Optim Eng (2018) 19:297–326
https://doi.org/10.1007/s11081-017-9371-4

A decomposition approach for optimal gas network extension with a finite set of demand scenarios

Jonas Schweiger[1] · **Frauke Liers**[2]

Received: 30 September 2016 / Revised: 15 August 2017 / Accepted: 25 November 2017 /
Published online: 17 February 2018

Abstract Today's gas markets demand more flexibility from the network operators which in turn have to invest in their network infrastructure. As these investments are very cost-intensive and long-lasting, network extensions should not only focus on a single bottleneck scenario, but should increase the flexibility to fulfill different demand scenarios. In this work, we formulate a model for the network extension problem for multiple demand scenarios and propose a scenario decomposition in order to solve the resulting challenging optimization tasks. In fact, each subproblem consists of a mixed-integer nonlinear optimization problem. Valid bounds on the objective value are derived even without solving the subproblems to optimality. Furthermore, we develop heuristics that prove capable of improving the initial solutions substantially. The results of computational experiments on realistic network topologies are presented. It turns out that our method is able to solve these challenging instances to optimality within a reasonable amount of time.

Keywords Decomposition · Robust · Gas network extension · MINLP

✉ Jonas Schweiger
schweiger@zib.de

Frauke Liers
frauke.liers@math.uni-erlangen.de

[1] Department Mathematical Optimization, Zuse Institute Berlin, Takustrasse 7, 14195 Berlin, Germany

[2] Department of Mathematics, University Erlangen-Nürnberg, Cauerstrasse 11, 91058 Erlangen, Germany

1 Introduction

Gas transmission networks are complex structures that consist of passive elements and active, controllable elements such as valves and compressors. The behavior of the passive elements is entirely governed by the laws of physics, and the network operator has no means to influence that behavior. Pipes are the most important representative of that group. Other passive elements are for example measurement equipment that causes a pressure drop or artificial resistor modeling e.g., extensive piping within stations. Active elements on the other hand are controlled by the network operator. Several active elements exist: Valves can be open or closed and are a means to decouple different parts of the network. Compressors and control valves can increase and decrease the pressure within technical limitations. For planning purposes, the relationship of flow through a pipe and the resulting pressure difference is appropriately modeled by a nonlinear equation. The description of the active elements on the other side involves discrete decisions, e.g., whether a valve is open or closed. Therefore, the model to describe a gas network is a *Mixed-Integer Nonlinear Program* (*MINLP*) and its feasible set is non-convex in general.

Recent changes in the regulation of the German gas market are creating new challenges for *Transmission System Operators* (*TSO*). Especially the unbundling of gas transport and trading reduces the influence of network operators on transportation requests. Greater flexibility in the network is therefore demanded, and the networks have to be extended accordingly. Traditionally, deterministic planning approaches focus on one bottleneck situation. Accordingly, the solutions are fine-tuned to that scenario. In practice, however, the TSOs are obliged by the regulators and by contracts with gas traders to ensure a feasible network operation in a large range of different demand scenarios (also known as *nominations*). Considering uncertainty in the form of a set of scenarios leads to more flexible network extensions that can meet future demands more efficiently. Furthermore, one can think of having available historical data scenarios for typical and relevant nominations against which protection is sought. In order to be protected against such a finite set of different flow scenarios, we model the problem in a robust optimization framework. The advantage of such a solution determined through this approach is feasible for all such historical 'typical' scenarios. It is furthermore not necessary to know anything about the distribution of the uncertain data.

Determining best possible network extensions at minimum cost is a difficult task as the network can be extended in various ways. In principle, any two points can be connected by a new pipe, and pipes are available in different standardized diameters. Building an additional pipe next to an existing one is called *looping*. Loops are the favorite extensions of network operators as they are considerably cheaper than new pipes as the regulatory process is much simpler and the TSO most often already owns the land the pipe is built on. In addition to pipes, new active elements can also be added anywhere in the network. Hence, generating meaningful extension candidates is a challenging task on its own. In this work we assume that extension candidates are given as part of the input to the problem, and we consider the question of choosing a subset that renders all scenarios feasible at minimal cost.

While typically network extensions increase the transport capacity of a network, they can also cause new infeasibilities. A new pipe allows flow but couples the pressures at the end nodes, possibly rendering previously feasible transport requests infeasible. An additional valve retains all possibilities of the original network. Closing the valve forbids flow over the pipe, which effectively removes the pipe from the system.

Gas network operation problems have been considered in the literature in various contexts and in different settings, mainly for the nominal versions in which uncertainties are ignored. The recent book by Koch et al. (2015) as well as the state-of-the-art survey of Ríos-Mercado and Borraz-Sánchez (2015) cover the main issues in great detail. We briefly summarize some further relevant references in the following. For further pointers to the literature, we refer to the above two documents. As an important result, it has been shown in Collins et al. (1978) and Ríos-Mercado et al. (2002) that the feasible flow of a given demand scenario for a network without pressure bounds is uniquely determined. Furthermore, some simplifications can be applied due to the fact that all pressure variables as well as the flow variables on a spanning tree can be eliminated from the system; see Gotzes et al. (2016). However, only a few contributions exist that take uncertainties into account. The thesis of Midthun (2007) develops mathematical optimization models for different natural gas optimization problems. In Tomasgard et al. (2007), a complex capacity booking problem with stochastically distributed demands is studied. In Fodstad et al. (2016) stochastic programming is used for minimum-cost topology planning for the European gas transport under demand uncertainty. A detailed case study is presented. Furthermore, in Hellemo et al. (2013), the task of designing gas network extensions together with some investment decisions is presented. The corresponding mathematical optimization model is complex and is approached via multi-stage stochastic programming techniques. Finally, scenario decomposition has been applied in Gabriel et al. (2012) to gas market models.

In this work, we present a robust model for gas network extension that protects the TSO against a finite set of scenarios (i.e., transport requests). The novelty of our work consists in the development of a branch-and-bound algorithm based on scenario decomposition that solves the network extension problem for the individual scenarios as subproblems. The algorithm provides lower bounds on the obtainable cost such that the quality of solutions can be accessed. While the algorithm is guaranteed to find the optimum solution, we incorporate heuristic methods that prove capable of finding high-quality solutions and in turn speed up the optimization. A computational study of realistic network topologies shows the effectiveness of our approach. Our method is able to solve challenging instances with up to 256 scenarios, whose scenario-expanded MINLP formulations have hundreds of thousands of variables and constraints, to global optimality in a reasonable amount of time.

This work was performed in the research project FORNE[1] in cooperation with the German gas network operator Open Grid Europe GmbH. Parts of this paper have been published in Schweiger (2016, 2017).

This paper is organized as follows. Section 2 gives an overview of the mathematical model for gas networks and its extension. The decomposition method is presented in

[1] ForNe—Research Cooperation Network Optimization: http://www.zib.de/projects/forne-research-cooperation-network-optimization.

Sect. 3 together with some details about primal and dual bounds and results on the ability to reuse solutions from previous optimization runs over the same scenario. Section 4 presents the results of computational experiments. Section 5 discusses planned future work on the topic.

2 Models and algorithms for gas network optimization

2.1 Modeling gas transportation networks

In this section, we describe the mathematical optimization model for topology planning in a deterministic setting. We restrict our presentation to the level needed to understand the mathematical structure of the problem. The reader is referred to Koch et al. (2015) and Pfetsch et al. (2014) for further details on the assumptions underlying our model and for precise formulas for the coefficients.

The gas network is modeled as a directed graph $G = (V, A)$, where the arcs A are physical network components. Within the network, gas is to be transported from entries to exits. The flow at these points is given by a so-called *nomination* and is modeled by a vector $q^{\text{nom}} \in \mathbb{R}^V$ where positive and negative values of q_u^{nom} mean that flow is leaving and entering the network at node u, respectively.

We assume a steady-state model where dynamic effects are not taken into account and pressure within the arcs is assumed to be constant. One consequence is that the nomination has to be balanced, i.e., $\sum_{u \in V} q_u^{\text{nom}} = 0$, as the gas in the network cannot be used to balance short-term imbalances. We introduce pressure variables p_u to track the pressure at node $u \in V$. Flow through an element is modeled by a variable q_a for arc $a \in A$. Flows in the direction of the arcs are encoded by positive values for q_a while negative values encode flow in the reverse direction. We assume a homogeneous gas composition. Under this assumption, gas blending effects at the nodes can be ignored, and the flow respects the flow conservation constraints

$$\sum_{a \in \delta^+(u)} q_a - \sum_{a \in \delta^-(u)} q_a = q_u^{\text{nom}},$$

where $\delta^-(u)$ and $\delta^+(u)$ are the arcs entering and leaving node u, respectively. Flow conservation constraints at all nodes ensure that the flow is compliant with the given nomination. Additionally, flow and pressure can have technical upper and lower bounds.

The relationship between flow and pressure depends on the network element, i.e., on the type of the arc. Generally, network elements can be partitioned into two groups: Passive and active elements. In the following, we will briefly review the model for pipes as the most prominent representative of a passive element and the models of the different active elements.

Pipes Pipes are used to transport gas over long distances. A difference in the pressures in the end points is needed for gas to flow. Mathematically, the relationship between the pressure at the end nodes u and v of a pipe a and the flow q_a is modeled by the equation

$$\alpha_a |q_a| q_a = p_u^2 - \beta_a p_v^2 \tag{1}$$

The parameters α_a and β_a are determined by the properties of the pipe and are assumed to be constant.

The right-hand side can be linearized by reformulating it using variables for the square of the pressure $\pi_u = p_u^2$. Since the pressure is always positive (above the atmospheric pressure), this reformulation does not introduce ambiguities. Furthermore, we introduce an auxiliary variable z_a and split the equation into a nonlinear and a linear equation:

$$\alpha_a |q_a| q_a = z_a \tag{2}$$

$$z_a = \pi_u - \beta_a \pi_v. \tag{3}$$

The only remaining nonlinearity is then present in Eq. (2); see Fig. 1 for a plot. The algorithmic approach to handle this nonlinearity is described in Sect. 2.3. The model of some network elements, e.g., compressors, needs pressure variables. In this case, the pressure variable is added to the model together with the coupling constraint $\pi_u = p_u^2$; otherwise it is omitted.

