are given as norm volumetric flow $Q^0$ in $1000\text{m}^3/\text{h}$, and the pressure in bar for entries with positive inflow. The norm volumetric flow is defined as $Q^0 = q/\rho^0$, where $\rho^0$ is the norm density depending on the gas mixture. We then create a final score $\Delta$ for the amount of changes of a time horizon by adding the two sums, using the quantities' values for the given units, and weighting the pressure changes by a factor of 100. Finally, we choose from each of the two lists of time horizons a set of 20 demand scenarios by iteratively selecting from the remaining time horizons the one with the highest $\Delta$ value, which does not overlap with any of the already selected ones. Overlapping is hereby defined as sharing at least one full hour in the time horizon. If one time horizon ends at time point $t$ and a second one starts at $t$, they do not overlap.

An overview of the $\Delta$ values for each of the 20 demands scenarios for each time horizon is given in Figure 5. The largest found score value is about 6600, which would be equivalent to a total amount of inflow changes of $6600\,1000\text{m}^3/\text{h}$ or a total amount of pressure changes of 66 bar at entry nodes over the whole network. The 40 demand scenarios, together with the corresponding network topology and initial network state at the start of the time horizon, define the 40 instances we use as our test set for the computational experiments. When using a time horizon whose time steps are larger than 1 hour, we aggregate the hourly demands in the same way as in the function `createTimeAggregatedProblemInstance` used in the aggregated horizon heuristic presented in Algorithm 4.



(a) 12h horizon



(b) 24h horizon

Figure 5: Change scores of the 20 non-overlapping demand scenarios with the highest change score $\Delta$ for each of the two lengths of the time horizon. Each bar represents one selected instance and is divided according to the pressure and flow share of $\Delta$, where the pressure is scaled by 100. The instances are sorted by change score. Figures created with [Plotly Technologies Inc., 2015].

## 4.2 Setup and parameters

In this section, we give an overview of the computational hardware setup and solver choice, the applied model adjustments to improve its numerical properties, and specify the parameter used for the computational experiments.

### 4.2.1 Computational setup

The computations were executed on a cluster using 4 cores and 40 GB of RAM of a machine composed of two *Intel Xeon Gold 5122* running at 3.60 GHz. The MIP problems are solved by Gurobi in version 9.5.1 [Gurobi Optimization, LLC., 2021], for which we change the general solution strategy to focus on finding primal solutions by setting the `MIPFocus` parameter to 1. As the NLP solver, we used Ipopt in version 3.14.3 [Wächter and Biegler, 2006]. Finally, we used Scip in version 8.0.0 [Bestuzheva et al., 2021] as the global MINLP solver. For Scip, we change the feasibility tolerance for constraints `numerics/feastol` to $10^{-3}$, which is even higher than the corresponding default value of $10^{-4}$ for Ipopt, to improve the acceptance rate for start solutions. We accessed both Ipopt and Scip via Gams in version 38.1.0 [GAMS Development Corporation, 2022].

### 4.2.2 Coefficient scaling and rounding

To improve the numeric properties of the models, we use similar ranges for the continuous variables by scaling them such that pressure variables are used in bar and flow values in kg/s. Furthermore, we apply the following modification to the pipes' Continuity equation (2a), their Momentum equation (2b) in the original or the linearized variant, and the linearized maximum power constraint of artificial configurations (11c): First, we set all coefficients to zero, whose absolute value divided by the largest absolute value coefficient is smaller than $10^7$. Afterwards, we scale all coefficients of the corresponding constraint such that the smallest non-zero coefficient has an absolute value of 1.

#### 4.2.3 Parameters

In the following, we list the parameters used in the model, in the heuristic algorithms, and as resources limits for the different solvers.

**Model parameters** For the model, we specify the maximum slack parameters to be 2 bar for the pressure slack. In the case of the inflow, we restrict the possible maximal deviation from the demands to 50%. Hence, we use, instead of a fixed upper bound $\bar{\sigma}^d$ on the inflow slack, an upper bound specific to each boundary node and time step of

$$\bar{\sigma}^d_{v,t} = 0.5 \cdot |\hat{d}_{v,t}|$$

for all boundary nodes $v \in \mathcal{V}^{\mathrm{b}}$ and future time steps $t \in \mathcal{T}$. Furthermore, we set the minimum flow value used to determine the absolute velocity in the initial linearization of the friction term to $q^{\min} = 100 \cdot 1000\mathrm{m}^3/\mathrm{h}$.

We define the different weights used in the objective function defined in Equation (30) as

$$w^{\sigma\text{-d}} = 100 \cdot 3.6/\rho^0 \qquad w^{\sigma\text{-p}} = 1000.0 \qquad w^{\mathrm{ru}} = w^{\mathrm{arc}} = 50.0$$
$$w^{\mathrm{va}} = w^{\mathrm{rg}} = 500.0 \qquad w^{\text{rg-pl}} = w^{\text{rg-pr}} = w^{\text{fn-p}} = 10.0 \qquad w^{\text{rg-q}} = w^{\text{fn-q}} = 1.0 \cdot 3.6/\rho^0$$
$$w^{\mathrm{st}}_s \in [0, 4000].$$

Note that the multipliers for the flow change and the hourly flow slack are given with respect to norm volumetric flow $Q^0$ in $1000\mathrm{m}^3/\mathrm{h}$. Hence, we multiply them by a factor of $3.6/\rho^0$ to get the corresponding mass flow equivalent. The costs for turning on a simple state of a network station are manually created by the industry experts at OGE, which is why we can only give the range of values for them.

**Algorithm parameters** Each heuristic algorithm defined in Section 3 is given a set of parameters to adjust them. First, we specify the converge tolerances for the friction term linearization as

$$n^{\mathrm{tolRel}} = 0.001 \qquad n^{\mathrm{tolAbs}} = 0.01\,\mathrm{bar}.$$

Next, the maximum number of sequential mixed-integer programming iterations we use is

$$n^{\mathrm{it}} = n^{\mathrm{itAgg}} = 5 \qquad n^{\mathrm{itFull}} = 3.$$

Finally, we do not fix the parameters defining the reduced time horizon size globally but define different variants of the heuristics algorithms based on these. Hence, we denote the rolling horizon heuristic of Algorithm 3 with a rolling horizon size of $n^{\mathrm{rolHor}} = X$ by `RXH`, the aggregated horizon heuristic of Algorithm 4 with aggregated horizon size of $n^{\mathrm{aggHor}} = X$ by `AXH`, and the aggregated and rolling horizon heuristic of Algorithm 5 using an aggregated horizon of length $n^{\mathrm{aggHor}} = X$ and a rolling horizon of length $n^{\mathrm{rolHor}} = Y$ by `AXRYH`.