Active elements: valve, control valve, and compressor The three most important active elements are valves, control valves and compressors. Valves can be used to disconnect parts of the network. Control valves have the additional feature that they are able to reduce the pressure while compressors can increase the pressure. In contrast to pipes, whose behavior is completely ruled by gas physics, the state of active elements can be controlled by the network operator and might be changed frequently to influence the behavior of the network.

The simplest active element is a valve. Valves have two possible states, open and closed, which is modeled by a binary variable s_a. An open valve ($s_a = 1$) does not cause a change in the pressure and allows flow within some technical bounds. A closed valve ($s_a = 0$) does not allow flow, but completely decouples the pressures at both end points. Mathematically, the conditions for a valve $a = (u, v)$ can be expressed as follows:

$$s_a = 0 \quad \Rightarrow \quad q_a = 0 \tag{4}$$

$$s_a = 1 \quad \Rightarrow \quad p_u = p_v \tag{5}$$

These implications can be implemented using indicator or so-called Big-M constraints; see Bonami et al. (2015) for a recent review.

In addition to open (called "bypass") and closed, control valves and compressors have an additional possible state: the active state. In this state, the actual increase or decrease of pressure takes place. Compressors and control valves use two binary variables s_a^{bp} and s_a^{ac} to decide the state of the element. If $s_a^{\mathrm{bp}} = 1$, then the element is in bypass mode. If $s_a^{\mathrm{ac}} = 1$, the element is in active mode. If both variables are zero, the element is closed.

A control valve $a = (u, v)$ allows the reduction of the pressure in the direction of the flow within certain bounds $\underline{\Delta}_a$ and $\overline{\Delta}_a$ when in active state. The constraints for a

control valve are thus:

$$s_a^{\mathrm{bp}} = 0,\ s_a^{\mathrm{ac}} = 0 \quad \Rightarrow \quad q_a = 0 \tag{6}$$

$$s_a^{\mathrm{bp}} = 1,\ s_a^{\mathrm{ac}} = 0 \quad \Rightarrow \quad p_u = p_v \tag{7}$$

$$s_a^{\mathrm{bp}} = 0,\ s_a^{\mathrm{ac}} = 1 \quad \Rightarrow \quad \begin{cases} q_a \geq 0 \\ 0 < \underline{\Delta}_a \leq p_u - p_v \leq \overline{\Delta}_a \end{cases} \tag{8}$$

$$s_a^{\mathrm{bp}} + s_a^{\mathrm{ac}} \leq 1 \tag{9}$$

Eqs. (6)–(8) describe the three states. Inequality (9) ensures that exactly one of the three states is selected.

Compressors are by far the most complex elements. The pressure increase depends on the flow and is governed by the so-called *characteristic diagram* (see for example Odom and Muster 2009 or Percell and Ryan 1987), which typically is a non-convex set. We use a linear approximation to the characteristic diagram and remain with the statement that the triple (p_u, p_v, q_a) must be in a certain polytope.

Since bypass and closed state of compressors and control valves behave identically to a valve, we also model valves with the two binary variables s_a^{bp} and s_a^{ac}, but fix s_a^{ac} to zero.

The question whether a network allows feasible operation for a given nomination q^{nom} is called *nomination validation*. Nomination validation is a challenging task that network operators routinely face in daily operation as well as in tactical and strategic planning. Formulated in this way, it is a feasibility problem without an objective function. We refer to Koch et al. (2015) and Pfetsch et al. (2014) for a more detailed description of the network elements and their coefficients as well for details on the nomination validation problem.

2.2 Deterministic network extension

In this section, we extend the feasibility problem of checking whether a nomination allows a feasible operation in a network to the selection of a cost-optimal set of network extensions that allows the operation of a previously infeasible nomination. More details on the approach for deterministic network extension can be found in Fügenschuh et al. (2011).

For this question to be well posed, we assume a set of possible extension candidates $\mathcal{E}$ is given. An extension $e \in \mathcal{E}$ can be a new pipe to be constructed (possibly as a loop) or the insertion of an active element at the beginning or end of an existing pipe in the network. In the case of a new pipe, an active element is always added at one of the end-points. This is not only important for our model, but has a practical background. A new pipe connects previously unconnected or only loosely connected parts of the network and might affect the flow and pressure distribution in the entire network. In the extreme case, the construction of a new pipe can render previously feasible nominations infeasible. Closing the active element at the end-points neutralizes the effect of the pipe and yields the original network.

We assign an integer variable x_e to each extension candidate $e \in \mathcal{E}$. Control valves and compressors have all the capabilities of valves with an additional active state. Therefore, when the active state is not used, a much cheaper valve should be constructed instead of a control valve or a compressor. There are three possible outcomes of the investment decision for extension e which are translated into the variable x_e:

$x_e = 0$: Do not construct e.
$x_e = 1$: Construct extension e with a valve instead of the proposed active element.
$x_e = 2$: Construct extension e with the proposed active element.

If the active element is a valve, then x_e can take only the values 0 and 1. The three options form a hierarchy, where every option is at least as powerful as the ones with a smaller value, but usually at a higher cost.

The general approach consists in extending the network by the candidates and penalizing the use of the extensions by cost on the binary variables of the corresponding active elements. The operation decision of the active element is then translated into the decision about whether and how the extension is constructed. Consider for example a new pipe or loop and its corresponding active element. As closing the active element means that no flow goes through the pipe and the effect of the pipe is neutralized, no cost is associated with the closed state. Using the bypass state means that flow goes along the pipe, but the active element is not used in its active state. Thus, it suffices to construct a valve to activate and deactivate the new pipe. The cost for the bypass state is thus the cost for the new pipe plus the cost of a valve. Finally, if the active state is used, then the pipe and the proposed element have to be constructed, and therefore the cost associated with the active state is the cost of the pipe plus the cost of the proposed active element. The translation from the operation variables s_a^{bp} and s_a^{ac} to the investment variable x_e is in this case

$$x_e = s_a^{\mathrm{bp}} + 2s_a^{\mathrm{ac}}.$$

In the other case of an active element being added to the end of an existing pipe, the meanings of bypass and closed state are reversed. When in bypass, the proposed element has no effect and no costs occur; when in closed state, a valve has to be constructed:

$$x_e = 1 - s_a^{\mathrm{bp}} + s_a^{\mathrm{ac}}.$$

The cost for the extension e is modeled by an increasing function $c_e(x_e)$ which is evaluated only at integer points. Using the variables s_a^{bp} and s_a^{ac} the objective can be formulated easily.

We denote by $\mathcal{F}$ the set of all vectors $x = (x_e)_{e\in\mathcal{E}}$ such that the extended network allows a feasible operation. In our situation, a closed form description of $\mathcal{F}$ is not at hand, and optimization over this set corresponds to the solution of a non-convex MINLP due to the complex model for physics and discrete decisions. In an abstract form, the

deterministic extension planning problem can now be stated as

$$\begin{aligned} &\min\ c(x) && \text{(SingleScen)}\\ &\text{s.t. } x \in \mathcal{F} \end{aligned}$$

This formulation is complete, but hides the difficulties in describing and optimizing over the set $\mathcal{F}$. An MINLP model of this problem can be solved by different techniques. The approach used in this paper uses an outer approximation of the nonlinear function that is refined in the course of the algorithm and is described in the following section. An alternative is the approximation or relaxation of the nonlinear function by piecewise linear functions, which yields, an MILP to be solved (see for example Koch et al. 2015).

2.3 Algorithmic approach for the deterministic network extension problem

The model has two complicating features: Discrete decisions and nonlinear equations. Discrete decisions are essential to model the settings of active devices, which are naturally discrete. Nonlinearities arise from the model of the gas physics and add to the nonconvexity of the feasible set. The result is a non-convex MINLP, which is a very broad class of mathematical programs and belongs to the most challenging optimization tasks that are currently studied. In this section we briefly describe the algorithmic framework which most MINLP solvers use to tackle such a problem. We sketch only the basic principles and refer to Berthold et al. (2012) and Vigerske (2012) for the details on the algorithm and the implementation in the solver SCIP. SCIP has been extended to solve gas network optimization problems effectively, and our method uses the corresponding subroutines in the subproblems.

The algorithmic paradigm to handle both features is branch-and-bound. To this end, we first construct a linear relaxation for the nonlinear equation Eq. (2). This is done by replacing the non-convex set of feasible points by a larger set that contains all feasible points, but is described solely by linear inequalities. The convex hull of the feasible points yields the best relaxation, but it is not necessarily a polyhedron as its number of extreme points and rays is not necessarily finite. Figure 1a shows the feasible set for the equation $|q_a|q_a = z_a$ as a solid line and a linear relaxation as a shaded area. Usually the relaxation is strengthened by additional valid inequalities (also called *cutting planes*) during the algorithm.

The relaxation is then used within an LP-based branch-and-bound algorithm. When a relaxation is integer feasible, however, it is not necessarily feasible for the nonlinear problem as the nonlinear equation is relaxed. If the point is in the interior of the convex hull of the feasible points it cannot be separated by a linear cutting plane without also cutting off feasible points. In this case, the two subproblems are created by splitting the domain of z_a into two parts. As the domain of z_a is smaller in both subproblems, the relaxation can be improved and the current solution of the relaxation can be separated. Branching on a continuous variable is generally referred to as *spatial branching*. The effect of spatial branching is then illustrated in Fig. 1b, where the branching point is $z_a = 0$ and the outer approximations in both branches are drawn in the same picture.

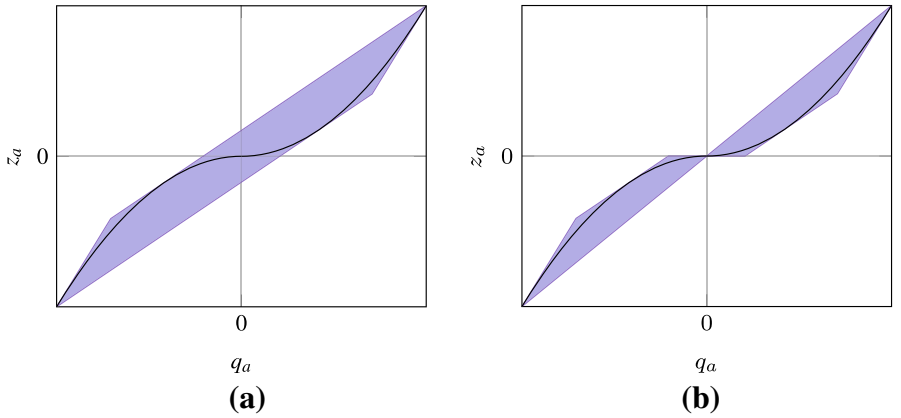

Fig. 1 Nonlinear equation $|q_a|q_a = z_a$ and a linear approximation on the original interval (**a**) and after one branching at $q_a = 0$ (**b**)

3 Gas network topology planning for multiple scenarios

The extension of the gas infrastructure involves long-lasting and cost-intensive investments. The operators therefore seek extensions that solve several potential bottleneck situations at once, are flexible in the future operation and play well with possible subsequent network extensions. Clearly, deterministic optimization does not respect any of these objectives. Instead it selects a set of extensions that is tailored towards the particular nomination that might not be relevant for the future. It is therefore of high importance to consider several nominations simultaneously in order to avoid over-tuning and prepare the network for a large range of different flow distribution patterns.

When facing protection against uncertainties in an optimization problem, several options exist. Robust optimization is a framework to protect against uncertainty in the input data of an optimization problem. Instead of assuming that the data that describes the objective and the constraints is known, the input data is assumed to realize within an uncertainty set. The decisions that are to be determined then must be robust, i.e., they need to be feasible no matter how the data manifests with the uncertainty set. Furthermore, a robust solution is sought that yields the best guaranteed cost. In recent decades, robust mathematical optimization has received increased attention. Here, we can briefly mention some relevant references. Soyster (1973) can be seen as one of the first references to linear programming under uncertainty. In this work, column-wise uncertainty is assumed. Later, Ben-Tal and Nemirovski (1999) considered constrained-wise uncertain linear optimization problems. Among other things, the book by Kouvelis and Yu (1997) introduced the concept of minimax regret as a special modeling approach for robust protection against uncertainties. For robust combinatorial optimization, Bertsimas and Sim (2003) presented a flexible robustness approach for a computationally tractable robust counterpart that is protected against at most Γ realized uncertainties, where Γ is given as input. The book by Ben-Tal et al. (2009) covers many of these mathematical theories and approaches. Gorissen et al.

(2015) gives a recent overview of effective reformulations and algorithms, together with how they can be applied in practice.

In order to solve robust optimization tasks effectively, we can derive equivalent reformulations of the robust problems such that the corresponding robust counterparts become computationally and algorithmically tractable. For linear mixed-integer problems, such tractable robust counterparts can be derived for several classes of uncertainty sets, such as conic or polyhedral sets. This means that robust mixed-integer optimization tasks can be solved effectively in practice. As we are facing a complex MINLP here, much less is known about tractable robust counterparts. We therefore consider a discrete uncertainty set that consists of a finite number of nominations. This reflects the situation in which different scenarios are collected from historical data or from future forecasts, which was also the case for our industry partner. We will refer to the elements in the uncertainty set as *scenarios*.

The decision variables in our application naturally decompose into two stages: In the first stage, we decide which extensions are constructed. In the second stage, we determine the operational decisions, i.e., the control of the active devices and the resulting physical values such as pressure and flow, for all scenarios. While the first-stage decisions are taken once for all scenarios, the second-stage decisions are taken independently in each scenario and have to take the first-stage decisions into account. Accommodating robust multi-stage optimization problems is an active research field, and several different approaches have been proposed, among them *Adjustable Robust Optimization* (Ben-Tal et al. 2004) and *Recoverable Robust Optimization* (Liebchen et al. 2009).

In the context of our multi-scenario extension planning problem, the scenarios represent nominations and we seek a set of extensions at minimal cost such that the resulting network allows the feasible operation of *all* scenarios. We stress that in the different scenarios not all extensions that have been constructed have to be used. However, the hierarchical model of network extensions from Sect. 2.2 ensures that the extensions can always be used at a smaller level.

The problem can then be formulated as a two-stage robust program. The first-stage variables y indicate the extent to which extensions are constructed. This decision is independent of the scenario. In a particular scenario ω, the second-stage variables $x^\omega \in \mathcal{F}^\omega$ describe the extent to which the extensions are used. The multi-scenario problem can then be stated as:

$$\min c(y) \tag{MultiScen}$$

$$\text{s.t. } x^\omega \in \mathcal{F}^\omega \quad \text{for all} \quad \omega \in \Omega \tag{10}$$

$$x^\omega \leq y \quad \text{for all} \quad \omega \in \Omega \tag{11}$$

$$\underline{y} \leq y \leq \overline{y} \tag{12}$$

Constraint (11) ensures that an extension used in at least one scenario is constructed. Together with the increasing objective function $c(y)$, (11) and the lower bound on y from (12) form the linearization of

$$y = \max\left(\max_{\omega\in\Omega} x^{\omega}, \underline{y}\right). \tag{13}$$

Clearly, all variables have to be nonnegative, but we state explicit bounds as they will be handy in the description of the algorithm in Sect. 3.1. Integrality constraints can be omitted as they are encoded in $\mathcal{F}^{\omega}$ for x^{ω} and enforced by (13). Note that this model is valid only because the extensions form a hierarchy where more expensive extensions only add functionality. The model also accounts for the fact that extensions might be used to a smaller extent than possible.

This model can also be viewed as a two-stage stochastic programming problem. While in robust optimization all scenarios from the uncertainty set have to be feasible at a minimal cost, in two-stage stochastic optimization a probability distribution on the scenarios in the second stage is given and the expected cost is to be optimized. See, for example, Birge and Louveaux (2011) for a detailed overview of stochastic programming. The linking constraints Eq. (11) are then called non-anticipativity constraints. Without costs on the second-stage variables and without scenarios with probability zero, these two concepts coincide since in stochastic programming infeasibility of a scenario would be punished by an infinitely high cost.

In principle, a scenario-expanded problem can be formulated by adding all constraints that describe the relationship $x^{\omega} \in \mathcal{F}^{\omega}$ explicitly to the model. Then the operational decisions get another index for the scenario as they act on the second level of the problem. This formulation could be solved with the algorithm from Sect. 2.3. However, since the problem is challenging for even one scenario, there is no hope that the resulting MINLP can be solved for a non-trivial number of scenarios.

Without the constraints (11) the model would decompose as each scenario problem could be solved individually. A decomposition approach therefore seems most promising for this model. Several decomposition approaches have been proposed. Classical Generalized Benders Decomposition (Geoffrion 1972) requires convexity to provide optimal solutions (Sahinidis and Grossmann 1991). A non-convex variant by Li et al. (2011, 2014) was used to solve a stochastic pooling problem for gas network planning under uncertainty (Li et al. 2016). In general, due to the lack of knowledge of the structure in the set $\mathcal{F}^{\omega}$, feasibility cuts that carry more information than just forbidding one particular assignment of y are difficult to obtain.

We propose a decomposition where the constraints (11) are ensured by branching on the y variables. An approach similar in spirit has been proposed by Carøe and Schultz (1999). They use Lagrangian decomposition by dualizing the constraints (11) to get dual bounds and use a branch-and-bound algorithm to ensure feasibility. Lagrangian decomposition is known to provide good bounds in a large range of applications and is a common technique in stochastic programming, but it is impractical here as the subproblems need to be solved to optimality several times in order to compute good bounds. In the present application, even the single-scenario subproblems can hardly be solved to optimality, as we will see in Sect. 4. We therefore present a branch-and-

bound algorithm in Sect. 3.1 which is tailored towards minimizing the number of times the subproblems are solved by avoiding changes in the objective as they would appear in Lagrangian decomposition. The dual bounds for the bounding step are presented in Sect. 3.2. We discuss heuristic methods to quickly find good feasible solutions in Sect. 3.3. In Sect. 3.4 we develop conditions on when previously found solutions for a subproblem can be reused and the corresponding subproblem does not need to be solved again.

3.1 Scenario decomposition: a branch-and-bound approach

In the following we outline the algorithmic approach that consists of scenario decomposition in combination with branching on y variables.

First, we solve the scenario subproblems (2.2) independently and possibly in parallel for all scenarios $\omega \in \Omega$. Due to the complexity of the problem, which is a non-convex MINLP even for one scenario, we aim to leverage our capabilities of solving these problems and choose a setting where second-stage decisions are decided by a black-box solver we do not interfere with. In this way we also directly benefit from future improvements of solvers for non-convex MINLP. The non-convex problems are encapsulated, but we can still use the known structure of the solution space in the design of the algorithm.

If one scenario subproblem is infeasible, the multi-scenario problem is infeasible. If all subproblems are feasible, we denote the best solution found for scenario ω by x^ω. A feasible solution to the multi-scenario problem can be computed by setting

$$y_e^\star = \max_{\omega \in \Omega} x_e^\omega,$$

i.e., by constructing all extensions that are used in at least one scenario.

Next, we identify extensions that differ in the extent to which the extension should be constructed, i.e., extensions $e \in \mathcal{E}$ for which

$$\min_{\omega \in \Omega} x_e^\omega \neq \max_{\omega \in \Omega} x_e^\omega. \tag{14}$$

Branching on the y variables is used to synchronize the investment decisions in the different scenarios. To this end, an extension e for which (14) holds and a value τ between $\min_{\omega \in \Omega} x_e^\omega + 1$ and $\max_{\omega \in \Omega} x_e^\omega$ is chosen and two subproblems, i.e., nodes in the branch-and-bound tree, are created: one with the condition $y_e \leq \tau - 1$ and one with the condition $y_e \geq \tau$. In the two nodes that emerge the variables y now have non-default bounds, but otherwise the structure of (MultiScen) is unchanged. In consequence, a branch-and-bound tree is built, where each node is identified by the bound vectors $\underline{y}$ and $\overline{y}$.

In the nodes, the subproblems have to be modified in order to reflect the bounds on the y variables. Extensions e whose lower bound $\underline{y}_e$ is greater than zero are constructed to this extent and the cost is charged as a fixed cost in the subproblems. In addition, the extension might be used with a value larger than $\underline{y}_e$, in which case additional cost is charged. The cost is thus computed as $\max\left(c_e(\underline{y}_e), c_e(x_e^\omega)\right)$, an expression which

is easily linearized. Upper bounds $\overline{y}_e$ are applied to x_e^ω to control the use of extension e in scenario ω.