**Resource limits** As final parameters, we define the resource limits of the different solving processes, which are given in Table 4. There we distinguish five different model variants: MIP with the dynamic node limit defined in Section 3.3, MIP without this node limit, an NLP run to complete binary assignments, a *fast* NLP variant having a time limit of 5 minutes, and finally, the MINLP run. The given parameters for the dynamic node limit specify that the smallest node limit to stop is 1000 and that a significant primal solution upgrade featuring an objective function value improvement of at least 2% increases the node limit to 150% of the current number of processed nodes. Note that all variants are additionally restricted by the general memory limit of 40 GB.

### 4.3 Solution quality of general `SMIP`

In our first experiment, we check the quality of solutions obtained by using the general sequential mixed-integer programming `SMIP` Algorithm 2 to determine binary assignments in Algorithm 1 and complete these to a full MINLP solution by an NLP solver. The lower bound to compare the solutions against is obtained via a global MINLP solver. Due to the size of our network and the general complexity of the

| Variant | Rel. Gap | Time limit | $n^{\text{limitMin}}$ | $n^{\text{limitRel}}$ | $n^{\text{improve}}$ |
|---|---|---|---|---|---|
| MIP w/node limit | 0.0 | $\infty$ | 1000 | 1.5 | 0.02 |
| MIP wo/node limit | $10^{-2}$ | 86400 s | – | – | – |
| NLP | – | $\infty$ | – | – | – |
| NLP fast | – | 300 s | – | – | – |
| MINLP | $10^{-6}$ | 600000 s | – | – | – |

Table 4: Optimality conditions and resource limits for the different model variants. We distinguish solving a MIP with and without the dynamic node limit defined in Section 3.3, solving an NLP with and without a time limit, and solving an MINLP.

problem, we were only able to obtain reasonable lower bounds for the smallest time horizon of size 1. Depending on the actual instance, the corresponding MINLP model has the following sizes (rounded to 2 significant digits):

$$\# \text{ variables} \in [2300, 2900] \qquad \# \text{ binary variables} \in [690, 870]$$
$$\# \text{ constraints} \in [3100, 3800] \qquad \# \text{ non-linear constraints} \in [300, 390].$$

Since the overall run time of the solving process has no priority for this experiment, we use for the single MIP models the parameter variant without the heuristic node limit, and for the NLP the general variant without a time limit, see Table 4. Also, we use the solution obtained from the `SMIP` algorithm as a starting solution for the MINLP.

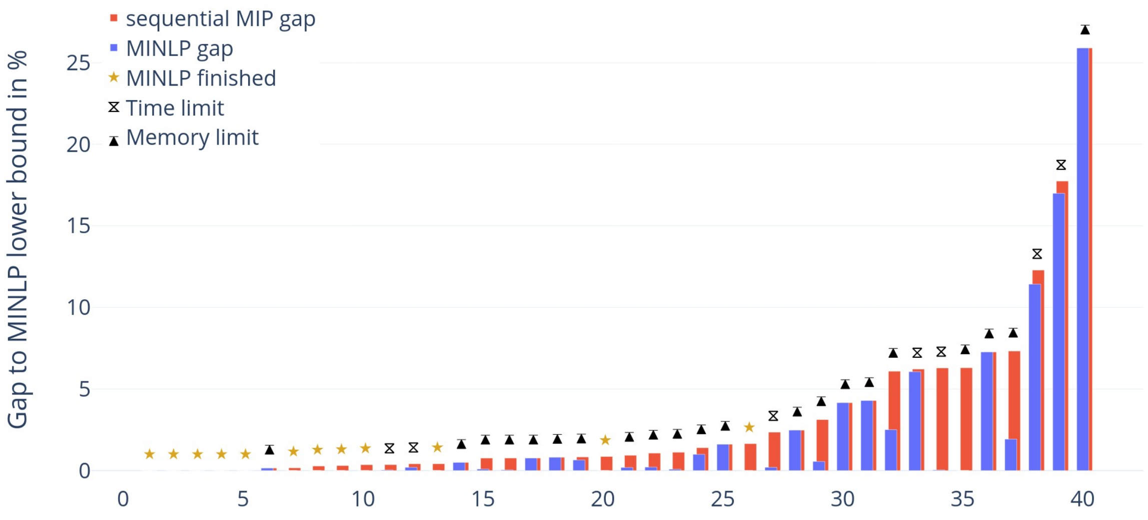


Figure 6: Comparison of the gap of the solutions created by the `SMIP` Algorithm 2 used in Algorithm 1 with respect to the lower bound found in the MINLP run and the gap of the MINLP itself. Each pair of bars represents one instance, sorted by the gap of the `SMIP` solution. The values are given in percent. For the instances marked by a star, the MINLP proved optimality. For those instances marked by an hourglass, the MINLP time limit was hit, while for instances marked by a black triangle pointing upwards to a horizontal bar, the MINLP hit the 40 GB memory limit. Figure created with [Plotly Technologies Inc., 2015].

The result is given in Figure 6. First, we notice that for only 12 out of the 40 instances, the MINLP was solved to optimality. In 7 cases, the time limit of roughly one week was hit, while for the remaining 21 instances, the memory limit of 40 GB was exceeded. As a consequence, we do not always have the best possible lower bound available. However, the average gap of 3.2% of the `SMIP` solutions to that lower bound is still rather small and even closer to the average gap of 2.3% of the MINLP itself. Using the best found primal solutions of the MINLP as a reference for the highest possible lower bound values, the average gap for the `SMIP` solutions is 0.9%. Regarding run time, Algorithm 1 using the `SMIP` always finishes in less than 30 seconds, while the geometric mean run time of the MINLP is larger than a day. In summary, the `SMIP` solutions are very close to the global optimum and can be computed in only a fraction of the global solver's execution time.