The modified single-scenario problem for scenario ω and bounds $\underline{y}$ and $\overline{y}$ then reads as:

$$\begin{aligned} \min \ & \sum_{e \in \mathcal{E}} \max\left(c_e(\underline{y}_e), c_e(x_e^\omega)\right) && \text{(SingleScen}_\omega\text{)} \\ \text{s.t. } & x^\omega \in \mathcal{F}^\omega \\ & x^\omega \leq \overline{y} \end{aligned}$$

Hence, the subproblem is again a single-scenario extension planning problem with an adapted objective function and upper bounds on some variables, which prevents the usage of certain extensions.

In most branch-and-bound algorithms a relaxation is used to guide the algorithm and produce lower bounds. Tighter bounds on the problem remove solutions from the relaxation and thus the objective function of the relaxation can only become worse by tightening the bounds. In our case, the value of the multi-scenario problem (MultiScen) also deteriorates as the bounds get tighter since the search space is restricted. Therefore, nodes which are deeper in the tree, i.e., have tighter bounds, have a greater than or equal optimal objective value than higher ones. A lower bound on the solution value of the multi-scenario problem associated with a node of the branch-and-bound tree, i.e., to a pair of bounds $(\underline{y}, \overline{y})$, makes it possible to prune the node if this lower bound is worse, i.e., larger, than the value of the best-known solution. In this case, no improving solution can be found in the subtree associated with the node, and the node can be pruned from the branch-and-bound tree. A lower bound on the solution value of a minimization problem is also referred to as a *dual bound* while feasible solutions are also known as *primal solutions*, and the value of the best-known feasible solution as a *primal bound*. If the primal and dual bounds coincide, the problem is solved to optimality since the dual bound ensures that no solution with a better objective value can exist. The following two sections study dual bounds and primal solutions for our problem.

3.2 Dual bounds

Lower bounds for the single-scenario problems can be instantly translated into lower bounds for the multi-scenario problem. Intuitively, the cost to ensure simultaneous feasibility of all the scenarios has to be greater than that for any single scenario. The following short lemma formalizes the argument.

Lemma 1 *Let the objective function be non-negative. Then any dual bound for problem* (15a) *for any scenario is also a dual bound for problem* (MultiScen)*.*

Proof Let $\overline{c}$ be a dual bound to (15a) for scenario ω, i.e., $\overline{c} \leq c(x^\omega)$ for $x^\omega \in \mathcal{F}^\omega$. With constraint (11) and the fact that the objective function $c(.)$ is increasing in every direction, we have $\overline{c} \leq c(x^\omega) \leq c(y)$ for any feasible y. Therefore, $\overline{c}$ is also a lower bound for problem (MultiScen). □

It is clear that the constant value $c(\underline{y})$ is a lower bound on the objective value for (15a). As tighter bounds alter both the objective function and the solution space of the subproblems, we need to ensure that the solution value of the subproblem might only increase with tighter bounds $(\underline{y}, \overline{y})$. The following lemma however states that this is the case.

Lemma 2 *Let c^* be the optimal value of (15a) for some scenario ω and for the bounds $(\underline{y}, \overline{y})$. Consider a second pair of more restrictive bounds $(\underline{\tilde{y}}, \overline{\tilde{y}})$ with $\underline{\tilde{y}} \geq \underline{y}$ and $\overline{\tilde{y}} \leq \overline{y}$ and its optimal value $\tilde{c}^*$ for (15a). Then $c^* \leq \tilde{c}^*$.*

Proof We use induction over $n = \|\underline{\tilde{y}} - \underline{y}\| + \|\overline{\tilde{y}} - \overline{y}\|$ where $\|.\|$ is the ℓ_1-norm. For $n = 0$, the problems and thus their optimal objective values coincide and our claim holds. For the induction step $n = 1$, we distinguish between a tightened upper and lower bound. If a lower bound is tightened, i.e., $\underline{\tilde{y}}_e > \underline{y}_e$ for some e, then the search space remains the same, but the objective function increases for $\underline{y}_e = x_e^\omega$. In the case where an upper bound is tightened, i.e., $\overline{\tilde{y}}_e < \overline{y}_e$, the search space is restricted and the objective function remains unchanged. In both cases the objective function value deteriorates. □

This result will be used in later sections as it ensures the consistency of the dual bound of the subproblem.

3.3 Primal solutions

We propose three ways to generate or to improve feasible solutions:
From the solutions of the subproblems First, by construction the union of all extensions used in the different scenarios constitutes a primal solution for the multi-scenario problem. Therefore, we construct a solution to (MultiScen) in every node by setting $y = \max_{\omega \in \Omega} x_e^\omega$ where x_e^ω is taken as the best solution for scenario ω.
1-opt heuristic Second, we observed that checking if a small subset of extensions is feasible is typically very fast. This observation is used by a 1-opt procedure that takes a solution to (MultiScen), decreases by 1 a variable that has been chosen to take value $y_e > 0$, and checks all scenarios for feasibility. Of course, a priori it is not clear which $y_e > 0$ to choose. Several options have been explored. The most promising is to sort the y_e according to the possible saving $c_e(y_e) - c_e(y_e - 1)$ realizable by decreasing its value by one and to consider extensions with small savings first. The rationale behind this is that often these "small" extensions are used by scenarios which are rather close to feasibility, and some more expensive extensions used in the more challenging scenarios often also ensure the feasibility of these almost-feasible scenarios. Therefore, it is likely that these extensions can be removed. Extensions with higher savings are likely to render some scenario infeasible as their effect cannot be compensated for by the other extensions in the solution.

In order to protect against outliers, a strict time limit is used when checking the scenarios for feasibility. Note that during the 1-opt heuristic all extensions are fixed and the problem is a feasibility problem (nomination validation). Therefore, no time is spent proving optimality.

Best-known heuristic Third, we solve an auxiliary problem to compute the best-known multi-scenario solution taking into account all solutions to the subproblems that the solver found during its solution process.

Optimal single-scenario solutions are very specialized in fixing the bottleneck of the particular scenario. When facing uncertainty in the data, these solutions are not likely to be feasible in the perturbed problems. The optimal solution to the problem with uncertain data is thus often suboptimal for each individual scenario but is able to balance the needs for the different scenarios (e.g., Wallace 2010). In this light, it seems reasonable to consider all known solutions to scenarios in order to construct a multi-scenario solution.

During the solution process of a branch-and-bound solver, all feasible solutions the solver encounters are collected and stored in a *solution pool*. These include solutions that are the best known at the time of finding them but also non-improving solutions that might be used in improvement heuristics by the solver. During and in particular after the optimization, the user can query all solutions in the pool. We collect the solutions for some scenario ω in the set $\mathcal{S}^\omega \subseteq \mathcal{F}^\omega$. Using these solutions provides two benefits. First, they might be suitable solutions to one of the single-scenario problems that are solved in the remaining solution process. All solutions from $\mathcal{F}^\omega$ are therefore added to the solver as starting solutions whenever a single scenario needs to be solved. Second, we construct the "best known" solution for the multi-scenario solution by solving an auxiliary MILP. To this end, we use indicator variables z^x for each $x \in \mathcal{S}^\omega$ and each scenario ω. To ensure the scenario feasibility constraint (10) the MILP selects one solution for each scenario. Furthermore, we need to ensure that all extensions that are used in the selected solutions are constructed. This leads to the following MILP:

$$\min\ c(y) \tag{15a}$$

$$\text{s.t.} \sum_{x \in \mathcal{S}^\omega} z^x = 1 \qquad \text{for all } \omega \in \Omega \tag{15b}$$

$$x \cdot z^x \leq y \qquad \text{for all } x \in \mathcal{S}^\omega,\ \omega \in \Omega \tag{15c}$$

$$z^x \in \{0, 1\} \qquad \text{for all } x \in \mathcal{S}^\omega,\ \omega \in \Omega \tag{15d}$$

$$\underline{y} \leq y \leq \overline{y} \tag{15e}$$

The program has the indicator variables y to decide about the construction of the extensions and the same objective function as before. Constraint (15b) says that a feasible solution has to be selected for every scenario ω. The term $x \cdot z^x$ in (15c) is a vector-scalar multiplication whose result is a (column-)vector, as is y. Constraint (15c) says that the extensions used in a solution x have to be constructed if the solution is selected, i.e., $z^x = 1$. The solution to this program gives the best solution to the multi-scenario problem taking into account all solutions that the solver has found so far for the single scenarios.

Note that the program (15) is also a complete description of the multi-scenario problem if $\mathcal{S}^\omega = \mathcal{F}^\omega$, i.e., if $\mathcal{S}^\omega$ is the set of all feasible solutions. Singh et al. (2009) use this formulation and perform a Dantzig–Wolfe decomposition to solve the continuous relaxation of (15). The pricing problem is then again a single-scenario

problem with the dual variables in the objective function. In their application these solutions are mostly integer feasible. In principle, heuristics or a branch-and-price scheme are needed to generate integer feasible solutions with their approach.

3.4 Reusing solutions

During the proposed branch-and-bound procedure, the bounds $\underline{y}$ and $\overline{y}$ are tightened, and adjusted single-scenario problems are repeatedly solved. Two instances of the single-scenario problem for a scenario differ only in the objective function and the upper bound on the extension variables, as discussed in Sect. 3.1. In some important cases, not all scenario subproblems need to be solved again since we already know the optimal solution. As an example, take the extreme case where a scenario is found to be feasible without extensions. Clearly, the procedure should never touch this scenario again.

In order to decide whether a solution $x \in \mathcal{F}^\omega$ from a previous node can be reused, we need to take into account the bounds $(\underline{y}^x, \overline{y}^x)$ under which the solution was computed and the current bounds $(\underline{y}, \overline{y})$.

We start with the simple observation that if all the extensions in a solution are already constructed, then the solution is optimal for the restricted problem:

Lemma 3 *If $x \in \mathcal{F}^\omega$ and $x \leq \underline{y}$, then x is an optimal solution for (15a) for bounds $(\underline{y}, \overline{y})$.*

Proof Clearly x is feasible for (15a) for the bounds $(\underline{y}, \overline{y})$. As the objective function is increasing, its cost equals $c(\underline{y})$ which is a lower bound for the subproblem. □

Observe that in this case it is irrelevant for which bounds x was computed and that it is not required that it was an optimal solution when it was computed.

The previous lemma examined the situation where all extensions that are used are already constructed. The next lemma treats the opposite situation, where we are using more than have been constructed and no constructed extension is unused. In this situation the solution has to be optimal for some bounds, and the new bounds need to be stronger than the previous ones.

Lemma 4 *Let $x \in \mathcal{F}^\omega$ be an optimal solution to (15a) given the bounds $(\underline{y}^x, \overline{y}^x)$. Let $(\underline{y}, \overline{y})$ be tighter than $(\underline{y}^x, \overline{y}^x)$, i.e., $\underline{y} \geq \underline{y}^x$ and $\overline{y} \leq \overline{y}^x$.*

If $\underline{y} \leq x$ and $x \leq \overline{y}$, then x is an optimal solution to (15a) for bounds $(\underline{y}, \overline{y})$.

Proof The crucial point is the optimality of x given the bounds $(\underline{y}^x, \overline{y}^x)$. Since $x \leq \overline{y}$, the solution is still feasible for $(\underline{y}, \overline{y})$. Since $\underline{y} \leq x$ the solution value of x remains the same for $(\underline{y}, \overline{y})$ as it was for $(\underline{y}^x, \overline{y}^x)$. The value $c(x)$ was a dual bound to (15a) with respect to the bounds $(\underline{y}^x, \overline{y}^x)$. Due to Lemma 2, the dual bound can only have increased by using tighter bounds. In total, x is a feasible solution whose objective value matches the dual bound and is therefore optimal. □

The previous lemmas dealt with extreme cases where the solution dominates the lower bound or vice versa. The situation becomes tricky if a solution x neither dominates

nor is dominated by the current lower bound vector $\underline{y}$. In this case the solution does not fully use extensions that are already constructed but uses extensions that are still undecided. We have to make sure that these unused, but constructed extensions cannot help to find a cheaper overall solution. Define for some solution $x \in \mathcal{F}^\omega$ the set I of extensions where the lower bound has been increased compared to when the solution was found

$$I = \{e \in \mathcal{E} \underline{y}_e > \underline{y}_e^x\}$$

and the set J of those extensions where the solution x uses more than what is already constructed

$$J = \{e \in \mathcal{E} \underline{y}_e \leq x_e\}.$$

The following lemma generalizes Lemma 4.

Lemma 5 *Let $x \in \mathcal{F}^\omega$ be an optimal solution given the bounds $(\underline{y}^x, \overline{y}^x)$ and let $(\underline{y}, \overline{y})$ be tighter than $(\underline{y}^x, \overline{y}^x)$, i.e., $\underline{y} \geq \underline{y}^x$ and $\overline{y} \leq \overline{y}^x$.*

If $I \subset J$ and $x \leq \overline{y}$, then x is an optimal solution to (15a) for the bounds $(\underline{y}, \overline{y})$.

Proof In contrast to Lemma 4, there might be extensions $e \in \mathcal{E}$ that are decided to be constructed to a larger extent than they are used in the solution ($\underline{y}_e > x_e$). The extensions for which the lower bound increased, however, the new lower bound is still below the value of the solution ($I \subset J$). Hence, the objective function value for x is identical for both sets of bounds. Since the dual bounds can only increase by tightening the bounds and since the solution is optimal for the bounds $(\underline{y}^x, \overline{y}^x)$, the solution is still optimal given the bounds $(\underline{y}, \overline{y})$. □

To illustrate the usefulness of the previous lemmas, we consider the situation after branching for the first time. At the root, we assume all scenarios are solved to optimality and a branching point τ is chosen for some extension e. In the first branch, the constraint $y_e \leq \tau - 1$ is added. Clearly, all scenarios ω with $x_e^\omega \geq \tau$ in the optimal solution x need to be solved in this branch because their optimal solution has been cut off and the dual bound for these scenarios might increase due to the additional constraint. For those scenarios ω with $x_e^\omega \leq \tau$, Lemma 4 holds and we know that optimal solutions computed in the root remain optimal in this branch. In the second branch, the constraint $y_e \geq \tau$ is added and the lower bound is tightened. All scenarios with $x_e^\omega < \tau$ have to be solved again, unless they fulfill the conditions of Lemma 3 which in this case means that $x = \underline{y}^x$. Optimal solutions of those scenarios with $x_e^\omega \geq \tau$ again fulfill the conditions of Lemma 4 and those scenarios do not have to be solved again.

3.5 Finite termination and global optimality

In principle, the branch-and-bound algorithm implicitly enumerates all possible assignments to integer variables y (however, in practice many methods are known that allow early fathoming of certain subtrees such that usually only a tiny fraction

of the assignments actually has to be enumerated). Each branching step reduces the search space. Since branching is done on bounded integer variables, each path starting at the root node of the branch-and-bound tree ends at a node where all the variables are fixed, i.e., a leaf of the branch-and-bound tree. When branching on bounded integer variables, the branch-and-bound tree is thus finite. The original problem is infeasible if all leaf nodes are infeasible. Otherwise, the algorithm stops with an optimal solution. Branching on continuous variables is also possible in principle, but a tolerance is needed to ensure that the branch-and-bound tree is finite.

Many algorithms have been developed in order to make branch-and-bound effective in practice. In particular, a good bounding procedure leading to tight bounds usually prevents us from exploring many nodes of the tree. Good bounds are used for early fathoming of subtrees, while still ensuring a global solution is found by the algorithm. While solving the single-scenario subproblems to optimality provides the best dual bound and ensures a small tree, it is not necessary for the correctness of the approach. For the latter, it is sufficient to consider the leaf nodes of the tree. On leaf nodes, all y variables are fixed and scenarios only need to decide whether the constructed network extensions allow a feasible solution. It is thus strictly necessary to solve the single-scenario problems on the leaf nodes. In the inner nodes of the tree, proven optimality of the single-scenario problems is not required. If all the single-scenario problems can be solved at the leaf nodes, this approach is thus guaranteed to prove infeasibility or provide an optimal solution.

4 Computational experiments

To show that our approach can solve practical problems we conducted extensive computational experiments on realistic gas network topologies. Our subproblem is a non-convex MINLP and topology optimization for even one scenario is a challenge; even deciding the feasibility of a nomination is a difficult task; see for example Koch et al. (2015) where the nomination validation problem is extensively studied and it is shown that problems which are on the border between being feasible and infeasible are particularly challenging. In this situation, the linear relaxation is often feasible and it needs much spatial branching to prove infeasibility. The situation where a set of extensions almost suffices to ensure the feasibility of a scenario and a large effort of spatial branching has to be made to prove that this is not the case is expected to occur frequently during our algorithm.

Nevertheless, we are able to provide optimal solutions for instances with up to 256 scenarios whose deterministic equivalent problems have almost 200,000 variables and over 225,000 constraints of which 80,000 are nonlinear. Furthermore, we provide feasible solutions with a proven optimality gap of 16% for an instance whose deterministic equivalent has more than 360,000 variables and 220,000 constraints of which more than 68 000 are nonlinear.

4.1 Computational setup

The approach is implemented using the MINLP solver SCIP (http://scip.zib.de, Achterberg 2009; Vigerske and Gleixner 2016) in the framework Lamatto++ (Geißler et al.

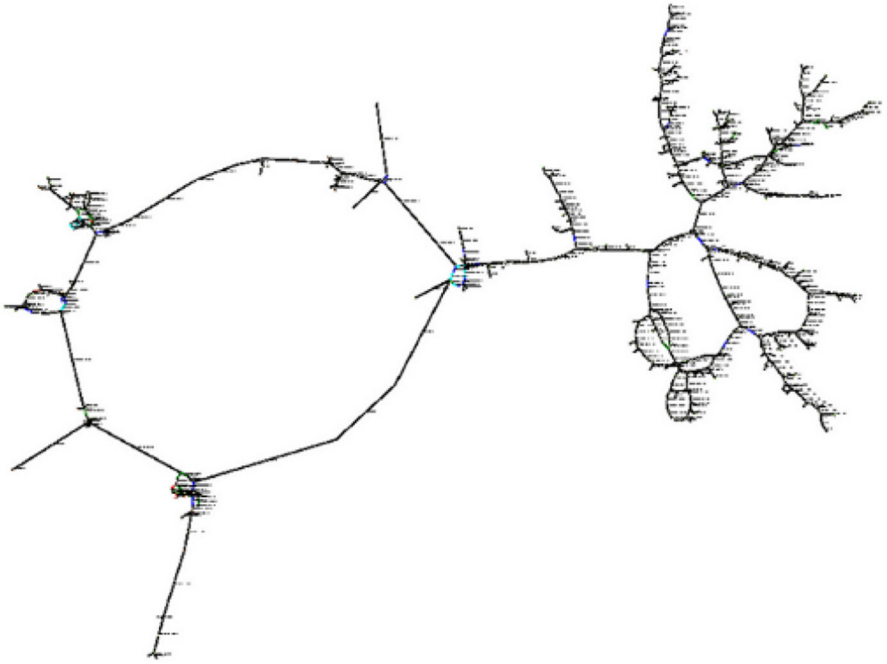

Fig. 2 Visualization of the `gaslib-582` network

2014), which is used for data handling. SCIP provides the core of the branch-and-bound algorithm and is a plugin-based system. Several additional plugins implement the algorithms and models presented in this paper. A constraint handler plugin ensures the abstract constraint (10). A SCIP relaxator plugin triggers the solution of the single-scenario problems, returns dual bounds to SCIP and identifies branching candidates. Methods to solve the single-scenario problems were developed in the project FORNE, see Fügenschuh et al. (2011) and Koch et al. (2015) for detailed descriptions of the models and algorithms. The heuristics from Sect. 3.3 are implemented as heuristic plugins. SCIP is also used to solve the MILP (15). SCIP is used in a development version (shortly before the 3.1 release) and calls Cplex version 12.5.1 to solve LP-relaxations. SCIP is a sequential framework, such that all components are called sequentially, including the various heuristics.

We use `Best First Search` as node selection strategy, where essentially the node with the lowest bound is processed next. This strategy aims at improving the dual bound as fast as possible and yields small search trees (Achterberg 2009), which is exactly what we need as the dual bound of our nodes are very expensive to compute. As branching rule we use the `Most Fractional` rule which branches an a variable y_e whose fractional part $\lceil y_e \rceil - \lfloor y_e \rfloor$ is maximal.