## 4.4 Performance of reduced time horizon heuristics

In our main experiment of the paper, we evaluate our proposed algorithm of using a reduced-time-horizon heuristic of Section 3.2 for creating binary assignments to be completed in Algorithm 1. As a reference approach, we use the `SMIP` for the binary assignment creation, which was shown to find close-to-optimal solutions in the previous section. Not having to compare against a global MINLP solver enables us to solve instances with a time horizon consisting of 12 equally large time steps of 1-hour length for the 12-hour horizon and 2-hour length for the 24-hour horizon. Depending on the actual instance, the MINLP model, which is never formulated and tackled as a whole model here, has the following sizes (rounded to 2 significant digits):

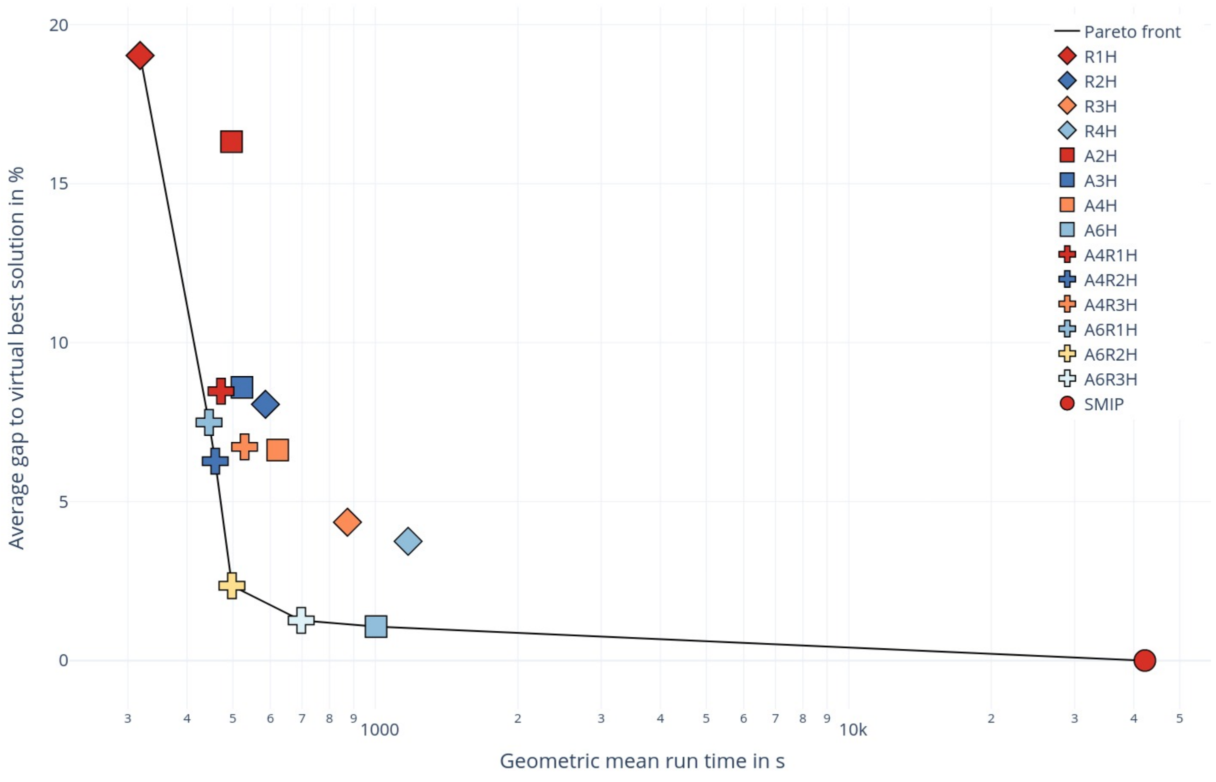
$$\# \text{ variables} \in [25000, 32000] \qquad \# \text{ binary variables} \in [5600, 7000]$$
$$\# \text{ constraints} \in [35000, 44000] \qquad \# \text{ non-linear constraints} \in [3600, 4700].$$

After conducting some preliminary experiments, we determined for each of the reduced horizon heuristics the smallest time horizon lengths, which still give reasonably good results, and the largest time horizon lengths, at which the run times are still reasonably short. As a result, we choose the following variants to test for each of the heuristics: The rolling horizon heuristics from `R1H` to `R4H`, the aggregated time horizon heuristics `A2H`, `A3H`, `A4H`, and `A6H`, and for the last two aggregated horizon lengths also the heuristic using the rolling horizon heuristic for some of the sub-models with horizon lengths of 1 to 3, resulting in the variants `A4R1H`, `A4R2H`, `A4R3H`, `A6R1H`, `A6R2H`, and `A6R3H`. For all variants, we solve the MIP models using the dynamic branch-and-bound node limit setting and solve the NLP using a time limit, see Table 4. For the reference `SMIP` approach, we ensure the highest possible solution quality by not using the node limit for MIP or the time limit for NLP.