Tests were run on a cluster of servers equipped with Intel Xeon E5-2670 v2 CPUs running at 2.50GHz and 64GB RAM.

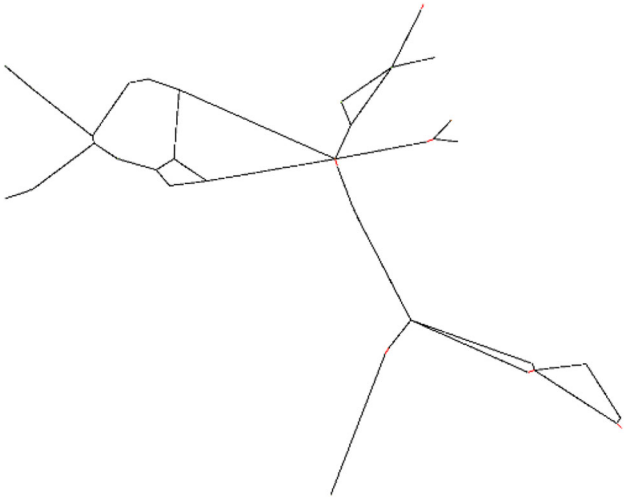

Fig. 3 Visualization of the `gaslib-40` network

4.2 Test sets and instances

To test our approach, we used the networks of the publicly available `gaslib` (Gaslib 2013; Pfetsch et al. 2014). The `gaslib` contains three networks of different sizes. The biggest one, `gaslib-582`, is a distorted version of real data from the German gas network operator Open Grid Europe GmbH and comes with a large number of nominations. The network has 582 nodes and 609 arcs from which 269 are short pipes for which pressure drop is neglected. The smaller ones `gaslib-40` and `gaslib-135` are abstractions of the first and contain 135 and 40 nodes and 160 and 45 arcs, respectively. They are given mainly for testing purposes and come with one reference scenario (Figs. 2, 3).

The generation of meaningful scenarios and extension candidates is a difficult task. The scenarios should be infeasible in the original network and have different bottlenecks that can be fixed by a large number of different extensions from the candidate set in order to yield interesting instances for a robust planning approach. It is important to select nominations from which we know that they can be fixed by extensions in the candidate set. Nominations for which the single-scenario problem is infeasible, i.e., no feasible set of extensions exists, directly render the multi-scenario problem also infeasible. Nominations where no feasible solution for the single-scenario problem can be found will block our approach. Both situations are not in the interest to study the behavior of our approach.

For `gaslib-582` we used the associated nominations and scaled them to simulate growing demand which renders increasing infeasibilities with the original network. For `gaslib-40` we were able to generate scenarios and extensions that yield promising instances. For `gaslib-135`, however, the same algorithm as for `gaslib-40` did not produce instances where SCIP could find feasible solutions for the single scenarios within the time limit for subproblems. We therefore skip this network and use only the

largest `gaslib-582` and the smallest `gaslib-40` networks for our computational study. In the following we describe the instances on the two networks in more detail.

`gaslib-582`: As a first step, a list of possible network extensions has to be created. The generation of meaningful extension candidates is an art by itself. Considering that every possible network extension adds binary decisions and in the case of pipes also nonlinearities to the problem, adding too many candidates renders the selection of an optimal subset for a single scenario impossible. Too few extensions on the other hand might not allow a feasible solution for all scenarios and are too inflexible for challenging multi-scenario instances.

For the generation of network extensions we relied on techniques developed in the ForNe project. For the `gaslib-582` network, we used two methods to create extensions. For the first method, we select one scenario and do a simple flow bound propagation. If the flow on some arc is fixed by this procedure, the pressure difference is known and thus the nonlinearity describing the physics of this pipe disappear. Loops of these pipes are expected to be computationally cheap as also in the presence of loops, the nonlinearity can be eliminated easily. We therefore add loops on all pipes with fixed flow value to the extension candidates. The second method proposes new pipes by evaluating the reduction in the transport momentum caused by adding a pipe. The transport momentum for a scenario is computed by disregarding the gas physics and considering only a linear flow problem. The objective is to minimize the sum over all pipes of the product of the length of the pipe with the flow through it. It is a measure how efficiently flow can be routed through the network. New pipes whose end points are far enough apart and whose addition to the network results in the largest reductions of the transport momentum are selected as extension candidates. Meaningful geographic coordinates (or a meaningful distance matrix) are essential for the computation of the transport momentum and we therefore consider this approach especially useful for the realistic `gaslib-582` network.

The `gaslib-582` test set comes with 4227 nominations. To ensure we have a sufficiently large number of infeasible scenarios, we scaled all input and output flows of the nominations by a factor of 1.2. Then, we performed a single-scenario topology optimization to filter out those instances that are still feasible without network extension and those that don't find a solution within the time limit of 600 for sub-problems. The result are 107 scenarios that exhibit a positive objective value after 600 s.

It is worthwhile noting that this procedure is non-deterministic because of the time limit used. The path taken to solve the problem is deterministic, but if the solution is found very close to the time limit, a random disturbance might cause a slowdown and the solution might not be found within the time limit in the next run. It therefore can (and actually does) happen that in a multi-scenario run some scenario does not find a solution in the root node. A limit based on the number of simplex iterations or number of branch-and-bound nodes would eliminate this problem, but is not practical in our application.

In the next step, the 107 scenarios have been grouped together to construct multi-scenario instances. The aim was to construct a set of instances and to make sure that all scenarios participate in the mix. The procedure was to first shuffle the list of scenarios.

Then, assuming that k is the desired number of scenarios in an instance, the first k scenarios in the list are selected and removed from the list. This is repeated until the length of the list of unused scenarios is smaller than k. Then the list of all scenarios is shuffled again and the procedure is repeated. We composed 50 instances of 4, 8, and 16 scenarios each, 25 instances with 32 scenarios and 10 instances with 64 scenarios. Finally, we added one instance with all 107 scenarios.

For the `gaslib-582` test set, the subproblems are solved in parallel with up to 16 threads whenever possible. We used a time limit of 600 s for the subproblems which is reduced to 60 in the 1-opt heuristic. The total time limit was set to 12 h.

`gaslib-40`: In contrast to the `gaslib-582` network which contains 4227 realistic nominations, the `gaslib-40` network only has one nomination. This scenario is quite artificial as it evenly distributes the flow amount over all 3 entries and 29 exits. The motivation to work with this network, however, is not primarily the practical relevance of the instances but to challenge our approach using a network where the subproblems are easier to solve than on the more realistic `gaslib-582`.

We created 2000 nominations using the following algorithm. First, we sampled the number of entries/exits that should have inflow/outflow uniformly between 1 and the number of entries/exits. Then, we randomly picked that number of entries/exits. The total flow from the reference scenario is scaled with a uniformly sampled factor between 0.75 and 2. The resulting scaled total flow is then uniformly distributed among the selected entries and exits, respectively.

For this network, only loops are considered as possible extensions. As before, all pipes where the flow after flow bound propagation of the reference scenario is fixed can be looped. In addition all pipes that are longer than 20 km are in the candidate set.

From the 2000 nominations, a large number is feasible in the original network or still infeasible even with the proposed loops and thus ignored. Furthermore, we select only nominations for which the single-scenario problem is solved to optimality within a time limit of 600 s. We note that many different loops are used in these solutions such that challenging multi-scenario instances can be expected. The resulting 425 nominations are then grouped into 50 instances of 4, 8, and 16 scenarios each, 25 instances with 32 scenarios and 10 instances with 64, 128, and 256 scenarios each.

For `gaslib-40` the parallel solution of single-scenario problems caused buffer overflow errors in the CppAD package for algorithmic differentiation used with SCIP. As these errors are out of our control, these instances are solved purely sequentially which avoids this error.

We used the same time limits of 600 s for the subproblems, 60 s for subproblems within the 1-opt heuristic. For the overall algorithm we first used 12 h for all numbers of scenarios on the `gaslib-582`. As this time limit is found to be short for 64 and more scenarios, we ran the instances with 64, 128, and 256 scenarios also with a time limit of 48 h.

4.3 Results

Table 1 summarizes the performance of our approach. The table is divided into three parts; the first part for results on the `gaslib-582` test set and then two parts for

Table 1 Summary of computational results

Scenarios	Instances	Solution status			MsaS	Union	Nodes		Gap (%)
		Opt	LTL	Subopt			Opt	All	
`gaslib-582` (Time limit 12 h)									
4	50	28	20	2	20	14	3.6	16.4	42.8
8	50	21	26	3	9	7	6.3	24.4	41.9
16	50	8	34	8	0	0	7.2	36.9	27.3
32	25	1	16	8	0	0	7.0	47.2	17.4
64	10	0	6	4	0	0	–	56.5	23.2
107	1	0	1	0	0	0	–	48.0	16.4
`gaslib-40` (Time limit 12 h)									
4	50	49	0	1	12	22	13.7	14.2	44.3
8	50	43	0	7	7	9	24.8	29.7	12.3
16	50	33	0	17	1	2	35.7	38.5	26.4
32	25	11	1	13	0	0	27.4	33.2	24.6
64	10	3	0	7	0	0	23.7	25.4	28.5
128	10	1	0	9	0	0	67.0	16.1	64.1
256	10	0	0	10	0	0	–	4.5	98.6
`gaslib-40` (Time limit 48 h)									
64	10	8	0	2	0	0	56.8	64.1	25.7
128	10	2	2	6	0	0	80.0	60.7	26.5
256	10	1	0	9	0	0	27.0	21.2	49.3

results on the `gaslib-40` test set with time limit 12 and 48 h. The rows are grouped by the number of scenarios considered in each instance. The first two columns then report the number of scenarios and instances in the respective group. The next group of columns partitions the instances w.r.t. the final solution status. The first column in this group gives the number of instances solved to optimality within the time limit. The next column relates to the situation where the branch-and-bound algorithm comes to the point when all y variables are fixed and the subproblems only have to decide feasibility given the fixed extensions. In this case, the remainder of the global time limit might be used for the feasibility problem as otherwise the algorithm can't proceed. The column *LTL* states how many instances did not finish to optimality because the remaining time was used in the subproblems and the algorithm was stuck. The last column (Subopt) in this block gives the instances that were not stuck, but also not solved to optimality. The next two columns analyze the structure of the best solution found by our approach and compare it with the best solutions known for each scenario. For `gaslib-40` the optimal solutions to the single scenario runs are known. For `gaslib-582` the best solution after solving the scenario for 12 h is used as the best known solution for the scenarios. The column *MsaS* states the number of instances where the best solution to the multi-scenario problem constructs the same extensions as the best known solution to one of the scenarios. In this case one scenario dominates the others as the extensions needed by the scenario suffice to ensure feasibility of all

other scenario problems as well. The column *Union* in contrast brings light into the opposite case where the solution of the multi-scenario problem contains the union of the extensions constructed in the single scenarios. An instance appears in both columns MsaS and Union if all scenarios select the same extensions. The last two groups of columns show the average number of nodes, split by the instances that were solved to optimality and all instances, and the average gap of those instances that were not solved to optimality. The gap is taken from SCIP where for primal bound p and dual bound d it is computed by

$$\left| \frac{p-d}{\min(|p|, |d|)} \right|$$

and set to 0 if $p = d$ and to ∞ if $\min(|p|, |d|) = 0$.

While our approach is able to solve 28 out of 50 instances, or 56% of the instances, with 4 scenarios on the realistic `gaslib-582`, this percentage decreases with increasing number of scenarios. At the same time the number of instances where the algorithm gets stuck because one of the subproblems can't properly decide feasibility is constantly high. At the maximum more than 11 of the 12 h are spent trying to decide the feasibility of one scenario. On this test set there is also a remarkable difference between the number of nodes of those instances that could be solved to optimality, where the average number is at most 7.2 nodes for 16 scenarios, to those instances that hit the time limit, where the number goes up to 56.5 for 64 instances. This shows that the instances that could be solved don't need much branching and that our heuristics do a good job in finding the optimal solution. The high numbers of nodes over all instances are because at many nodes some scenarios don't find a feasible solution, but also don't prove infeasibility or even provide good bounds. In this case, our branching mechanism branches on some unfixed variable. In general, the number of nodes is quite low compared to what we are used to from branch-and-bound MILP or MINLP solvers. This shows that the solutions of the single scenarios provide good indications for the structure of multi-scenario solution, even though their solution is rather time consuming. Also good solutions are found very early in the tree. The primal bound makes pruning and propagation very effective, especially as solutions can typically use only very few extensions because otherwise the cost is higher than the dual bound.

Although being large, the average gap values reported on the `gaslib-582` are quite satisfactory. One has to keep in mind that these instances are very challenging so that large gaps need to be expected. Note that the gap is computed as the average of only those instances that are not solved to optimality and again we have to see them in the light of the difficulty of the problem. Particularly, an average gap of 23.2% on the instances with 64 scenarios and 16.4% gap on the instance with all 107 scenarios shows that the solutions are of high quality. Overall, the ability to provide bounds on the solution quality and, if possible, a certificate for optimality is an advantage of our approach.

High numbers in the MsaS and Union columns indicate that the structure of the optimal solution is such that a manual approach might find a good or even the optimal solution. In this case, either one scenario dominates the solution or the solution consists of the union of all the constructed extensions in the single scenarios; a situation that

is easily recognized in a manual fashion and which renders a more sophisticated approach unnecessary. On both test sets, many optimal solutions to the instances with 4 and 8 scenarios have such a structure. In the instances with 4 scenarios 30 instances on `gaslib-582` and 27 `gaslib-40` are either dominated by one scenario or the multi-scenario solution is the union of the extensions in the best single-scenario solutions. We note that both can also happen simultaneously. The number goes down to 15 and 11 for `gaslib-582` and `gaslib-40`, respectively, when 8 scenarios are considered and completely disappear for higher number of scenarios except for 3 instances with 16 scenarios on the `gaslib-40` network. This shows that for a few scenarios only a manual planning approach based on solving the single-scenarios could provide good or even optimal solutions. For larger numbers of scenarios the manual approaches are unlikely to find good solutions as there the synchronization between the scenarios becomes more important. Of course manual planning approaches also lack quality guarantees in terms of gap to the best possible solution which our approach provides.

On the smaller `gaslib-40`, all but one instances with 4 scenarios can be solved to optimality. Then the percentage of instances solved to optimality decreases, but still 3 out of 10 instances with 64 scenarios are solved within the time limit. Still one instance with 128 scenarios is solved, but we observe a strong decrease in the number of nodes processed which indicates that the time limit is very short for these large numbers of scenarios. Note that on `gaslib-40` on average 24 nodes are used in the instances that are solved to optimality, but with 128 and 256 only 16.1 and 4.5 nodes are processed on average, respectively. The large average gaps in these groups of instances then also do not surprise. When increasing the time limit from 12 to 48 h many more nodes are processed and 8 out of 10 instances with 64 scenarios, 2 with 128 and 1 instance with 256 scenarios are solved to optimality within the increased time limit. Also the average gap is reduced considerably giving with 25.7%, 26.5%, and 49.3% for 64, 128, and 256 scenarios, respectively, very reasonable results. On this test set, as intended the subproblems can be solved much more reliably and the algorithm is stuck only on one instance where feasibility of the subproblem can't be decided in more than 10 h.

Table 2 analyses the components of the algorithm that produce primal solutions. The structure of the table is similar to Table 1 and the numbers refer to the same experiments as in Table 1. The column *Sols* states the average number of solutions that have been found in the instances of the respective group. Then three blocks analyze the heuristic components of the algorithm. In each block, we report the number of instances where the component found at least one solution and where it found the best solution (columns *Succ* and *Best*, respectively), the average number of solutions found (column *Sols*), and the average time spent per instance in the respective heuristic (column *Time [s]*). The first block with header *Subprob* belongs to the solution that is derived by constructing all extensions that are used in the best solutions of the scenarios, i.e., by setting $y = \max_{\omega \in \Omega} x_e^{\omega}$. This approach finds solutions for all instances (in one instance which is not marked as success, all scenarios use exactly the same extensions and thus the heuristic is not called as the relaxation already found the optimal solution). The second block *1opt* belongs to the highly effective 1opt heuristic. Even though it can be time-consuming, it finds plenty of solutions which often constitute big improvements. It is also able to find the best known solutions for a large number of instances. The

Table 2 Statistics about solutions found by the different parts of the algorithm

Scen.	Inst.	Sols	Subprob				1opt				Best known			
			Succ	Best	Sols	Time (s)	Succ	Best	Sols	Time (s)	Succ	Best	Sols	Time (s)
`gaslib-582` (Time limit 12 h)														
4	50	3.2	50	7	1.1	2.0	42	39	2.4	144.8	6	2	1.0	0.2
8	50	4.7	50	13	1.4	4.8	45	29	3.3	399.2	11	4	1.1	1.0
16	50	6.0	50	13	2.1	11.7	47	28	3.6	940.1	24	8	1.1	4.1
32	25	8.0	25	5	2.4	27.1	22	8	5.1	2194.6	17	12	1.6	17.4
64	10	6.5	10	7	2.6	57.3	9	0	3.1	2236.2	6	3	1.8	34.7
107	1	6.0	1	1	2.0	103.9	1	0	1.0	2094.8	1	0	3.0	152.3
`gaslib-40` (Time limit 12 h)														
4	50	3.0	49	24	2.3	0.4	17	13	1.3	24.6	18	12	1.0	0.2
8	50	6.5	50	10	4.7	0.7	33	29	1.9	64.5	29	11	1.0	0.8
16	50	10.5	50	4	6.5	1.5	43	33	3.3	345.8	41	11	1.3	2.5
32	25	14.3	25	2	8.0	3.0	24	21	5.4	951.4	22	2	1.2	4.6
64	10	13.0	10	0	6.1	5.5	10	8	5.3	1336.2	10	2	1.6	11.1
128	10	15.3	10	0	7.4	12.2	9	5	6.3	4377.3	10	5	2.2	26.8
256	10	10.6	10	0	3.5	22.2	10	8	5.5	10174.9	10	2	1.6	36.5
`gaslib-40` (Time limit 48 h)														
64	10	20.0	10	1	11.2	6.9	10	7	7.1	1935.3	10	1	1.6	17.9
128	10	21.7	10	0	13.2	12.5	9	6	6.7	5782.0	10	4	2.5	44.0
256	10	14.4	10	0	6.5	22.3	10	5	6.0	10732.0	10	5	1.9	52.3

Table 3 Size of the deterministic equivalent formulations

Scenarios	Variables		Constraints			
	Total	Binary	Total	Linear	SignPower	Indicator
gaslib-582						
4	13740	624	12336	8296	2568	1472
8	27480	1248	24672	16592	5136	2944
16	54960	2496	49344	33184	10272	5888
32	109920	4992	98688	66368	20544	11776
64	219840	9984	197376	132736	41088	23552
107	367545	16692	329988	221918	68694	39376
gaslib-40						
4	3044	168	3524	1948	1256	320
8	6088	336	7048	3896	2512	640
16	12176	672	14096	7792	5024	1280
32	24352	1344	28192	15584	10048	2560
64	48704	2688	56384	31168	20096	5120
128	97408	5376	112768	62336	40192	10240
256	194816	10752	225536	124672	80384	20480

last block *Best Known* corresponds to the approach where the best known solution is computed by the auxiliary MILP (15). This approach is also successful on a broad range of instances and in particular on the most difficult instances with larger numbers of scenarios where it often finds the best solution. The short running times show that the MILP is solved without problems. Overall, we conclude that all proposed heuristics contribute to the success of the algorithm.

As an alternative solution approach, one could also use the deterministic equivalent formulation solve it with an available state-of-the-art solver for mixed-integer nonlinear optimization tasks. Table 3 summarizes the size of this formulation for the resulting multi-scenario instances. The table is split into two parts, each for one network topology. Each line contains the number of variables and constraints for the deterministic equivalent for the number of scenarios that is given in the first column. We report the total number of variables and the number of binary variables. For the constraints, we report the total number of constraints and break them down into the three relevant classes for our problem which are linear constraints, SignPower constraints, i.e., constraints of type (2), and indicator constraints which are used to model implications such as (4)–(8). It is to be expected that solving the deterministic equivalent formulation directly fails for a non-trivial number of scenarios.