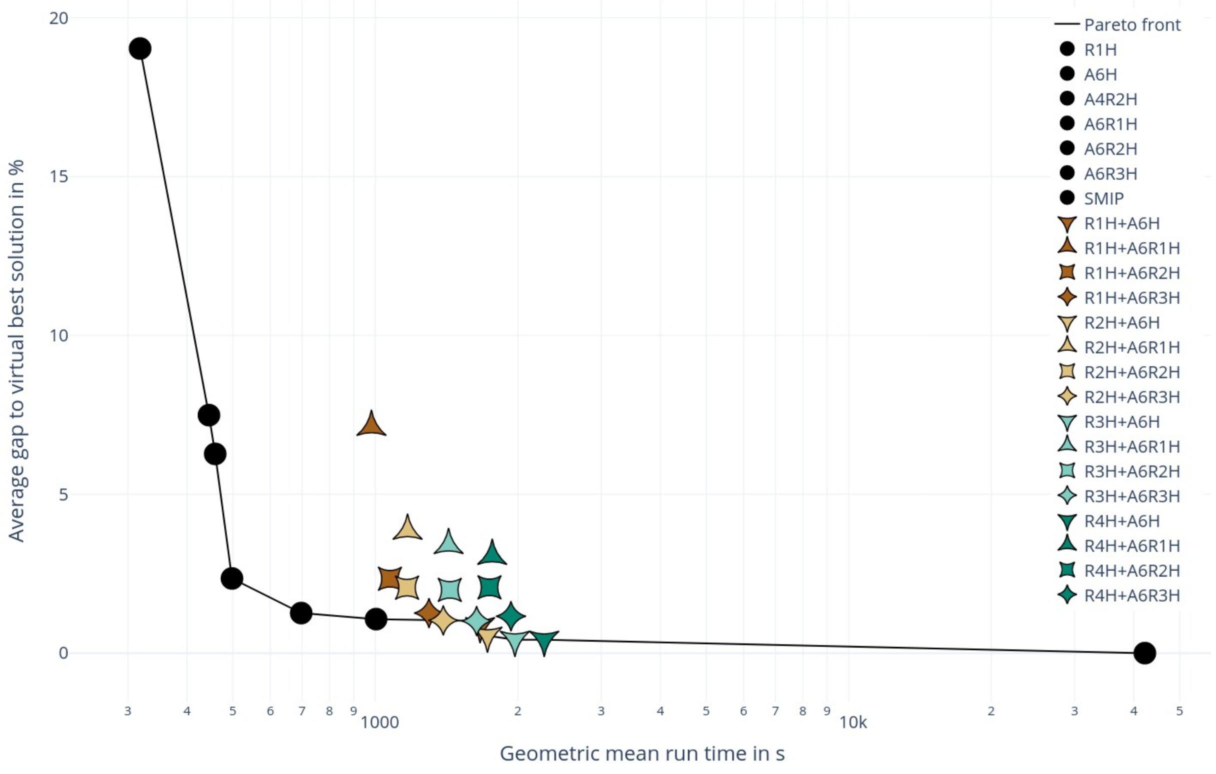
| | | # avg MIP runs | | | | # avg NLP runs | | | | mean NLP time | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | fail | total | opt | DNL | inf | total | conv | time | fail | by NLP | by inst |
| `R1H` | 4 | 46.73 | 46.62 | 0.00 | 0.10 | 3.38 | 3.12 | 0.05 | 0.20 | 161.6 | 307.4 |
| `R2H` | 1 | 46.08 | 45.35 | 0.70 | 0.03 | 3.67 | 3.58 | 0.03 | 0.07 | 132.8 | 418.6 |
| `R3H` | 0 | 44.00 | 41.42 | 2.58 | 0.00 | 3.88 | 3.75 | 0.05 | 0.07 | 130.3 | 474.9 |
| `R4H` | 0 | 38.25 | 30.80 | 7.45 | 0.00 | 3.77 | 3.70 | 0.03 | 0.05 | 120.8 | 430.9 |
| `A2H` | 1 | 10.40 | 8.82 | 1.48 | 0.10 | 4.55 | 4.45 | 0.05 | 0.05 | 86.6 | 258.3 |
| `A3H` | 1 | 11.22 | 8.82 | 2.33 | 0.07 | 4.65 | 4.65 | 0.00 | 0.00 | 78.3 | 246.6 |
| `A4H` | 0 | 11.68 | 8.90 | 2.67 | 0.10 | 4.55 | 4.55 | 0.00 | 0.00 | 77.5 | 278.0 |
| `A6H` | 0 | 12.62 | 9.15 | 3.42 | 0.05 | 4.90 | 4.85 | 0.00 | 0.05 | 76.6 | 315.6 |
| `A4R1H` | 0 | 24.55 | 24.48 | 0.07 | 0.00 | 4.78 | 4.78 | 0.00 | 0.00 | 71.4 | 278.6 |
| `A4R2H` | 0 | 19.02 | 18.62 | 0.40 | 0.00 | 4.47 | 4.42 | 0.00 | 0.05 | 75.5 | 245.8 |
| `A4R3H` | 0 | 15.03 | 12.57 | 2.42 | 0.03 | 4.47 | 4.45 | 0.03 | 0.00 | 75.9 | 260.5 |
| `A6R1H` | 1 | 33.77 | 33.75 | 0.00 | 0.03 | 5.35 | 5.33 | 0.00 | 0.03 | 74.4 | 285.5 |
| `A6R2H` | 0 | 28.12 | 27.38 | 0.72 | 0.03 | 4.75 | 4.75 | 0.00 | 0.00 | 70.3 | 266.8 |
| `A6R3H` | 0 | 25.23 | 22.52 | 2.70 | 0.00 | 5.08 | 5.05 | 0.00 | 0.03 | 76.9 | 324.8 |

Table 5: Statistics for the used heuristic variants. The first two columns state the name of the heuristic and the number of instances for which it fails to find a feasible solution. The next four columns state the average number of MIP models solved per instance, followed by the corresponding average number of those MIPs finished with an optimal solution, finished due to reaching the dynamic node limit, or finished with proven infeasibility of the corresponding model. Next, we have a similar block of four columns for the average number of NLP models solved per instance. First, the total average is stated (which is equal to the average number of binary assignments found), then the average number of converged NLPs, the average number of NLP finishing with a feasible solution at the given time limit, and finally, the average number of failed attempts to find a feasible solution within the time limit. The final two columns depict the geometric mean run time for the NLP in seconds, first for a single NLP run and second for the summed run time of all NLP runs per instance. If a heuristic did not produce any binary assignments for an instance, the NLP run time was set to 1 second.

We present the results averaged over all instances for each heuristic variant in two pairs of figures. Each figure shows the position of each variant with respect to the run time and the average gap to the virtual best solution. In the Figures 7, the run time is determined as the geometric mean run time of all

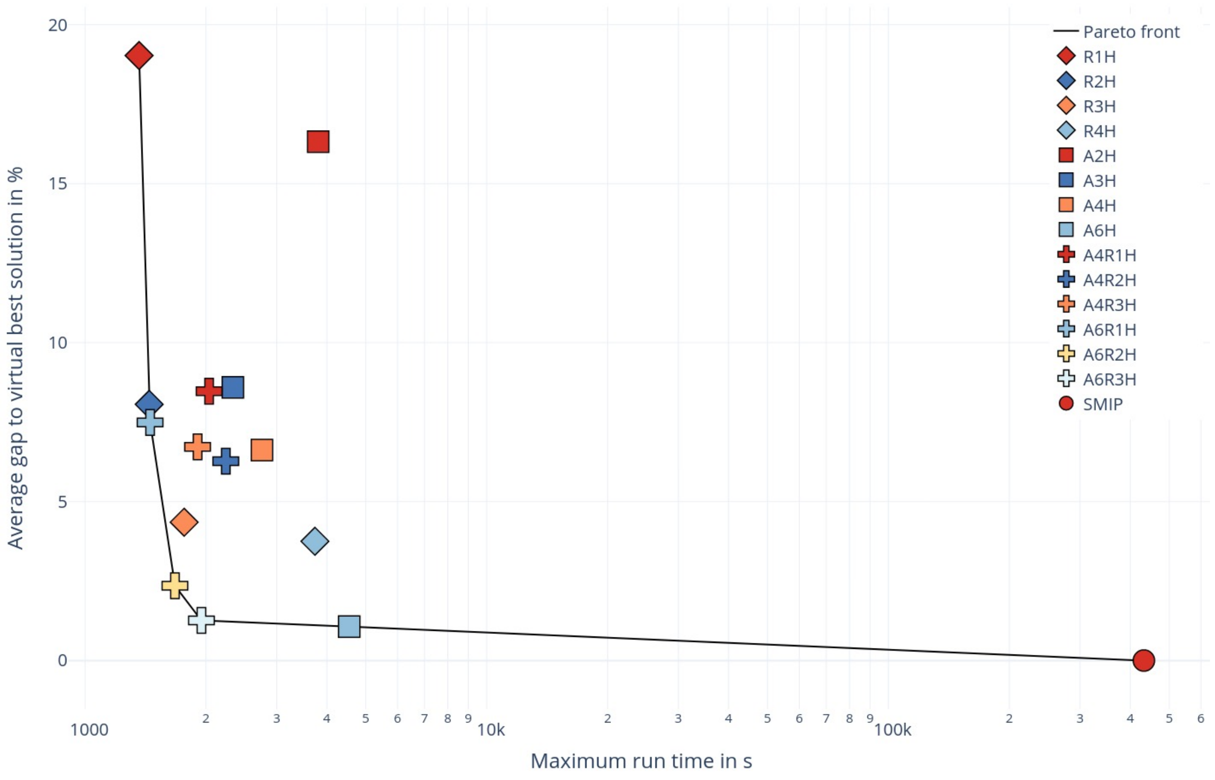


(a) Individual heuristics

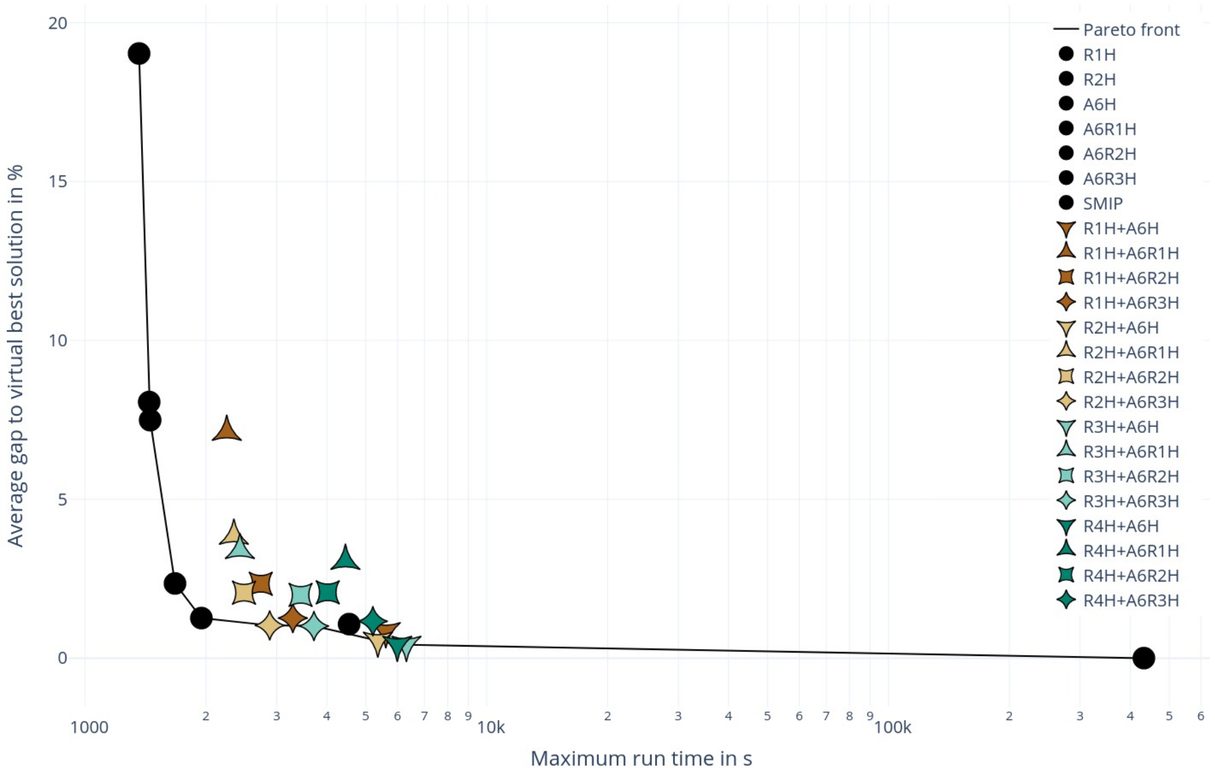


(b) Combine heuristics and Pareto points of the individual heuristics

Figure 7: Position of the heuristic variants in a geometric-mean-run-time vs. gap-to-virtual-best-solution plot. If a heuristic does not find a solution for some instance, we count the corresponding gap as 100%. The Pareto efficient points are linked by a black line, forming the Pareto front. Figures created with [Plotly Technologies Inc., 2015].



(a) Individual heuristics



(b) Combine heuristics and Pareto points of the individual heuristics

Figure 8: Position of the heuristic variants in a maximum-run-time vs. gap-to-virtual-best-solution plot. If a heuristic does not find a solution for some instance, we count the corresponding gap as 100%. The Pareto efficient points are linked by a black line, forming the Pareto front. Figures created with [Plotly Technologies Inc., 2015].

the instances and in the Figures 8, we use the maximum run time. If a heuristic failed to find a solution for a specific instance, we used a gap value of 100%. In Table 5, we gave an overview of the number of fails as well as the average results for the single MIP solves and NLP solves per heuristic variant. In the figures, we connected those heuristics being Pareto efficient by a black line, where Pareto efficient heuristics are those not having another heuristic with a smaller average gap and, at the same time, a shorter run time value.

In addition to the individual heuristics displayed in the upper Figures 7a and 8a, we also present in the lower Figures 7b and 8b results for the *combination* of two different heuristic variants. For each combination and instance, we use the best solution found in any of the two heuristics and add up the overall run time of both approaches. As a reference, we also included all Pareto efficient individual heuristics from the corresponding upper pictures. For the combination, we always choose a pair of a pure rolling horizon heuristic and a heuristic using an aggregated time horizon. We do this since we expect the two fundamentally different approaches to have a higher probability complementing each each other in finding high-quality solutions. For the combination, we chose all four rolling horizon heuristics. For the time-aggregation-based heuristics, we used those that are part of the Pareto front for the geometric mean and the maximum run time figures, i.e., `A6H`, `A6R1H`, `A6R2H`, and `A6R3H`. Note that the combined heuristics in the figures have the same color for using the same rolling horizon heuristics and the same symbol for using the same time-aggregation-based heuristic.

The figures show that the reference `SMIP` approach is the overall best approach in terms of solution quality with an average gap of 0.0%, as it yields the best solution for all of the instances. However, it is much slower than every of the reduce time horizon heuristics both on average and regarding the maximum run time. This is mainly due to the fact that, on average, 1.5 out of the at most 5 solved MIP models reached the 1-day time limit before proofing optimality. For the reduced time horizon heuristics, the solution quality is, in general, very good: The average gap to the virtual best solution is below 20% for all considered variants and even below 10% for all but two. The best average gap apart from the general `SMIP` approach was achieved by the `A6H` heuristic with a gap of 1.07% and 17/40 instances, for which the best solution has been found. Two very close variants are `A6R2H` and `A6R3H`, which are close in terms of the average gap but have better run time values. Hence, the summary for the individual variants is to choose an aggregated horizon heuristic with a long time horizon and either solve it directly for the best results or internally use the rolling horizon approach for shorter run times at the cost of a slightly worse solution quality. When comparing the geometric mean and the maximum run time, the pictures are very similar, with only two heuristic variants being in only one of the corresponding Pareto fronts. The maximum run time values are between 2 to 5 times higher for all the heuristics except `A2H`, where the factor is over 7. For the two variants `A6R2H` and `A6R3H` with the lowest run times of those with an average gap below 3%, the maximum run time is about half an hour, which is very fast for these types of problems.