Indeed, we performed this experiment on `gaslib-40` using SCIP version 3.1 and Cplex version 12.5.1 on the same hardware as before. We set a time limit of 12 h and a memory limit of 57.6GB to leave 10% of memory for the operating system. On the smallest instance with 4 scenarios still 13 out of the 50 instances could be solved to optimality. The remainder of 37 instances timed out and on 22 instances no feasible solution was found. As expected when increasing the number of scenarios,

the performance deteriorates. For 8 scenarios, one instances is solved to optimality and the remainder ended without feasible solution. Beyond that SCIP was not able to find a feasible solution for any instance. In contrast, our approach determines feasible solutions separately for each scenario which is usually easier than for the full deterministic equivalent.

In robust and stochastic programming, decomposition methods are known to scale better with increasing number of scenarios and there is a point, where the scenario-expanded model even does not even fit into the memory of one machine. Our decomposition approach solves only the single-scenario problems and their solution within one node of our branch-and-bound tree is parallelized and could furthermore be parallelized rather easily beyond one machine.

5 Conclusion

We presented a method for gas network planning with multiple demand scenarios. The computational experiments show that our approach can provide good solutions with reasonable quality guarantees on realistic network topologies. A large range of instances is solved to proven optimality. We showed that our approach clearly outperforms solving the corresponding deterministic equivalent formulations by standard methods.

Even though developed in the context of gas network planning, the limited assumptions on the underlying problem structure suggest the generalization to other capacity planning problems in the future. Recall that we only assume that the extensions form a hierarchy where higher levels, i.e., more expensive extensions, have all the functionality of all lower levels and the availability of a black box solver for the adjusted single-scenario problems (15a). Singh et al. (2009), for example, use the same framework and Dantzig–Wolfe decomposition on model (15) to approach a rather generic capacity expansion problem. A comparison to Lagrangian decomposition based methods in the spirit of Dual Decomposition by Carøe and Schultz (1999) would also be of interest.

While it is an advantage of our approach that it assumes no particular structure in the subproblems, the algorithm can be enhanced by using more information about the solution space of the subproblems. For gas networks without active devices in the original network and with only loops as extensions candidates, Humpola (2014) describes inequalities that enforce that a certain amount of loops has to be constructed in order to make a scenario feasible. Using inequalities of this type that are found during the solution of the subproblems to propagate bounds on the y variables or to steer the search could be promising ways to improve the algorithm in this special application and is subject to future research.

Acknowledgements thank The authors thank the anonymous referees for their valuable and constructive feedback. The authors are grateful to Open Grid Europe GmbH (OGE, Essen/Germany) for supporting this work and furthermore thank the DFG for their support within projects B06 and Z01 in CRC TRR 154. This research was performed as part of the Energie Campus Nürnberg and supported by funding through the 'Aufbruch Bayern (Bavaria on the move)' initiative of the state of Bavaria. The authors thank all collaborators in the FORNE project and all developers of `Lamatto++`.

References

Achterberg T (2009) SCIP: solving constraint integer programs. Math Program Comput 1(1):1–41

Ben-Tal A, Nemirovski A (1999) Robust solutions of uncertain linear programs. Oper Res Lett 25(1):1–13

Ben-Tal A, Goryashko A, Guslitzer E, Nemirovski A (2004) Adjustable robust solutions of uncertain linear programs. Math Program 99(2):351–376

Ben-Tal A, El Ghaoui L, Nemirovski A (2009) Robust optimization. Princeton series in applied mathematics. Princeton University Press, Princeton

Berthold T, Heinz S, Vigerske S (2012) Extending a CIP framework to solve MIQCPs. In: Lee J, Leyffer S (eds) Mixed integer nonlinear programming, volume 154, part 6 of the IMA volumes in mathematics and its applications. Springer, Berlin, pp 427–444

Bertsimas D, Sim M (2003) Robust discrete optimization and network flows. Math Program 98(1):49–71

Birge JR, Louveaux R (2011) Introduction to stochastic programming. Springer series in operations research and financial engineering. Springer, New York

Bonami P, Lodi A, Tramontani A, Wiese S (2015) On mathematical programming with indicator constraints. Math Program 151(1):191–223

Carøe CC, Schultz R (1999) Dual decomposition in stochastic integer programming. Oper Res Lett 24(1):37–45

Collins M, Cooper L, Helgason R, Kennington J, LeBlanc L (1978) Solving the pipe network analysis problem using optimization techniques. Manage Sci 24(7):747–760

Fodstad M, Egging R, Midthun K, Tomasgard A (2016) Stochastic modeling of natural gas infrastructure development in Europe under demand uncertainty. Energy J 37

Fügenschuh A, Hiller B, Humpola J, Koch T, Lehman T, Schwarz R, Schweiger J, Szabó J (2011) Gas network topology optimization for upcoming market requirements. In: IEEE proceedings of the 8th international conference on the European energy market (EEM), 2011, pp 346–351

Gabriel SA, Rosendahl KE, Egging R, Avetisyan HG, Siddiqui S (2012) Cartelization in gas markets: studying the potential for a "gas OPEC". Energy Econ 34(1):137–152

Gaslib (2013) A library of gas network instances. http://gaslib.zib.de

Geißler B, Martin A, Morsi A (2014) Lamatto++. http://www.mso.math.fau.de/edom/projects/lamatto.html

Geoffrion A (1972) Generalized benders decomposition. J Optim Theory Appl 10(4):237–260

Gorissen BL, Yanıkoğlu İ, den Hertog D (2015) A practical guide to robust optimization. Omega 53:124–137

Gotzes C, Heitsch H, Henrion R, Schultz R (2016) On the quantification of nomination feasibility in stationary gas networks with random load. Math Methods Oper Res 84(2):427–457

Hellemo L, Midthun K, Tomasgard A, Werner A (2013) Multi-stage stochastic programming for natural gas infrastructure design with a production perspective. In: Stochastic programming applications in finance, energy, planning and logistics. World scientific book chapters, chapter 10. World Scientific Publishing Co. Pte. Ltd., pp 259–288

Humpola J (2014) Gas network optimization by MINLP. Ph.D. thesis, Technische Universität Berlin

Koch T, Hiller B, Pfetsch ME, Schewe L (eds) (2015) Evaluating gas network capacities. SIAM-MOS series on optimization. SIAM, Philadelphia

Kouvelis P, Yu G (eds) (1997) Robust discrete optimization and its applications. Series on non-convex optimization and its applications. Kluwer, Dordrecht

Li X, Tomasgard A, Barton PI (2011) Nonconvex generalized Benders decomposition for stochastic separable mixed-integer nonlinear programs. J Optim Theory Appl 151(3):425–454

Li X, Sundaramoorthy A, Barton PI (2014) Nonconvex generalized benders decomposition. In: Rassias TM, Floudas CA, Butenko S (eds) Optimization in science and engineering. Honor of the 60th birthday of Panos M. Pardalos. Springer, New York, pp 307–331

Li X, Tomasgard A, Barton PI (2016) Natural gas production network infrastructure development under uncertainty. Optim Eng 1–28

Liebchen C, Lübbecke M, Möhring R, Stiller S (2009) The concept of recoverable robustness, linear programming recovery, and railway applications. In: Ahuja RK, Möhring RH, Zaroliagis CD (eds) Robust and online large-scale optimization: models and techniques for transportation systems. Springer, Berlin, pp 1–27

Midthun KT (2007) Optimization models for liberalized natural gas markets. Ph.D. thesis, Norwegian University of Science and Technology NTNU

Odom FM, Muster GL (2009) Tutorial on modeling of gas turbine driven centrifugal compressors. Technical report 09A4, Pipline Simulation Interest Group

Percell PB, Ryan MJ (1987) Steady-state optimization of gas pipeline network operation. In: Proceedings of the 19th PSIG annual meeting, 10

Pfetsch ME, Fügenschuh A, Geißler B, Geißler N, Gollmer R, Hiller B, Humpola J, Koch T, Lehmann T, Martin A, Morsi A, Rövekamp J, Schewe L, Schmidt M, Schultz R, Schwarz R, Schweiger J, Stangl C, Steinbach MC, Vigerske S, Willert BM (2014) Validation of nominations in gas network optimization: models, methods, and solutions. Optim Methods Softw

Ríos-Mercado RZ, Wu S, Scott LR, Boyd EA (2002) A reduction technique for natural gas transmission network optimization problems. Ann Oper Res 117(1):217–234

Ríos-Mercado RZ, Borraz-Sánchez C (2015) Optimization problems in natural gas transportation systems: a state-of-the-art review. Appl Energy 147:536–555

Sahinidis NV, Grossmann IE (1991) Convergence properties of generalized Benders decomposition. Comput Chem Eng 15:481–491

Schweiger J (2016) Gas network extension planning for multiple demand scenarios. In: Lübbecke M, Koster A, Letmathe P, Madlener R, Peis B, Walther G (eds) Operations research proceedings 2014: selected papers of the annual international conference of the German operations research society (GOR). RWTH Aachen University, Germany, 2–5 Sept 2014. Springer International Publishing, Cham, pp 539–544

Schweiger J (2017) Exploiting structure in nonconvex quadratic optimization and gas network planning under uncertainty. Ph.D. thesis, Technische Universität Berlin

Singh KJ, Philpott AB, Wood RK (2009) Dantzig–Wolfe decomposition for solving multistage stochastic capacity-planning problems. Oper Res 57(5):1271–1286

Soyster AL (1973) Convex programming with set-inclusive constraints and applications to inexact linear programming. Oper Res 21:1154–1157

Tomasgard A, Rømo F, Fodstad M, Midthun K (2007) Optimization models for the natural gas value chain. In: Hasle G, Lie K-A, Quak E (eds) Geometric modelling, numerical simulation, and optimization: applied mathematics at SINTEF. Springer, Berlin, pp 521–558

Vigerske S (2012) Decomposition in Multistage stochastic programming and a constraint integer programming approach to mixed-integer nonlinear programming. Ph.D. thesis, Humboldt-Universität zu Berlin

Vigerske S, Gleixner A (2016) SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. Technical report 16-24, ZIB, Takustr.7, 14195 Berlin

Wallace SW (2010) Stochastic programming and the option of doing it differently. Ann Oper Res 177(1):3–8