Using two heuristics in combination improves the overall solution quality as expected, with 6 of the 16 combinations having an average gap better than the best individual solution and all of the combinations involving `A6H` reaching an average gap below 1%. However, this comes at the cost of rather large run time increases. Only the fastest of the combined heuristics has a better geometric run time than the one of the slowest Pareto efficient reduced time horizon heuristic. However, in that larger run time segment, several Pareto efficient combinations of the `A6R3H` and `A6H` heuristic combined with the rolling horizon heuristics can be found for both run time metrics. The best of them decrease the gap to the `SMIP` variant to a minimum of 0.42% for the two combinations of `R3H` and `R4H` with `A6H` while still being much faster than `SMIP` itself.

The reason for the overall bad run time for the combination of two heuristics is the rather long solve time for the NLP model, as displayed in the last two columns of Table 5. For heuristics like `R1H` and `R2H` that determine binary assignments very fast, the geometric mean NLP time per instance is equal to 96.5% and 71.5% of the geometric mean overall run time, respectively. Hence, testing more binary assignments for yielding good overall solutions to the problem is apparently a considerable time investment.

To compare our algorithm against a procedure similar to the one proposed in [Hoppmann-Baum et al., 2021], we also tested the `SMIP` approach with only a single iteration, i.e., $n^{\text{it}} = 1$. This is close to the algorithm proposed there, as the linearization used for the model determining the binary solution values is only updated in the rare case of failure in the subsequent procedure searching for a non-linear solution for the found binary assignment. The `SMIP` heuristic with $n^{\text{it}} = 1$ has an average gap of 42.89% to the virtual best solution and is, therefore, worse than all of our proposed heuristic variants. For 14 out of the overall 40 instances, it reached the time limit of 1 day. This leads to a geometric mean run time of 9541.0 seconds, which is again considerably larger than those of all our heuristics variants. Hence, we can conclude that our proposed algorithms are clearly superior.

## 5 Outlook

In this article, we presented several improvements for solving the control optimization problem on transient gas networks in a time-critical environment. First, we introduced a new model for the artificial arcs representing the compressors, which is configuration-based and closer to the original compression capabilities while using fewer variables and constraints. Furthermore, we propose a new algorithm that finds better solutions faster. It first determines a set of solution values for all binary variables, which we call a binary assignment, and afterwards solves a corresponding non-linear program to complete it to a full solution of the problem. The binary assignments are obtained based on sequential mixed-integer programming, two heuristics based on reduced time horizons, and a specialized dynamic node limit. Our approach was tested on particularly challenging real-world instances featuring a large amount of supply and demand changes. The results confirm that we are able to find solutions close to the global optimum while being very fast compared to black-box MINLP solvers and comparable approaches from the literature.

We see two main reasons for the success of our heuristic approach for the given problem. First, we recognized that from the multitude of possible future control recommendations, only a few combinations fit a given scenario. This is especially true for the given objective function that mainly punishes control modes and network state changes. Second, the network's pipes act as gas storage and can therefore compensate for a local shortage or surplus of gas for a short amount of time. For this reason, changing the control of an element a little too early or too late is often feasible and only causes a small objective function value penalty. This helps for both of the heuristics: The rolling horizon heuristic results in too late control changes due to not having a full view of the future. In the case of the aggregated horizon, the time points in which control decisions can be made are limited. Hence, they may be a little late or a little early. Due to the pipe storage, solutions with these control time offsets are likely to be similar to one of the few reasonable control recommendations while only having a slightly worse objective function value.

Future research on the overall topic can continue in many directions. Model-wise, we could further refine the compressor model to guarantee that the corresponding feasible region is an actual relaxation of the original feasible region. Another possibility to improve the accuracy would be to also update the linearization of the power bound during the sequential mixed-integer programming and include the original non-linear formula in the final non-linear programming model. However, as we have seen in the results, the run time for solving the NLP models is already quite high. Therefore, it might be beneficial to just complete the most promising binary assignments to full solutions, for example, by comparing their corresponding linearized MIP solution value, or to only complete one of a set of similar binary assignments. Regarding the instance set used in our computational experiments, it would be interesting to examine if heterogeneous step lengths in the time horizon lead to more complex solutions or to evaluate our algorithm in the context of pure hydrogen networks similar to those presented in [Hoppmann-Baum et al., 2020].

## Acknowledgements

The work for this article has been conducted in the Research Campus MODAL, funded by the German Federal Ministry of Education and Research (BMBF) (fund numbers 05M14ZAM & 05M20ZBM).

## References

T. Berthold, G. Hendel, and T. Koch. From feasibility to improvement to proof: Three phases of solving mixed-integer programs. *Optimization Methods and Software*, 33(3):499–517, May 2018. ISSN 1055-6788. doi: 10.1080/10556788.2017.1392519.

K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske,

F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig. The SCIP optimization suite 8.0. ZIB-Report 21-41, ZIB, Takustr. 7, 14195 Berlin, Dec. 2021.

N. Boland, A. Ernst, T. Kalinowski, M. Rocha de Paula, M. Savelsbergh, and G. Singh. Time Aggregation for Network Design to Meet Time-Constrained Demand. In *MODSIM 2013: 20th International Congress on Modelling and Simulation - Adapting to Change: The Multiple Roles of Modelling*, pages 3281–3287. Modelling and Simulation Society of Australia and New Zealand, 2013. ISBN 978-0-9872143-3-1.

P. Bonami, A. Lodi, A. Tramontani, and S. Wiese. On mathematical programming with indicator constraints. *Mathematical Programming*, 151(1):191–223, June 2015. ISSN 1436-4646. doi: https://doi.org/10.1007/s10107-015-0891-4.

R. Burlacu, H. Egger, M. Groß, A. Martin, M. Pfetsch, L. Schewe, M. Sirvent, and M. Skutella. Maximizing the storage capacity of gas networks: A global MINLP approach. *Optimization and Engineering*, 20(2):543–573, June 2019. doi: https://doi.org/10.1007/s11081-018-9414-5.

F. Capitanescu. A relax and reduce sequential decomposition rolling horizon algorithm to value dynamic network reconfiguration in smart distribution grid. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6, Sept. 2017. doi: 10.1109/ISGTEurope.2017.8260094.

P. Domschke, B. Geißler, O. Kolb, J. Lang, A. Martin, and A. Morsi. Combination of Nonlinear and Linear Optimization of Transient Gas Networks. *INFORMS Journal on Computing*, 23(4):605–617, 2011. doi: 10.1287/ijoc.1100.0429.

P. Domschke, B. Hiller, J. Lang, V. Mehrmann, R. Morandin, and C. Tischendorf. Gas Network Modeling: An Overview. Technical Report, Technische Universität Darmstadt, 2021.

Federal Ministry for Economic Affairs and Climate Action. KfW, Gasunie and RWE sign MoU to build an LNG terminal at Brunsbüttel. `https://www.bmwi.de/Redaktion/EN/Pressemitteilungen/2022/03/202203-kfw-gasunie-and-rwe-sign-mou-to-build-an-lng-terminal-at-brunsbuettel.html`, 2022. Accessed: 2022-03-24.

FNB Gas – Association of German transmission system operators. Scenario Framework Gas Network Development Plan 2022–2032. `https://fnb-gas.de/en/scenario-framework/scenario-framework-2022/`, Aug. 2021. Accessed: 2022-03-24.

A. Fügenschuh, B. Geißler, R. Gollmer, A. Morsi, M. E. Pfetsch, J. Rövekamp, M. Schmidt, K. Spreckelsen, and M. C. Steinbach. Physical and technical fundamentals of gas networks. In Koch et al. [2015].

GAMS Development Corporation. General Algebraic Modeling System (GAMS) Release 38.1.0. Fairfax, VA, USA, Jan. 2022.

L. Glomb, F. Liers, and F. Rösel. A rolling-horizon approach for multi-period optimization. *European Journal of Operational Research*, 300(1):189–206, July 2022. ISSN 0377-2217. doi: 10.1016/j.ejor.2021.07.043.

Á. M. González Rueda, J. González Díaz, and M. P. Fernández de Córdoba. A twist on SLP algorithms for NLP and MINLP problems: An application to gas transmission networks. *Optimization and Engineering*, 20(2):349–395, June 2019. ISSN 1573-2924. doi: 10.1007/s11081-018-9407-4.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, version 9.5. https://www.gurobi.com, 2021.

M. Hahn, S. Leyffer, and V. M. Zavala. Mixed-Integer PDE-Constrained Optimal Control of Gas Networks. *Preprint.*, 2017.

F. Hennings. Benefits and Limitations of Simplified Transient Gas Flow Formulations. In *Operations Research Proceedings 2017*, pages 231–237. Springer International Publishing, 2018. ISBN 978-3-319-89920-6. doi: 10.1007/978-3-319-89920-6_32.

F. Hennings. Large-scale empirical study on the momentum equation's inertia term. *Journal of Natural Gas Science and Engineering*, 95:104153, July 2021. ISSN 1875-5100. doi: 10.1016/j.jngse.2021.104153.

F. Hennings, L. Anderson, K. Hoppmann-Baum, M. Turner, and T. Koch. Controlling transient gas flow in real-world pipeline intersection areas. *Optimization and Engineering*, 22(2):687–734, June 2021a. ISSN 1573-2924. doi: 10.1007/s11081-020-09559-y.

F. Hennings, M. Petkovic, and T. Streubel. On the Numerical Treatment of Interlaced Target Values - Modeling, Optimization and Simulation of Regulating Valves in Gas Networks. ZIB-Report 21-32, ZIB, Takustr. 7, 14195 Berlin, Dec. 2021b.

K. Hoppmann-Baum. Mathematical programming for stable control and safe operation of gas transport networks. (unpublished manuscript), 2022.

K. Hoppmann-Baum, F. Hennings, J. Zittel, U. Gotzes, E.-M. Spreckelsen, K. Spreckelsen, and T. Koch. From Natural Gas towards Hydrogen - A Feasibility Study on Current Transport Network Infrastructure and its Technical Control. ZIB-Report 20-27, ZIB, Takustr. 7, 14195 Berlin, 2020.

K. Hoppmann-Baum, F. Hennings, R. Lenz, U. Gotzes, N. Heinecke, K. Spreckelsen, and T. Koch. Optimal Operation of Transient Gas Transport Networks. *Optimization and Engineering*, 22(2):735–781, June 2021. ISSN 1573-2924. doi: 10.1007/s11081-020-09584-x.

T. Koch, B. Hiller, M. E. Pfetsch, and L. Schewe, editors. *Evaluating Gas Network Capacities*, volume 21 of *MOS-SIAM Series on Optimization*. SIAM, 2015.

O. Kolb, J. Lang, and P. Bales. An implicit box scheme for subsonic compressible flow with dissipative source term. *Numerical Algorithms*, 53(2):293–307, Mar. 2010. ISSN 1572-9265. doi: 10.1007/s11075-009-9287-y.

R. Lenz. *Optimization of Stationary Expansion Planning and Transient Network Control by Mixed-Integer Nonlinear Programming*. Doctoral Thesis, Technische Universität Berlin, Berlin, 2021.

D. Mahlke, A. Martin, and S. Moritz. A simulated annealing algorithm for transient optimization in gas networks. *Mathematical Methods of Operations Research*, 66(1):99–115, Aug. 2007. ISSN 1432-5217. doi: 10.1007/s00186-006-0142-9.

S. Moritz. *A Mixed Integer Approach for the Transient Case of Gas Network Optimization*. Doctoral Thesis, Technische Universität Darmstadt, Darmstadt, 2007.

A. M. Newman and M. Kuchta. Using aggregation to optimize long-term production planning at an underground mine. *European Journal of Operational Research*, 176(2):1205–1218, Jan. 2007. ISSN 0377-2217. doi: 10.1016/j.ejor.2005.09.008.

J. Nikuradse. *Laws of Flow in Rough Pipes*. National Advisory Committee for Aeronautics Washington, 1950.

OGE. Open Grid Europe GmbH. https://oge.net.

A. J. Osiadacz. Different Transient Flow Models - Limitations, Advantages, And Disadvantages. In *PSIG-9606*. Pipeline Simulation Interest Group, 1996.

J. Pápay. A termeléstechnológiai paraméterek változása a gáztelepek müvelése során. *OGIL Müsz. Tud. Közl.*, pages 267–273, 1968.

M. E. Pfetsch, A. Fügenschuh, B. Geißler, N. Geißler, R. Gollmer, B. Hiller, J. Humpola, T. Koch, T. Lehmann, A. Martin, A. Morsi, J. Rövekamp, L. Schewe, M. Schmidt, R. Schultz, R. Schwarz, J. Schweiger, C. Stangl, M. C. Steinbach, S. Vigerske, and B. M. Willert. Validation of Nominations in Gas Network Optimization: Models, Methods, and Solutions. *Optimization Methods and Software*, 30(1):15–53, 2015. ISSN 1055-6788. doi: 10.1080/10556788.2014.888426.

Plotly Technologies Inc. Collaborative data science. https://plot.ly, 2015.

K. Pratt and J. Wilson. Optimisation of the operation of gas transmission systems. *Transactions of the Institute of Measurement and Control*, 6(4):261–269, Sept. 1984. ISSN 0142-3312. doi: 10.1177/014233128400600411.

G. Takacs. Comparing methods for calculating Z-factor. *Oil Gas Journal*, 87(20):43–46, May 1989.

L. Tiacci and S. Saetta. Demand forecasting, lot sizing and scheduling on a rolling horizon basis. *International Journal of Production Economics*, 140(2):803–814, Dec. 2012. ISSN 0925-5273. doi: 10.1016/j.ijpe.2012.02.007.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar. 2006. ISSN 1436-4646. doi: 10.1007/s10107-004-0559-y.

T. Walther and B. Hiller. Modelling compressor stations in gas networks. ZIB-Report 17-67, ZIB, Takustr. 7, 14195 Berlin, 2017.

# A Appendix

| Variable | Meaning | Unit |
|---|---|---|
| $p_{v,t} \in \mathbb{R}_{\geq 0}$ | pressure at node $v \in \mathcal{V}$ | bar |
| $q_{v,a,t} \in \mathbb{R}$ | flow into or out of pipe $a = (l, r) \in \mathcal{A}^{\text{pi}}$ at end node $v \in \{l, r\}$ | kg/s |
| $q_{a,t} \in \mathbb{R}$ | flow over arc $a \in \mathcal{A} \setminus \mathcal{A}^{\text{pi}}$ | kg/s |
| $d_{v,t} \in \mathbb{R}$ | inflow into boundary node $v \in \mathcal{V}^{\text{b}}$ | kg/s |
| $m^{\text{op}}_{a,t} \in \{0,1\}$ | selection of open mode for $a \in \mathcal{A}^{\text{va}} \cup \mathcal{A}^{\text{rg}}$ | 1 |
| $m^{\text{cl}}_{a,t} \in \{0,1\}$ | selection of closed mode for $a \in \mathcal{A}^{\text{rg}}$ | 1 |
| $m^{\text{ac}}_{a,t} \in \{0,1\}$ | selection of active mode for $a \in \mathcal{A}^{\text{rg}}$ | 1 |
| $x^{\text{arc}}_{a,t} \in \{0,1\}$ | activity of artificial arc $a \in \mathcal{A}^{\text{ar}}$ | 1 |
| $x^{\text{fwd}}_{a,t} \in \{0,1\}$ | selection of forward direction for $a \in \mathcal{A}^{\text{arBiRg}}$ | 1 |
| $x^{\text{bwd}}_{a,t} \in \{0,1\}$ | selection of backward direction for $a \in \mathcal{A}^{\text{arBiRg}}$ | 1 |
| $q^{\text{fwd}}_{a,t} \in \mathbb{R}_{\geq 0}$ | forward flow of $a \in \mathcal{A}^{\text{arBiRg}}$ | kg/s |
| $q^{\text{bwd}}_{a,t} \in \mathbb{R}_{\geq 0}$ | backward flow of $a \in \mathcal{A}^{\text{arBiRg}}$ | kg/s |
| $x^{\text{cfg}}_{c,t} \in \{0,1\}$ | selection of configuration $c \in \mathcal{C}^{\text{ar}}_a$ for $a \in \mathcal{A}^{\text{arCp}}$ | 1 |
| $x^{\text{st}}_{s,t} \in \{0,1\}$ | selection of simple state $s \in \mathcal{S}$ | 1 |
| $x^{\text{fd}}_{f,t} \in \{0,1\}$ | selection of flow direction $f \in \mathcal{F}$ | 1 |
| $\sigma^{p+}_{v,t} \in \mathbb{R}_{\geq 0}$ | positive pressure slack for boundary node $v \in \mathcal{V}^{\text{b}}$ | bar |
| $\sigma^{p-}_{v,t} \in \mathbb{R}_{\geq 0}$ | negative pressure slack for boundary node $v \in \mathcal{V}^{\text{b}}$ | bar |
| $\sigma^{d+}_{v,t} \in \mathbb{R}_{\geq 0}$ | positive inflow slack for boundary node $v \in \mathcal{V}^{\text{b}}$ | kg/s |
| $\sigma^{d-}_{v,t} \in \mathbb{R}_{\geq 0}$ | negative inflow slack for boundary node $v \in \mathcal{V}^{\text{b}}$ | kg/s |
| $\delta^{\text{va}}_{a,t} \in \{0,1\}$ | mode change for original valve $a \in \mathcal{A}^{\text{vaOr}}$ | 1 |
| $\delta^{\text{rg}}_{a,t} \in \{0,1\}$ | mode change for regulator $a \in \mathcal{A}^{\text{rg}}$ | 1 |
| $\delta^{\text{rg-pl}}_{a,t} \in \mathbb{R}_{\geq 0}$ | change of incoming pressure of active $a \in \mathcal{A}^{\text{rg}}$ | bar |
| $\delta^{\text{rg-pr}}_{a,t} \in \mathbb{R}_{\geq 0}$ | change of outgoing pressure of active $a \in \mathcal{A}^{\text{rg}}$ | bar |
| $\delta^{\text{rg-qa}}_{a,t} \in \mathbb{R}_{\geq 0}$ | change of flow over of active $a \in \mathcal{A}^{\text{rg}}$ | kg/s |
| $\delta^{\text{arc}}_{a,t} \in \{0,1\}$ | activity change for artificial arc $a \in \mathcal{A}^{\text{ar}}$ | 1 |
| $\delta^{\text{st}}_{s,t} \in \{0,1\}$ | activation of simple state $s \in \mathcal{S}$ | 1 |
| $\delta^{\text{fn-p}}_{a,t} \in \mathbb{R}_{\geq 0}$ | change of pressure of fence node $v \in \mathcal{V}^{\text{fn}}_i$ for $i \in \mathcal{I}$ | bar |
| $\delta^{\text{fn-q}}_{a,t} \in \mathbb{R}_{\geq 0}$ | change of flow over valve of fence node $v \in \mathcal{V}^{\text{fn}}_i$ for $i \in \mathcal{I}$ | kg/s |

Table A.1: List of all used variables, specifying their domain, meaning and unit. Note that $1\,\text{bar} = 10^5\,\text{Pa}$.